2017-05-22

Human Fall and Activity Detection, and Muscle Spasm Identification

Faisal Sikder
University of Miami, faisal.sikder@gmail.com

UNIVERSITY OF MIAMI

HUMAN FALL AND ACTIVITY DETECTION, AND MUSCLE SPASM IDENTIFICATION

By

Faisal Sikder

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

May 2017

UNIVERSITY OF MIAMI

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

HUMAN FALL AND ACTIVITY DETECTION, AND MUSCLE SPASM IDENTIFICATION

Faisal Sikder

Approved:

Dilip Sarkar, Ph.D. Associate Professor of Computer Science	Hüseyin Koçak, Ph.D. Professor of Computer Science
Geoff Sutcliffe, Ph.D. Professor of Computer Science	Guillermo Prado, Ph.D. Dean of the Graduate School
Kamal Premaratne, Ph.D. Professor of Electrical and Computer Engineering	

SIKDER, FAISAL (Ph.D., Computer Science)

Human Fall and Activity Detection, and Muscle Spasm Identification (May 2017)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Dilip Sarkar.
No. of pages in text. (99)

While more and more inexpensive devices with embedded sensors are introduced to improve our living, the challenge is to process and analyze large datasets they collect for identification of vital events and activities. Datasets from wearable motion sensors are used to detect and monitor human fall and activities of daily life (ADLs). Existing methods for detection of fall and ADLs from motion datasets employ feature extraction and machine learning, but they have high classification errors. Thus, they produce false alarms for fall and wrong identifications of ADLs. Similar to motion dataset problem, detection of involuntary muscle activities from large EMG datasets (collected from spinal cord injured individual) is a challenging task. Recent studies have developed locations identification algorithms for spasms, motor units, and contractions on individual channel of the EMG datasets. It is important to know how and when repetitive muscle contractions happen in multiple muscles at the same time and is there any any reason for this involuntary co-activity.

We demonstrate that the k-means clustering algorithm can semi-automatically extract training examples from motion data. We also propose one- and two- layer classification networks using neural networks and softmax regression. Moreover, we propose a distance measure, called Log-Sum Distance, for evaluating difference between two sequences of

positive numbers[1]. We use the proposed Log-Sum Distance measure to develop algorithms for recognition of human activities from motion data. The sequences of m positive numbers for Log-Sum Distance are residual sum of squares errors produced from modeling m motion time-series with multiple linear regression method. To reduce incorrect classification we define a threshold test and use it in our proposed novel algorithm. Log-Sum Distance measure also has been employed to identify the locations for repetitive muscle contractions in one or multiple channels of EMG recordings. We also propose a method to identify the muscle that triggers the first contraction in an identified region. We extract features from EMG data using wavelet filter and decomposing co-variance matrix for eigenvector. Experiments with fall detection, ADLs recognition and monitoring, and repetitive contractions identification methods proposed here show very high accuracy rates with different benchmark datasets. The proposed use of threshold values for classification of activities decreased incorrect classification rates. In summary, this work introduces novel methods and the state-of-the-art development and training of wearable devices for fall and ADLs recognition and monitoring. It also extends the involuntary muscle activities identification across multiple channels.

[1]Prof. Kamal Premaratene brought to our attention that the proposed measure can be derived from Bregman divergence [1].

To my parents

Acknowledgements

There are many people that I would like to express my sincere thanks, without which this dissertation would not be completed. First of all, I would like to start thanking my advisor Professor Dilip Sarkar for his support, guidance, and freedom to work on this dissertation. I believe his personality and technical capability was an essential factor for me to finish this. I would also like to thanks my committee, Professor Hüseyin Koçak, Professor Geoff Sutcliffe, and Professor Kamal Premaratne for their excellent support, guidance and comments throughout my research. I would like to thank Dr. Saminda Abeyruwan for his enlightening discussions and knowledge sharing. I am very thankful to Department of Computer Science for providing me with graduate assistantship to support my studies, without which this journey would not be possible. I would like to thanks everyone who helped me throughout my course of research and providing me with feedback. Finally, I would like to thanks my parents, wife, and other family members for all their support, love, and care throughout all my life.

FAISAL SIKDER

University of Miami

May 2017

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Different types of sensors are promising us secure and better everyday living conditions [3]. As more and more sensors are introduced, larger volumes of sensor datasets have to be processed to extract essential information from them [4]. It is challenging to analyze and discover information from large datasets efficiently with microcontrollers in the devices. In this dissertation, our goal is to develop algorithms for extracting information from sensor datasets that are useful for improving daily human living.

To be more specific, we address the followings: (i) How can daily living be improved using sensor information? (ii) How well can existing state-of-the-art learning models identify patterns from sensor datasets for classification of activities in everyday living? and (iii) How can the performance of existing algorithms for extracting information from large volume of sensor datasets can be improved?

This dissertation quest to answer the above questions. Our goal is to find methods and algorithms for extracting information from large datasets collected by sensors embedded in wearable devices for monitoring activities to improve daily living.

1.2 Overview and Scope of the Work

In this dissertation, we present works related to sensors data analysis to identify patterns. Sensors are at the heart of the modern automation processes and health care systems. In this research, we developed and tested novel algorithms for human activity of daily living (ADL), fall, and posture detection and monitoring. These algorithms were evaluated with motion sensors datasets from several sources. We also used novel algorithms to identify repetitive muscle contractions regions where muscle contraction is present in one or multiple multiple channels of electromyographic (EMG) datasets for spinal cord injured (SCI) individual.

1.2.1 Fall and Posture Detection from Motion Senor Datasets

Inexpensive wearable motion-sensing devices have shown great promise for fall detection and posture monitoring [5]. But two major problems still exist and have to be solved for making them acceptable for regular use: i) a framework for the development of firmware, and ii) software for detection of fall and monitoring postures. To solve the first problem, we design and implement a generic framework for developing firmware for wearable devices. To address the second problem, we have found that from motion data the k-means clustering algorithm can semi-automatically extract training examples for machine learning algorithms. We use extracted examples to train and evaluate several one- and two-layer classification networks i) to monitor non-fall activities and ii) to detect fall events. The proposed classification-networks are combinations of neural networks and softmax regression.

1.2.2 Distance-Based Novel Algorithm for Detection of ADLs

For detection of human activities using motion data many techniques employ feature extraction and machine learning. The techniques we use in the previous section are based on existing example-based machine learning techniques. A drawback of these techniques is that any input will be classified into one of the predefined classes. Here in this project, we propose a distance measure, called Log-Sum Distance, for computing distance between two sequences of positive numbers. We use the proposed Log-Sum Distance measure to develop algorithms for recognition of human activities from motion data.

Log-Sum distance algorithm measures element wise distances between two sequences of positive numbers. The distance measured by this algorithm is non-negative. The sequences of m positive numbers used in our algorithm are *residual sum of squares* errors produced from modeling m motion time-series with *multiple linear regression method*. To reduce incorrect classification we introduce a threshold test and use it in our proposed novel algorithms. Also, we define an optimization function and use it for computing optimal values for thresholds.

1.2.3 Repetitive Muscle Contractions Detection in EMG Recordings

Paralyzed SCI patients can get affected by involuntary muscle activities in their paralyzed muscles. [6,7]. Repetitive muscle contractions are common in paralyzed SCI patients and it can occur throughout the day and night. Sometime this involuntary contractions are manageable by the individual, but in other times it interferes with individuals daily livings [8, 9]. By analyzing long-term EMG recordings of the paralyzed muscle, it might be possible to measure the muscle contraction frequencies, durations, co-activity and also

their severity. Different methods have been applied to EMG recodings for detection of contractions with varying degree of accuracies [10–15].

In this work, we use the Log-Sum Distance measure to identify regions where it indicates the presence of repetitive muscle contractions or EMG bursts in one or more of the five different channels in long-term EMG recordings. This five channels of EMG is collected from five different leg muscles of paralyzed individuals. In one or more of the five different leg muscle's EMG data, we identify involuntary muscle contraction which is indicated by periodic burst of EMG signal. Then we mark the start location of the EMG burst (if exist) in each individual leg muscle. By comparing the start time in an identified region, we can identify the muscle that started the first EMG burst. To the best of our knowledge, no studies have reported methods that identify regions where more than one muscles show existence of repetitive muscle contractions.

1.3 Contributions of this Dissertation

In summery, key contributions of this dissertation are:

- We have developed a method for semi-automatic extraction of training examples from motion sensors readings. We used our technique to extract training examples from motion data, and used these training examples to train one- and two-layer classification networks to detect fall and ADLs. Extensive evaluations of the trained networks demonstrated that two-layer networks achieved perfect detections of fall events and identification of ADLs.

- To measure distance between two sequences of positive numbers we made an independent finding of generalized Kullback-Leibler divergence (KL-divergence). We

named it Log-Sum Distance. It is possible to drive this Log-Sum Distance from Bregman Divergence[1]. We derive Log-Sum Distance measure from Bregman divergence. We also establish relation with the proposed Log-Sum Distance measure and KL-divergence.

- We have used Log-Sum Distance measure to develop algorithms for recognition of ADLs from motion data. In our ADL recognition algorithms we introduced threshold parameters for reducing incorrect-classification rates. Machine learning technique was used to find optimal values of the threshold parameter.

- We also used Log-Sum distance measure to identify regions which indicate the presence of repetitive muscle contractions in one or more of the five different channels in long-term EMG recordings. Moreover, we developed a method to identify muscle that starts the first contraction in the identified region and it might has initiated muscle contractions in that region.

1.4 Overview of Chapters

The rest of this dissertation is organized as follows. Chapter 2 provides a brief review of previous work related to our research. In Chapter 3, we present our method for semi-automatic extraction of training examples from motion data. To show efficacy of our semi-automatic training example extraction method, we extracted training examples from motion datasets of fall events and ADLs. We used these training examples to train one- and two-layer classification network for identification of fall events and ADLs. Extensive evaluations of the trained demonstrated that two-layer networks correctly identified all fall

[1]Prof. Kamal Premaratene brought to our attention that the proposed measure can be derived from Bregman divergence [1]

events and classified ADLs. In Chapter 4, first we present our Log-Sum Distance measure, and then a novel algorithm for identification of ADLs. We also present several other algorithms for comparing performance of the new algorithm. Finally, we provide extensive evaluation results using two different datasets. Performance of the Log-Sum Distance based ADL identification algorithm motivates us to use it to detect repetitive muscle contractions regions where contractions might be present in one or more channels of long-term EMG recordings. We present the algorithms for locating repetitive muscle contractions in Chapter 5. Finally, in Chapter 6, we present a summary of our novel contributions, and possible future directions for continuing the work.

CHAPTER 2

Literature Review

2.1 Introductory Remarks

In this chapter, we briefly review previous work directly related to this dissertation, and also some concepts that are related to our research. Section 2.2 reviews related previous works on human activity recognition and monitoring and fall detection. In the beginning of Section 2.3, we discuss some fundamental concepts related to muscle spasm. After that we discuss about the current state-of-the art spasm study literatures using electromyographic (EMG) data.

2.2 Human Fall, Posture, and Activity Recognition and Monitoring

The availability of inexpensive miniature micro-electromechanical system (MEMS) is making a revolution toward wearable devices [4]. And Motion sensor is in the heart of wearable devices. A motion sensor may include one or more of the following: (i) accelerometer, (ii) gyroscope, and (iii) magnetometer [5]. These motion sensors are embedded in such devices as cellphone, watches, toys, tablets, cars, etc. As these sensors are

very widely used in modern systems, researchers are using motion data to enhance many different everyday activities, including daily activity monitoring [5, 16].

Fall is a major cause of injury to the elderly population [17]. The Centers for Disease Control and Prevention (CDC) reported, "In 2010, falls among older adults cost the U.S. health care system $30 billion in direct medical costs, when adjusted for inflation." By 2020 it is expected to reach $67.7 billion [18]. Availability of inexpensive motion sensing devices have enabled researchers to study their potential application to assess fall risk and monitoring activities of daily livings (ADLs). Earlier studies demonstrated feasibility of various miniature wired or wireless devices, and recent studies have focused on extracting knowledge motion data. Results from past studies have proven that hardware for MEMS sensor based intelligent devices to (i) assess fall-risks, (ii) monitor ADLs, and (iii) identify fall events have reached maturity and can be assembled from off-the-shelf hardware.

The major steps for monitoring ADLs and recognition of fall events using wearable motion sensors are: (i) data collection, (ii) data preprocessing, (iii) feature extraction, and (iv) fall or activity identification [16]. Challenges are still exist in feature extraction and activity detection steps. Accuracy of the existing systems depends on many factors including, number of devices, number and type of sensors, and the recognition method used. With very few exceptions, a trained system classify an input data into one of the trained activities. A better and more desirable system would reduce incorrect recognition, maybe by having a "not sure" state.

Based on the platforms used, fall and activity detection methods can be broadly categorized into two groups: (i) inexpensive wearable embedded devices, and (ii) smart-phones. Different feature extraction and learning methods have been used in different researches to identify and monitor fall and/or daily activities. On the other hand, three major research ar-

eas for sensor-based activity detection systems are: (i) monitoring activities of daily living (ii) sport and training (iii) health care.

2.2.1 Fall Detection Using Wearable Sensors

Ojetola et al. [19] report a method for detecting four fall events – forward, backward, right, and left – using readings from both accelerometer and gyroscope. Their approach utilizes a decision tree to learn and classify falls and ADLs. It identified fall events with 81% precision and 92% recall rate. A system to detect events that may cause trauma and disabilities was proposed in [20]. It used readings from MEMS accelerometer embedded in a wearable wireless device. They use SVM for the classification of different events.

The method in [21] uses data from tri-axial accelerometer and gyroscope sensors, embedded in a necklace, for classifying the behavior and posture of human subjects. Their method distinguishes between ADLs and falls, with 80% or higher sensitivity and 100% specificity. Their experiments include both normal ADLs such as standing, sitting in the chair or floor, laying, walking, running, going upstairs/downstairs, and bending, as well as abnormal events such as falling forward, backward, leftward, rightward, and fall on the stairs.

2.2.2 Posture and Activity Recognition and Monitoring Using Wearable Sensors

Bao and Intille [22] used five small-biaxial wire-free accelerometers that were attached on the left bicep, the right wrist, the left quadriceps, the right ankle, and the right hip of the subjects for collecting motion data. They used motion data for recognizing 20 activities, including walking, riding elevator, strength training, and bicycle . They evaluated the

performance of a decision table, instance-based learning, a decision tree, and naïve Bayes classifiers. The results revealed that the decision-tree based method performed best with 84% accuracy. Similar efforts have been reported to detect human motions using motion tracking (see [23–27]).

In [28], high-level fuzzy Petri-net based systems were proposed and evaluated for the analysis and detection of normal human actions such as sitting-down, squatting, walking, running, and jumping as well as abnormal events such as falling forward, backward, sideways, and vertical. They reported 94% accuracy for fall-detection. However, the proposed system could not accurately detect some complex situations and movements such as falling down from stairs, multiple collisions, or temporal unbalanced motions.

Ward et al. [29] developed a framework for continuous recognition of wood workshop activities from sound data (sampled at 2kHz) and 3-axis accelerometers (sampled at 100Hz). Microphones and 3-axis accelerometers at wrist and upper arm of the body were used to collect data. Sound intensity from two different locations have been used to separate activities from continuous data stream. After that, activity detection is done on the separated activity data segments using Linear Discriminant Analysis (LDA) on the sounds data and Hidden Markov Models (HMMs) and Logistic Regression (LR) on the accelerometers data. They reported recognition accuracy of 98%, 95% and 87% for user-dependent, user-adapted, and user-independent categories, respectively.

Yang et al. [30] developed a distributed recognition framework to classify continuous human actions using wearable motion sensors. Their wearable devices have sensors for 3-axis accelerometer and 2-axis gyroscope. They attached five sensors on 5 different locations of human body that forms a *wireless sensor network* (WSN) and collected data from these devices. They have used a mixture subspace model for training.

Their experimental evaluations achieved on an average 93.46% accuracy for all activities. They have also create a publicly available wearable action recognition database (WARD). This database has data from twenty different subjects for thirteen activities.

The method in [21] used data from a wireless device attached to a necklaces. The device has 3-axis accelerometer and gyroscope sensors. Their method identified ADLs and falls, with 80% or higher sensitivity and 100% specificity. Their experiments include both normal ADLs such as standing, sitting in the chair or floor, laying, walking, running, going upstairs/downstairs, and bending, as well as abnormal events such as falling forward, backward, leftward, rightward, and fall on the stairs.

Methods presented in [31] are using data from 12 *Orientation Sensors* from ETH to monitor kinematic changes evoked by fatigue from running. These 12 *Orientation Sensors* captured full body movement, while 21 subjects with different skill level and running techniques, performed many running activities in the outdoor tracks and on the treadmills. They identified parameters that characterize fatigue during running. They also discovered that treadmill running is not always the same as the outdoor running.

In the health care domains wearable sensor devices to monitor patients at home or at hospitals are being investigated. Mariani et al. [32] have developed a small on shoe wearable device with 3-axis accelerometer and 3-axis gyroscope for gait and turning assessment of patients with Parkinson's disease (PD). They have used Spatio-Temporal Analysis on the both 3-D orientation and velocity. Their evaluation of the device on ten different PD patients and ten different age-matched non-patients identified PD accurately.

Mehta et al. [33] has developed a voice health monitoring system using wearable sensor and smartphone. They have put an wearable accelerometer on the neck skin above the collarbone and used smartphone to collect data. They have measured vocal function

and airflow using those sensor reading. Their smartphone based GUI can do the analysis and help the user to monitor vocal health.

2.2.3 Research using Smart-phone's Motion Sensors

There is an increasing interest in using smart-phones to detect activities related to health care. Using only readings from an accelerometer of a smart-phone, five human actions – walking, running, standing up, sitting down, and jumping – were analyzed in [34]. They compared the acceleration characteristics of these actions with those of three different fall events to infer the direction of a fall. The method recognizes a fall activity when a predetermined set of conditions are satisfied. However, the method do not provide any prediction or hint that a fall could occur in the near future.

It was show in [35], that the sensors in the smart-phones from different manufactures record significantly different range of values for identical tasks. To overcome this problem, they trained a SVM with extracted features from raw accelerometer readings and the directional changes of the constraining force exerted on an accelerometer to detect fall events. They compared these events to non-fall activities such as walking, running, jumping, and some actives which resembles falls such as sitting down on a chair. Their method detected fall events with an accuracy of 84.8%.

Methods that detect both simple and complex activities using readings from an accelerometer and a gyroscope of an Android smart-phone were studied in [36]. Simple activities include biking, climbing stairs, driving, running, sitting, standing, walking, and the phone-not-on-the-person. Complex activities include cleaning, cooking, medication, sweeping, washing hands, and watering plants. Both simple and complex activities were combined together for recognition. Six different classifiers – multi-layer perceptron (MLP),

naïve Bayes, Bayesian network, decision table, best-first tree, and K-star – were trained using the same set of features extracted from the datasets. They obtained 93% accuracy for simple activities (with MLP), but only 50% accuracy for complex activities.

2.2.4 Different Feature Extraction Methods

Preprocessed data are used for feature extraction. Feature exction methods are broadly grouped into three categories: (i) time-domain, (ii) frequency-domain, and (iii) others [16, 37]. The time-domain methods include (a) statistical feature extraction [31,32,38], (b) signal-amplitude normalization [21, 28], and (c) structure detectors [16]. Fast Fourier Transform (FFT) and Wavelet transforms are examples of frequency domain feature extraction methods [22, 29, 39–41]. The other methods include Principal Component Analysis (PCA) [30], Auto-regressive Method (AR) [40], K-Mean Clustering [42], and Linear Discriminant Analysis (LDA) [29, 30].

2.2.5 Different Learning Methods

A large number of training or learning methods have been used to train human-activity monitoring and recognition systems [16]. They include decision tree [22, 26, 39, 43], Bayesian [22,26,36,39,43], k-nearest neighbor [44], neural networks [26], support vector machines [24, 34, 38, 45, 46], fuzzy-logic [28], regression methods [2, 43], softmax regression [42], Hidden Markov Models [29, 47], and combinations of two or more of these methods [42, 48].

A standard method for testing performance of a trained model is k-fold cross validation [49]. In this method available examples are (randomized and) partitioned into approximately k equal-size subsets. Then $(k-1)$ of these k subsets are used for training and the

remaining subset is used for testing. The process is repeated k times, once for each subset and the results are averaged for estimating performance of the trained system.

2.3 Spasm Identification using EMG Recordings

Spasm is inventory muscle contraction happen in the paralyzed muscles of spinal cord injured person [6–9]. There are three main types of spasms [10]:(i) Unit, (ii) Tonic, and (iii) Clonus. Spasms are common in SCI patients and it can occur throughout the day and night. It is often manageable by the individual, but sometimes it can interfere with regular activities [6, 7]. Researchers use EMG recordings to identify spasms, its frequency of occurrence, its severity, and spasticity symptoms.

Among three different types of spasms, clonus spasm is a very well known type of involuntary muscle contraction for SCI patients. Over the last few decades, a number of studies have been focused on clonus muscle spasm that measures clonus location, overall clonus duration, contraction frequency, and contraction duration [10–12]. In the previous studies to detect and analyze clonus, EMG processing has been done over one type of muscles at a time [10, 11, 14]. Different EMG processing methods have been applied for clonus spasm detection with different varying accuracies [10–15].

EMG signal analysis is a very well know way to analyze spasms. Researchers use different techniques to process EMG recordings to identify different kind of spasms [8,50,51]. Rectification and integration of signals or root mean square values to extract information is a very popular method in EMG analysis [10, 11]. This time domain data analysis process is associated with the amplitude of the signals and one of the very conventional way of EMG signal data analysis. To analyze EMG recording in frequency domain Fourier transformation is the most well known way. Fourier transformation convert the signal from time

domain to frequency domain. On the other hand, short-term Fourier transforms (STFTs) and wavelets process the data in both time and frequency domain. The morlet wavelet is very popular for EMG data analysis. It is scaled linearly when the time resolution of events is unknown. But researchers use the non-linear scaling of wavelets when they want to locate the timing of events in the EMG signals [11]. Vincent Von Tscharner [52] developed the technique of scaling wavelets non-linearly and it is widely utilized in the current algorithm developed for analysis of EMG during clonus spasm.

Chaithanya et al. [11] developed algorithm to automatically detect contractions during clonus in long-term EMG records for per muscles channel. They have used the non-linearly scaling Morlet wavelet filter to envelope the EMG (74.8 - 193.9 Hz). They have identified contractions during clonus, marking start and end times, contraction intensity using root mean square EMG, and EMG duration.

Polynomial regression has been used by Takada and Yashiro to detect EMG pattern for spasm [13]. This method requires calculation of steady state parameters and equations for each EMG pattern. Vannozzi et al. [14] has develop a method that use wavelet transforms to detect the sudden changes in EMG that occur at the start and end of a contraction.

In [10], Jeffrey et al. has proposed and evaluated rule based algorithm and time-frequency methods that can classify different types of spasm events in long-term EMG recording. They have done their experiment with seven different subjects each with eight channels of EMG recordings. And their algorithms works one channel of EMG recording at a time.

2.4 Conclusion

In this chapter we have presented a brief literature review. We also have introduced basic concepts that are used in the rest of the proposed work. In Section 2.2, we have discussed about current state-of-the art research in human fall detection, and posture and activity identification and monitoring. After that in Section 2.3 we have introduced few necessary concepts to understand involuntary muscle contractions and spasms. Then we have presented a brief overview of current muscle spasm identification research. In the following chapter, we demonstrate that k-mean clustering algorithm can semi-automatically extract training examples from motion sensors data. And we also present one and two layers of classification networks for fall detection and posture monitoring from extracted features of the motion sensors data.

CHAPTER 3

Human Fall Detection And Posture Monitoring

In the previous chapter we provided a brief review of some fundamental concepts and current state-of-the-art research related to this dissertation. In this chapter we provide our methods, experiments and results of human fall and posture detection from motion sensor readings. This work uses k-mean clustering algorithm for feature extraction and current state of the art classification algorithms: softmax regression and neural networks for identification and monitoring of human fall events and activities.

3.1 Introductory Remarks

We have presented a brief review of human fall and posture detection in Section 2.2. Remaining major challenges include automatic development of (i) extracting features specific to applications and users, and (ii) software modules for making intelligent decisions utilizing features extracted from motion data. For example, if a device with a 9-axis MEMS motion sensor is used for identifying four fall events — fall forward, fall backward, fall left, and fall right — the device must have firmware modules to collect motion data, to preprocess motion data, and to extract sufficient features from the preprocessed data for the

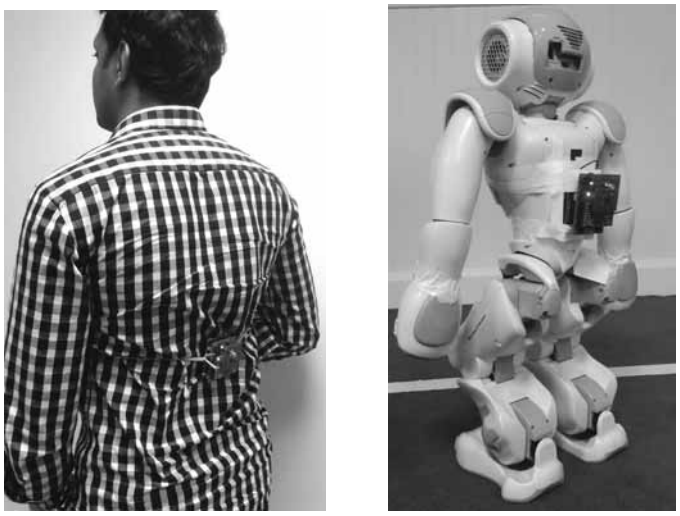
identification of the fall events. Also, the device must have a software module that takes the extracted features as input and correctly classifies the four fall events.

Next, we discuss the approaches and tools we have used to solve this fall detection and posture monitoring process. We also demonstrate that the clustering algorithm can semi-automatically extract training examples from motion data. Moreover, we present how to train and evaluate several one- and two-level classification networks to monitor non-fall activities and to detect fall events. The proposed classification-networks are combinations of neural networks and softmax regression.

3.2 Framework and Motion Sensors

This project used the μEnergia (*pronounced as*: "micro–Energia") framework for data collection and online monitoring. The framework provides generic functionalities to develop applications or rational agents on embedded devices that sense and actuate using add-on boards. The framework includes: (i) tools to develop modules and representations that execute on the micro-controllers or offline, (ii) methods to access functionalities for physical robots, and (iii) a real-time visualization system. This framework is lightweight, flexible, and consumes minimum memory and computational resources. This framework is available from the following *web site: http://muenergia.saminda.org*.

We have assembled a wireless sensing-device with a Tiva C Series TM4C123G micro-controller board, and three booster packs — a Sensor Hub BoosterPack for sensing 9-axis motion, a CC2533 BoosterPack for wireless networking, and a Fuel-Tank BoosterPack for power. We also assembled a wireless data collection device with a Tiva C Series TM4C123G micro-controller board, and a CC2533 BoosterPack. These two devices were networked to create a wireless sensor network (WSN) for collecting motion data from hu-



(a) (b)

Figure 3.1: (a) a wireless sensor device (assembled from a Tiva C Series TM4C123G LaunchPad, a Sensor Hub BoosterPack, a CC2533 BoosterPack, and a Fuel Tank BoosterPack) attached to the back of a human subject; and (b) the same device configuration was used on the back of a NAO humanoid robot [2].

mans (see Figure 3.1). The data collection device is connected to a computer with a USB cable for logging the data. This micro-controller board is programed with μEnergia framework.

3.3 Data Collection from Subjects

As shown in Table 3.1, this study has considered four types of fall events: 1) fall forward (FF), 2) fall backward (FB), 3) fall left (FL), and fall right (FR). We have used the following seven activities:1) walk forward (WF), 2) walk backward (WB), 3) walk left (WL), 4) walk right (WR), 5) marching (MR), 6) rotate counter clockwise (RC), and 7) rotate clockwise (RA) as non-fall activities.

We have setup the framework with our micro-controllers and enabled the motion tracking device to sample at $20ms$, which is equivalent to a $50Hz$ sampling rate. The motion

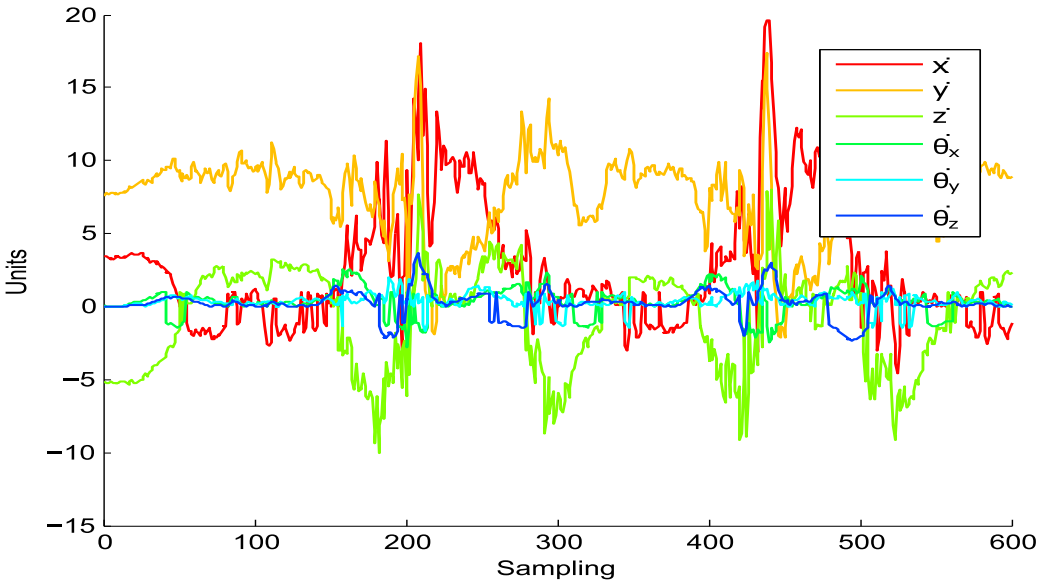
Table 3.1: Activities and Fall Events and their abbreviations

Activity	Abbreviation	Activity	Abbreviation
Fall Forward	**FF**	Fall Backward	**FB**
Fall Left	**FL**	Fall Right	**FR**
Walk Forward	**WF**	Walk Backward	**WB**
Walk Left	**WL**	Walk Right	**WR**
March	**MR**	Rotate Clockwise	**RC**
		Rotate Anticlockwise	**RA**

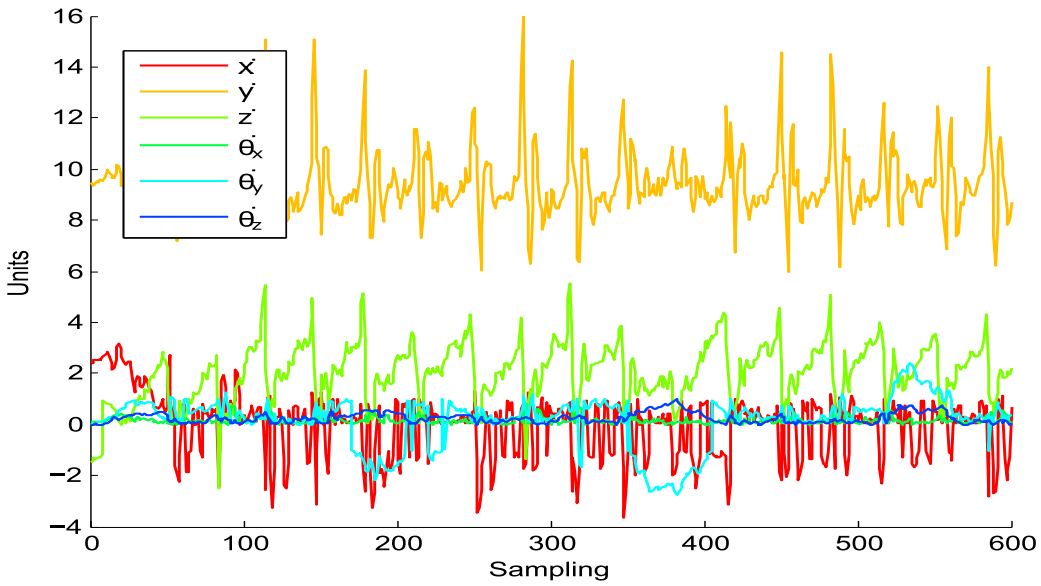
tracking device outputs nine values. Three-axis accelerometer readings are in meter per square second (m/s^2), 3-axis gyroscope readings are in radian per square second ($rads/s^2$), and 3-axis magnetometer readings are in Tesla (T). For experiments and results reported here, we have excluded the magnetometer readings. Therefore, our input data vectors are in $\mathbb{R}^6$ and they consist of accelerometer and gyroscope values. The plots of the raw motion datasets for fall forward and walk forward events are shown in Figures 3.2a and 3.2b, respectively. They demonstrate the complex nature of these time series, and indicate the difficulty of extracting features from them that are useful for training as well as monitoring of events and falls.

3.3.1 Identify and Adjust for Missing Data Points

Some data packets may be lost during transmission because of noise. We numbered each sensor reading sequentially and at the receiver the lost data points were identified from this number and replaced using linear interpolation. Since only very few data points were



(a) Motion dataset from fall forward of a human subject.



(b) Motion dataset from walk forward of a human subject.

Figure 3.2: Fall and walk forward motion datasets. The accelerometer ($\dot{x}$, $\dot{y}$, $\dot{z}$) and the gyroscope ($\dot{\theta_x}$, $\dot{\theta_y}$, $\dot{\theta_z}$) readings are shown with the respective traces.

lost and results obtained after linear interpolation was excellent, we did not experiment with other interpolation methods.

3.3.2 Noise Filtering

After checking for missing data and adding interpolated values for the lost data, high frequency noise was removed using a moving window averaging technique. We used a window of 20 consecutive samples, which amounts to $400ms$, and obtain the average value. Then the window was moved by ten samples and average value was computed again. The process was repeated for each time-series of all motion datasets. Thus, we have allowed ten samples to overlap between windows. Selection of an overlap of ten samples is based on our empirical observation that a transition from non-fall event to a fall event takes about $500ms$. Thus, averaged values will preserve transitions from non-fall events to fall events. We have used the preprocessed time-series as the input to our automatic training example extraction method.

In the next section, we propose a method for the semi-automatic extraction of training examples from motion datasets. These examples are used to train a device/system that identifies desired events.

3.4 Semi-Automatic Extraction of Training Examples

The barrier for collecting motion data is very low, if any. However, detection and identification of non-fall activities and fall events from motion data have remained an active area of research, as can be seen from the large number of recent publications [4, 5]. One of the main unsettled issues is how to characterize features from motion data that can be extracted easily for training and monitoring. In this section, we describe a method that semi-automatically extracts feature vectors that characterize non-fall activities and fall events.

The flow chart of the method used for extracting training examples is shown in Fig. 3.3. All the collected datasets were annotated as described next. If a dataset was for simulated-fall events, the dataset was treated as a 6-dimension vectors for clustering them into two clusters. We explored K-Mean and Gaussian Mixture clustering techniques. For completeness a brief description of each is provided next.

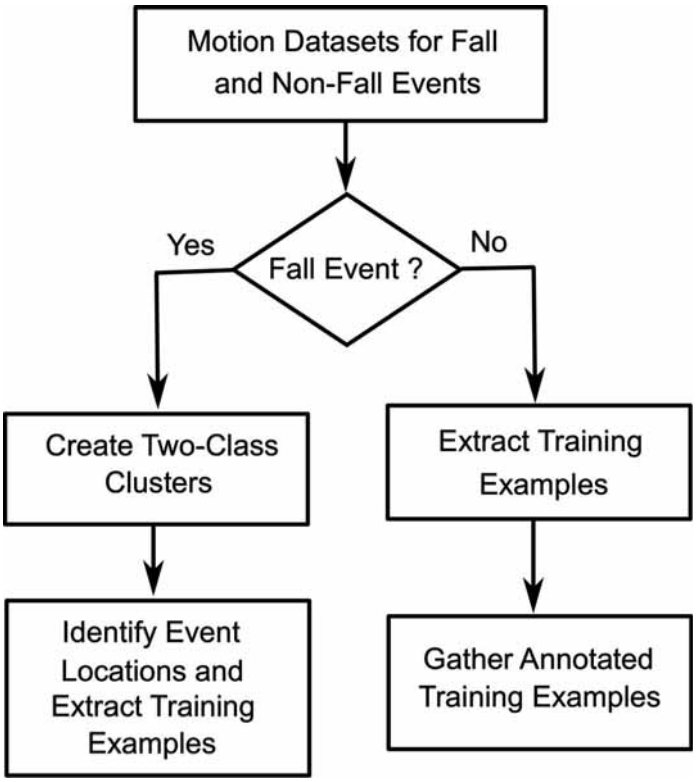


Figure 3.3: Flow chart of an algorithm for semi-automatic extraction of training examples

3.4.1 K-Mean Clustering

K-mean clustering [53] aims to partition a set of n observations into k clusters. Each observation is assigned to the cluster with the nearest mean. Let $\{(\mathbf{x}^{(i)})\}_{i=1}^{n}$ be the set of n observations, where each observation $\mathbf{x}^{(i)}$ is a d-dimensional real vector and is assigned to

one of the k sets $\mathrm{S}_j \in \mathcal{S}$, for $1 \leq j \leq k$ so that within-cluster sum of squares is minimized. Formally,

$$\min_{\mathcal{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathrm{S}_i} ||\mathbf{x} - \boldsymbol{\mu}_i||^2,$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

For our case, we set $k = 2$ for extracting training examples for fall events. Data samples grouped into two clusters and each cluster was annotated. Figures 3.4 shows annotated time-series for a fall-forward datasets. As can be seen for our fall-event datasets, the choice of $k = 2$ is the optimal value that clearly separated the duration of the fall.

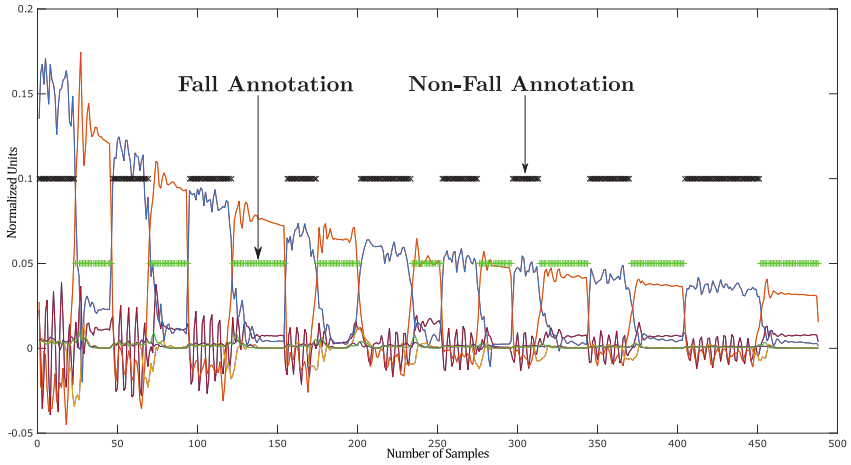


Figure 3.4: Semi-automatic annotation of falling-forward dataset.

3.4.2 Multivariate Gaussian Mixtures

Gaussian mixture clustering is a generalization of k-mean clustering, where each cluster is assumed to be from a Gaussian distribution parametrized by $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. The mean vector $\boldsymbol{\mu}_i$ represents the center of the cluster i. Usually, an *expectation maximization* (EM) algorithm iteratively identifies the clusters. Compared to the clustering results obtained

from k-mean, the mixture models did not provide an optimal separation for our fall datasets. Therefore, we omit results for the Gaussian mixture clustering, and present results for k-means algorithm only.

Our method is relatively simple and easy to implement in practice. We have used the samples from the preprocessing stage, and subjected them to k-means algorithm. We have empirically found that the minimum cost separation is achievable when we have two clusters. This intuitively supports the fall-event datasets as well. For example, if we consider the falling-forward event, there is a window in which the fall triggers and the subject falls to the ground. Therefore, the signature of this process differs from the states before the process starts (e.g., standing) and the states after (e.g., laying on the floor). Therefore, for each fall-event dataset, there exists a duration in which the fall triggered and continued.

Figures 3.4 shows plots and annotations of our semi-automatic annotation of falling-forward dataset, respectively. The subject in this experiment simulated a fall froward. After a fall, the subject immediately gets up and repeats the process. The "+" segments show the annotated fall events, while the "x" segments show the non-fall events. We have observed that there is a clear separation of the events with two clusters. Each segment has between 15 to 20 consecutive data points. Each fall-event triggers towards the end of the preceding non-fall data points and continues into the beginning of the fall data points; this is the transition zone of a fall event.

For visualization of 11 sets of training examples, a sample t-SNE (t-distributed stochastic neighbor embedding) [54] projection is displayed in Fig. 3.5. In this projection, the 4 fall events (numbered 1 through 4) show a clear separation from each other and all 7 non-fall events. Although Fig. 3.5 does show as good separations for non-fall events, the results,

reported in Section 3.7, show that the separations are for good enough for classification of the events with 100% accuracy.

Since we are interested in the signatures of the fall events, we have annotated all the samples that belong to "a" cluster as the fall data. In general, the k-means clustering finds a local minimum. Moreover, the cluster assignments may vary depending on the initialization. Therefore, after we have obtained the data belongs to two clusters, we only have to manually provide the semantics of the cluster centroid.

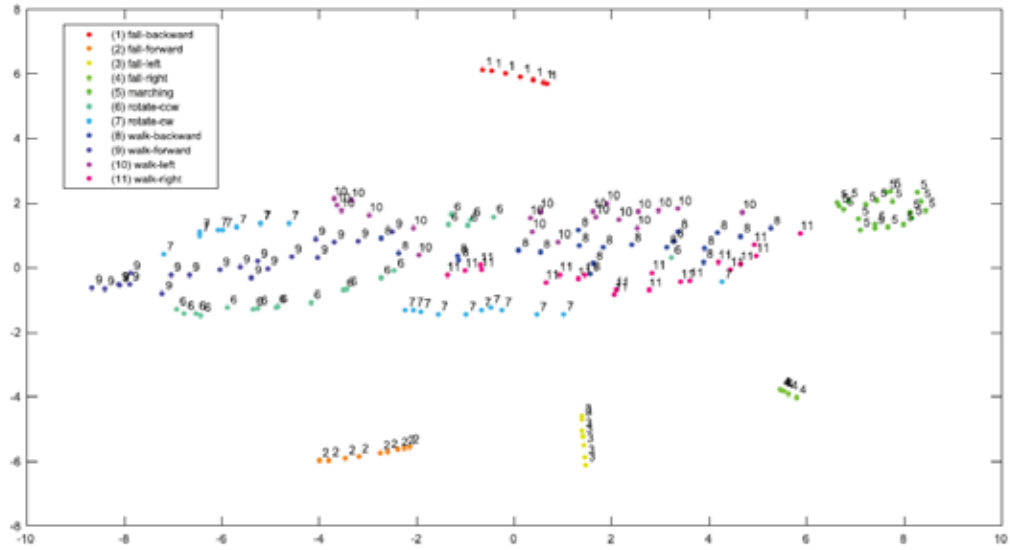


Figure 3.5: Visualization of 11 sets of training examples using t-SNE projection. In this projection, the 4 fall events (numbered 1 through 4) show a clear separation from each other and other 7 non-fall events.

3.5 Offline Learning and Predicting Algorithms

Our activity monitoring and identification requires to learn and to predict beliefs of multiple discrete hypothesis. This includes learning and predicting dichotomies such as falling forward and backward, falling events and non-falling events, different types of falling events, and so forth. Therefore, we have used multi-class classification networks

to rationally answer the question we have posed. We learn the classifiers offline from the training examples extracted according to the methods presented in Section 3.4. This section explains the classifiers that we have considered in our experiments and their configurations.

Using our semi-automated training example extraction method, we have created $i \in \mathbb{N}$ training examples $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{m}$ such that $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \{0,1\}^k$, where $k \in \mathbb{N}$ is the number of dichotomies. Each training example belongs only to one class and $\mathbf{y}$ uses the 1–of–k coding scheme.

3.5.1 Softmax Regression Algorithm

Next classification network is based on Softmax regression [53]. In this model, given an example $\mathbf{x}$, it will determine the probability, $\mathbb{P}(y|\mathbf{x})$, for each $y = 0, \ldots, k-1$. The input is appended with a constant bias term, $x_0 = 1$, therefore, $\mathbf{x} \in \mathbb{R}^7$. The model uses a parameter matrix $\mathbf{W} \in \mathbb{R}^{7\times k}$, and the output vector, $\mathbf{a} = \mathbf{W}\mathbf{x}$, is passed through the Softmax activation function $\mathbf{z}(\mathbf{a}) = \frac{e^{\mathbf{a}}}{\mathbf{1}^{\mathrm{T}} e^{\mathbf{a}}}$, which represents the probability mass function $\mathbb{P}(y|\mathbf{x})$. We have used the regularized cross-entropy cost function as our objective function,

$$l(\mathbf{x}) = -\mathbf{y}.\ln \mathbf{z} + \frac{\lambda}{2}\mathbf{W}.\mathbf{W} \tag{3.1}$$

where λ is the regularization parameter, and "." represents the scalar dot or Frobenius product. We have used L-BFGS [55] to learn the parameter vector $\mathbf{W}$.

We trained and tested two networks with the softmax regression algorithm. In the first case, we trained a one-layer network with 11 binary outputs, one for each fall or non-fall events. In the second case, we trained a two layer softmax regression network. The top layer network was trained with 2 binary outputs to distinguish between fall and non-fall events (see Fig. 3.6). The bottom layer has two sub-networks: one to identify fall events

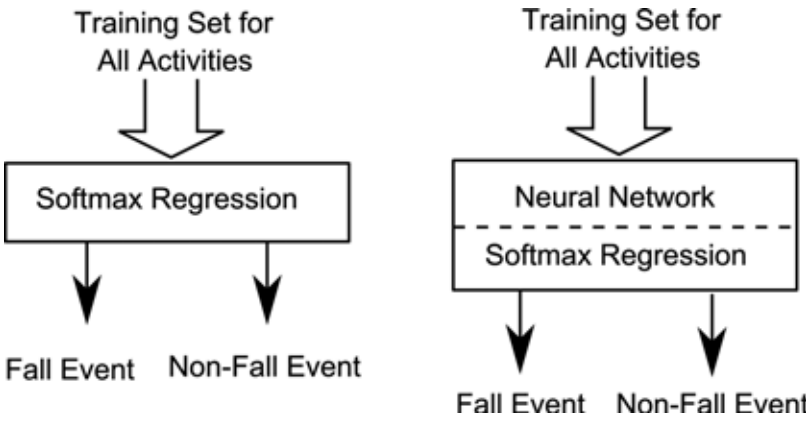


Figure 3.6: Layer 1 learning modules. Examples are classified into Fall and Non-Fall events. The left-side Figure is using Softmax Regression model. The Right-side Figure is a hybrid of Neural Network and Softmax Regression model.

and has 4 binary outputs, the other to identify non-fall events and has 7 binary outputs. Each sub-network was trained separately. Figure 3.7 shows a block diagram of a subnetwork for training and identification of four fall events. For conserving space, we omit the diagram for non-fall events. As will be discussed in the next section, none of them identified all events with 100% accuracy.

3.5.2 Hybrid Algorithm: Neural Network with Softmax Regression

The hybrid classification network is based on both Artificial Neural Network and softmax regression network. Fig. 3.6 shows a block diagram of a hybrid network: a row of neurons followed by a row of softmax regression units.

We have used the prior softmax activation and the negative-log likelihood functions in the output layer. The hidden layers have used *rectified linear units* (ReLUs). We have also included the bias terms for all hidden layers. The backpropagation algorithm [56] has been used to calculate the gradient, while the L-BFGS procedure has been used to learn the network parameters. We have independently set each dimension of the sample to

have zero-mean and unit-variance. We achieved this by first computing the mean of each dimension across the training set and subtracting this from each dimension. Then each dimension is divided by its standard deviation. Similar to softmax regression networks, we experimented with both single-layer hybrid networks and two-layer hybrid networks. While single-layer hybrid network failed to identify all events with 100% accuracy, the two-layer hybrid network successfully identified all event with 100% accuracy. There were no false positives or false negatives. In the rest of paper, we referred hybrid networks as neural networks.

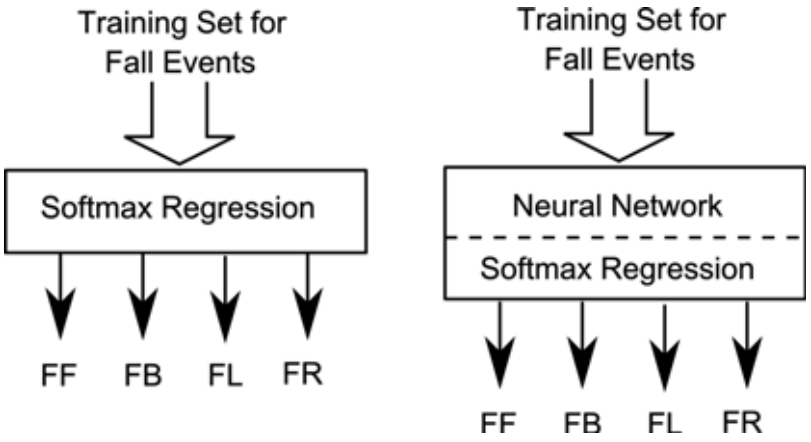


Figure 3.7: Layer 2 learning module for four Fall events. The left-side Figure is using Softmax Regression model. The right-side Figure is a hybrid of Neural Network and Softmax Regression

3.6 Online Activity Monitoring and Identification

In Sections 3.4 and 3.5, we have discussed the semi-automatic training example extraction and the learning of the parameter vectors of the classifiers for the event prediction and identification. This process is primarily offline. In operational mode, we require that the learned classifiers monitor and identify events online. This section provides our setup and methodology.

3.6.1 Monitoring of Events

For the monitoring of events, we have followed a methodology similar to that of Subsection 3.3. We collected the sample data to a circular buffer. From a starting marker (initially at the beginning of the circular buffer), we have averaged 20 consecutive sample and added that value to a second circular buffer. The first circular buffer is incremented by 10 samples, then average the next 20 samples (if the data is available), and pushed the average value into the second circular buffer. We have continued this process in the first circular buffer with 10 sample overlap. In the second circular buffer, we have maintained a window of 20 samples. In order to compensate for noise, we have used five average values from the window. These values are used in the event identification. We then incremented the second circular buffer by 10 samples and continued the process.

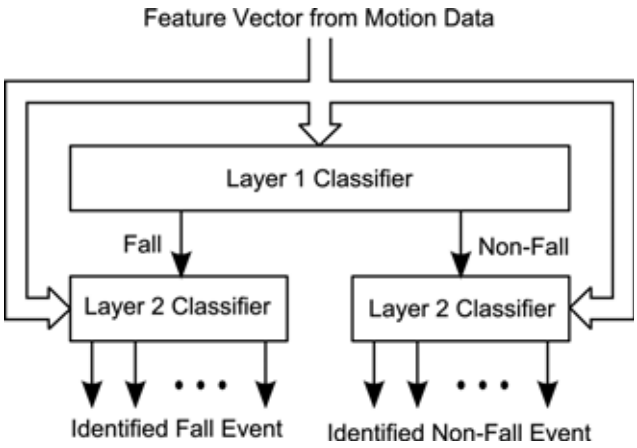


Figure 3.8: Block diagram of event identification system. The top layer separates fall events from non-fall events. The bottom layer modules receive the same input data vector as the top layer, but they also get a zero or one input from the upper layer module depending on where the input data vector is classified. The final output is a binary valued vector of 11 components.

3.6.2 Identification of Events

As stated earlier and also we will discuss in Section 3.7 for 100% success rate, we have used a layered classification network as shown in Fig. 3.8. The inputs to the classifiers are the pre-processed motion data from the previous subsection. In the first layer of our classification network (Layer 1), we have used a binary decision classifier that predicts whether the current input is a fall or a non-fall. Based on this decision, at the second layer of our network (Layer 2), one of the specific fall event or non-fall activity recognition classification network is activated. The decision of this classifier is considered as the output of the classification network. In order to compensate for noise, we have used the five samples as described in the previous section. We have marked the predicted events; if the five inputs (calculated from a window of the second circular buffer) predict the same outcome, then the network voted to that event, otherwise, we don't identify any event.

3.7 Empirical Evaluation of Proposed Algorithms and Methods

In this section we report the performance of our proposed algorithms and methods. A detailed description of the data collection and preprocessing steps can be found in Section 3.3. We have experimented with one-layer and two-layer classifiers. First, the performance of one-layer softmax and neural networks classifiers are reported. Then that for two-layer networks are presented. For all results reported here the value of λ in the cost function (Equation 3.1) is 0.1.

3.7.1 One-Layer Networks

We trained two one-layer networks: softmax regression and neural networks (see Fig. 3.6 for block diagrams of these networks). Performance of these networks are discussed in the next two paragraphs.

3.7.1.1 Softmax Regression

As can be seen from Table 3.2, one-layer softmax regression network with 7 inputs (6 for feature vector and one for bias) and 11 outputs recognized on an average 94.4% of all events. A closer examination reveals that all but three types of events — fall-forward, walk-forward, and walk-right events — are recognized correctly. Recognition rate for fall-forward events is only 66.7%. This event is sometimes incorrectly classified as a walk backward or a march event. Walk forward event is correctly recognized on an average rate of 87.5% and sometimes it is incorrectly classified as a walk-right event. Recognition rate for walk right events is same as walk forward and is confused with some walk-left events.

3.7.1.2 Neural Network

Performance of neural networks is shown in Table 3.3; it recognized 95.8% of all events correctly. Before we discuss our results, it should be noted that the network is composed of one hidden row of 16 neurons with ReLU followed by a row of 11 elements with softmax activation function (see Fig. 3.6 in Section 3.5). An advantage of this architecture is that the number of neurons in the hidden row can be varied for optimal performance.

As can be seen from Table 3.3, all events except fall-forward and walk-right events were recognized correctly. The neural network improved recognition rate of walk-forward

Table 3.2: Performance of a single layer softmax regression network with six inputs and 11 outputs. On an average 94.4% of the events were classified correctly.

	FF	**FB**	**FL**	**FR**	**WF**	**WB**	**WL**	**WR**	**MR**	**RC**	**RA**
FF	66.7	0	0	0	0	16.7	0	0	16.7	0	0
FB	0	100	0	0	0	0	0	0	0	0	0
FL	0	0	100	0	0	0	0	0	0	0	0
FR	0	0	0	100	0	0	0	0	0	0	0
WF	0	0	0	0	87.5	0	0	12.5	0	0	0
WB	0	0	0	0	0	100	0	0	0	0	0
WL	0	0	0	0	0	0	100	0	0	0	0
WR	0	0	0	0	0	0	12.5	87.5	0	0	0
MR	0	0	0	0	0	0	0	0	100	0	0
RC	0	0	0	0	0	0	0	0	0	100	0
RA	0	0	0	0	0	0	0	0	0	0	100

events to 100%. But recognition rates of fall-forward and walk-right events remained same as softmax network at 66.7% and 87.5%, respectively.

Because of failure of one-layer networks, we explored two-layer classification networks: the first layer classified an input into two categories, a fall event or a non-fall event. Then, a second layer differentiated events within its own category. A block diagram of the classifier is shown in Fig. 3.8.

Table 3.3: Performance of a Neural network with one row of neurons with ReLU and one row of elements with softmax activation function. On an average 95.8% of the events were classified correctly.

	FF	**FB**	**FL**	**FR**	**WF**	**WB**	**WL**	**WR**	**MR**	**RC**	**RA**
FF	66.7	0	0	0	0	0	16.7	0	16.7	0	0
FB	0	100	0	0	0	0	0	0	0	0	0
FL	0	0	100	0	0	0	0	0	0	0	0
FR	0	0	0	100	0	0	0	0	0	0	0
WF	0	0	0	0	100	0	0	0	0	0	0
WB	0	0	0	0	0	100	0	0	0	0	0
WL	0	0	0	0	0	0	100	0	0	0	0
WR	0	0	0	0	0	0	12.5	87.5	0	0	0
MR	0	0	0	0	0	0	0	0	100	0	0
RC	0	0	0	0	0	0	0	0	0	100	0
RA	0	0	0	0	0	0	0	0	0	0	100

3.7.2 Two-Layer Networks

3.7.2.1 Layer One - Classification of Fall Events or Non-fall Activities

Performance of layer-one of a two-layer classifiers has been tabulated in Table 3.4. As can be seen from the table, softmax network at layer-one (top layer) correctly classified only 87.2% of fall and non-fall events, but neural network correctly classified all fall and non-fall events. We used only two neurons in the hidden row.

Table 3.4: Performance of layer-one of a two-layer classifiers. Softmax network at layer-one correctly identified 87.17% of fall events from non-fall events, but the neural network correctly identified all fall and non-fall events.

	Softmax L1: 87.2		**Neural L1: 100**	
	Fall Event	**Non-Fall Event**	**Fall Event**	**Non-Fall Event**
Fall Event	85.7	14.3	100	0
Non-Fall Event	12.7	87.3	0	100

Because of its 100% recognition rate we choose neural networks as the layer-one network in our two-layer classification networks. In this layer-one neural network it has two hidden units.

3.7.2.2 Layer Two - Identification of Individual Fall Events or Non-fall Activities

Performances of layer-two networks are shown in Tables 3.5, 3.6, and 3.7. It is easy to see from Table 3.5 that both softmax and neural networks at layer-two correctly classified 100% of fall events. Results for seven non-fall events are shown in Tables 3.6 and 3.7. Again both softmax and neural networks achieved 100% accuracy. The neural network used one hidden row of 6 neurons.

3.8 Conclusions

During activities of daily life, an accident such as a fall may occur to humans. This may leads to injuries, and immediate identification of it can alert for a fast response. There are many inexpensive wireless motion sensing devices or one can be assembled using off-the-shelf components. These devices can be attached to a human to collect motion data

	Softmax: L2 Fall Event: 100				Neural L2 Fall Event: 100			
	FF	**FB**	**FL**	**FR**	**FF**	**FB**	**FL**	**FR**
FF	100	0	0	0	100	0	0	0
FB	0	100	0	0	0	100	0	0
FL	0	0	100	0	0	0	100	0
FR	0	0	0	100	0	0	0	100

Table 3.5: Performance of layer-2 network for four fall events. Both softmax and neural networks correctly classified 100% of fall events.

that can be used for monitoring activities of daily living. However, while performing these activities, an accidental fall may occur.

In order to accurately identify these events, it is necessary to extract distinguishing features from the motion data. Here, we have proposed a novel semi-automatic training example extraction method to identify features and automatically annotate the datasets. The proposed method significantly expedites the creation of the training set compared to manually extracting them. Moreover, we have proposed and implemented a two-level classification network with a combination of neural and softmax regression networks to identify seven type of non-fall activities and four types of fall events with very high accuracy.

While the evaluation of the proposed techniques using our off-the-shelf device has proven to be effective, the project is far from complete. In the next chapter we propose a distance measure. We also propose and evaluate three different algorithms to detect and monitor ADLs where we are able avoid miss-classification using a thresholding technique.

Table 3.6: Performance of layer-2 network for seven non-fall events. Softmax regression correctly classified 100% of non-fall events.

	WF	**WB**	**WL**	**WR**	**MR**	**RC**	**RA**
WF	100	0	0	0	0	0	0
WB	0	100	0	0	0	0	0
WL	0	0	100	0	0	0	0
WR	0	0	0	100	0	0	0
MR	0	0	0	0	100	0	0
RC	0	0	0	0	0	100	0
RA	0	0	0	0	0	0	100

Table 3.7: Performance of layer-2 network for seven non-fall events. Neural networks correctly classified 100% of seven non-fall events.

	WF	**WB**	**WL**	**WR**	**MR**	**RC**	**RA**
WF	100	0	0	0	0	0	0
WB	0	100	0	0	0	0	0
WL	0	0	100	0	0	0	0
WR	0	0	0	100	0	0	0
MR	0	0	0	0	100	0	0
RC	0	0	0	0	0	100	0
RA	0	0	0	0	0	0	100

CHAPTER 4

Log-Sum Distance Algorithm for ADLs Monitoring and Recognition

4.1 Introductory Remarks

In the previous chapter we presented fall and activity recognition algorithm using semi-automatically extracted features from motion data. There we used existing classification algorithm neural network and softmax regression. In this chapter we propose a distance measure, called Log-Sum Distance, for evaluating the distance between two sequences of positive numbers[1]. We use the Log-Sum Distance measure to develop algorithms for recognition of human activities (such as walking forward, going up and down stairs etc.) from motion data. Here, we propose, develop, and evaluate three models for recognition of human-activities from wearable motion-sensor data. For recognition of activities, (i) one model uses parameters from Multiple Linear Regression Model (MLRM), and (ii) the other two models use residual sum of squares (RSS) values from a MLRM. We also incorporate a automatic threshold parameter selection process to avoid misclassification.

[1]Prof. Kamal Premaratene brought to our attention that the proposed measure can be derived from Bregman divergence [1]

4.2 Linear Regression Model

Let $X = [X_1, X_2, ...X_m]$ be a matrix of n rows and m columns, where each $X_j = [x_{1j}, x_{2j}, \cdots, x_{nj}]^T$, for $1 \leq j \leq m$, is a column of n elements. Let $y = [y_1, y_2, ..., y_n]^T$ be a column vector of n elements.

Let a *linear model* $f(\cdot)$ relate element y_i of y with ith row, $[x_{i1}, x_{i2}, \cdots, x_{im}]$, of X. Mathematically,

$$y_i = f(x_{i1}, x_{i2}, \cdots, x_{im}) = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_m x_{im} \tag{4.1}$$

If we denote $\beta = [\beta_0, \beta_1, ...\beta_m]^T$ as a column vector, Equation 4.1 can be rewritten as,

$$y_i = [1, x_{i1}, x_{i2}, \cdots, x_{im}]\beta.$$

Inserting a '1' before the first element of each row of X, we get a matrix of n rows and $(m+1)$ columns, $\boldsymbol{X} = [\mathbf{1}, X]$. Now using matrix notation we can write,

$$y = f(\boldsymbol{X}) = \boldsymbol{X}\beta \tag{4.2}$$

Suppose y and X are obtained from observing a system, and we want to estimate a model $\hat{f}(\cdot)$ from the data. This is equivalent to estimating values of β from y and $\boldsymbol{X}$. Let $\hat{\beta}$ denote an estimate of β, that is, $\hat{\beta} = [\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_m]$. Now the approximation of y from $\boldsymbol{X}$ using $\hat{f}(\cdot)$ (that is, $\hat{\beta}$) can be written as,

$$\hat{y} = \hat{f}(\boldsymbol{X}) = \boldsymbol{X}\hat{\beta}, \tag{4.3}$$

where

$$\hat{y}_i = [1, x_{i1}, x_{i2}, \cdots, x_{im}]\hat{\beta}. \tag{4.4}$$

The selection of the parameters $\hat{\beta}$ for approximation of the function $\hat{f}(\cdot)$ depends on the minimization of errors,

$$e_i(\hat{\beta}) = y_i - \hat{y}_i. \tag{4.5}$$

By far the most popular method for selection of $\hat{\beta}$ is that minimizes the *residual sum of squares* [57].

$$RSS(\hat{\beta}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = ||y - \hat{y}|| \tag{4.6}$$

Denoting mean of $RSS(\hat{\beta})$ as $\bar{e}(\hat{\beta})$, we can write.

$$\bar{e}(\hat{\beta}) = RSS(\hat{\beta})/n = ||y - \hat{y}||/n. \tag{4.7}$$

Being a quadratic function of the parameters $\hat{\beta}$, its minimum always exists. The minimum is unique if $\boldsymbol{X^T X}$ is nonsingular, and the corresponding approximation of β is given by

$$\hat{\beta} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X}^T y. \tag{4.8}$$

Above result can be found in any standard book on matrix computation, but for completeness, next we state it as a theorem.

Theorem 1. (Least square error model [58]) *The linear model in Equation 4.2 that minimizes sum of squared error (given by Equation 4.6) always has a solution. The solution is unique if and only if* $\boldsymbol{X}^T \boldsymbol{X}$ *is nonsingular, and the solution is given by Equation 4.8.*

□

From this estimated model $\hat{f}(\cdot)$ we can get an estimate of y_k from a new observation x_k. Expressing in vector-matrix notation,

$$\hat{y} = \boldsymbol{X}\hat{\beta}. \tag{4.9}$$

It is also important to recall that vector $\hat{\beta}$ points in the steepest uphill direction in the input subspace [57]. Therefore, two characterizing features of the data are $\hat{\beta}$ and $RSS(\hat{\beta})$. In this research we use both of these features.

Before we discuss their applications for identification of activities of daily living from motion datasets, we briefly introduce entropy, KL-distance, and establish some of its relation to log-sum measures of two sequences of positive numbers.

4.3 Entropy and KL-divergence

Let $p = \{p_1, p_2, \cdots, p_m\}$ and $q = \{q_1, q_2, \cdots, q_m\}$ be two probability mass distributions [59]. Note that p and q satisfy all axioms of probability mass distributions, including $\sum_{i=1}^{m} p_i = \sum_{i=1}^{m} q_i = 1$.

Definition 1. *(*Entropy*) The* entropy $H(p)$ *of a probability mass distribution* p *is defined by*

$$H(p) = -\sum_{i=1}^{m} p_i \log p_i \tag{4.10}$$

□

Definition 2. *(*KL-distance *[60]) The* Kullback-Leibler distance *or* KL-distance *or* relative entropy *between two probability mass distributions* p *and* q *is defined as,*

$$D(p||q) = \sum_{i=1}^{m} p_i \log \frac{p_i}{q_i} \tag{4.11}$$

□

Theorem 2. *(*KL-distance inequality [60]*) For two probability mass distributions p and q*

$$D(p||q) \geq 0. \tag{4.12}$$

Equality holds if and only if $p_i = q_i$ for all $1 \leq i \leq m$. □

While the above theorem is used in many applications to measure entropy distance between probability mass distributions p and q, it should be noted that the measure is not symmetric, that is, $D(p||q) \neq D(q||p)$ unless p and q are identical. For application that benefits from a symmetric measure, sum of $D(p||q)$ and $D(q||p)$ is used, and the corresponding inequality is

$$D(p||q) + D(q||p) \geq 0 \tag{4.13}$$

We experimented with both the original KL-distance and the symmetric KL-Distance measures. We found that identification of ADLs with original KL-distance measures were much worse than with symmetric KL-distance measures. Thus, to conserve space performance of the original KL-distance measures is not reported here.

Next we introduce a symmetric log-sum distance measure for two sequences of positive numbers of identical length.

4.4 Log-Sum Distance

Definition 3. *(*Log-Sums distance*) For two sequences of positive numbers $U = < u_1, u_2, \cdots, u_m >$ and $V = < v_1, v_2, \cdots, v_m >$ log-sum distance of U and V is defined as,*

$$LD(U||V) = \sum_{i=1}^{m} u_i \log \frac{u_i}{v_i} + \sum_{i=1}^{m} v_i \log \frac{v_i}{u_i} \tag{4.14}$$

Next we show that $LD(U||V)$ is non-negative.

Theorem 3. *(*Log-sums inequality*) For two sequences of positive numbers* $U = < u_1, u_2, \cdots, u_m >$ *and* $V = < v_1, v_2, \cdots, v_m >$,

$$LD(U||V) \geq 0 \tag{4.15}$$

Equality holds if and only if $u_i = v_i$ *for all* $1 \leq i \leq m$.

Proof. Assume without loss of generality that $u_i > 0$ and $v_i > 0$. We start with the definition of $LD(U||V)$ in Equation 4.14,

$$\begin{aligned} LD(U||V) &= \sum_{i=1}^{m} u_i \log \frac{u_i}{v_i} + \sum_{i=1}^{m} v_i \log \frac{v_i}{u_i} \\ &= \sum_{i=1}^{m} (u_i - v_i) \log \frac{u_i}{v_i} \end{aligned} \tag{4.16}$$

Now, we show that value of each term of the sum is (i) zero when $u_i = v_i$ and (ii) greater than zero when $u_i \neq v_i$. Thus, the sum cannot be negative and, moreover, it is greater than zero if there exist at least one pair of u_i and v_i such that $u_i \neq v_i$.

For each term of the sum in Equation 4.16, we have to consider two cases: $u_i = v_i$, and $u_i \neq v_i$.

Case 1 ($u_i = v_i$): In this case, since $(u_i - v_i) = 0$ and $\log(u_i/u_i) = 0$, we have $(u_i - v_i) \log \frac{u_i}{v_i} = 0$.

Case 2 ($u_i \neq v_i$): In this case we have to consider two situations, $u_i < v_i$, and $u_i > v_i$. If $u_i < v_i$, both $(u_i - v_i)$ and $\log(u_i/v_i)$ are negative numbers, and hence, their product is greater than zero. On the other hand, if $u_i > v_i$, both $(u_i - v_i)$ and $\log(u_i/v_i)$ are greater than zero, and hence, their product is also greater than zero.

Thus, the sum in the right-side of Equation 4.16 is zero, if $u_i = v_i$, for all $1 \leq i \leq m$. On the other hand, if for one or more terms of the right-hand side of Equation 4.16 is greater than zero, the right-hand side of the equation is greater than zero. This completes the "if" part of the proof.

For the only if part of the proof it is easy to show that if the sum is zero, then each individual term must be zero. Because if that is not the case then one or more terms of the sum must be negative; in that case $(u_i - v_i)$ and $\log(u_i/v_i)$ must have opposite signs, which is impossible. Similar arguments hold for the case when the sum is greater than zero. □

Next we establish a relation between $LD(U||V)$ and KL-distance.

Theorem 4. *(*Log-Sums distance and KL-distance relation*) For two sequences of positive numbers U and V,*

$$LD(U||V) = (u - v)\log\frac{u}{v} + uD(p||q) + vD(q||p) \tag{4.17}$$

where $u = \sum_{i=1}^{m} u_i$, $v = \sum_{i=1}^{m} v_i$, $p_i = u_i/u$, and $q_i = v_i/v$ for all $1 \leq i \leq m$.

Note that $p = \{p_1, p_2, \cdots, p_m\}$ and $\sum_{i=1}^{m} p_i = 1$. Also, $q = \{q_1, q_2, \cdots, q_m\}$ and $\sum_{i=1}^{m} q_i = 1$. While p and q are not probability mass distributions, they can be used for measuring $D(p||q)$ and $D(q||p)$ because sum of the terms in p (and q) is 1.

Proof. Since $u_i = up_i$, we have

$$\begin{aligned}\sum_{i=1}^{m} u_i \log\frac{u_i}{v_i} &= \sum_{i=1}^{m} up_i \log\frac{up_i}{vq_i} \\ &= u\sum_{i=1}^{m} p_i \log\frac{u}{v} + u\sum_{i=1}^{m} p_i \log\frac{p_i}{q_i} \\ &= u\log\frac{u}{v} + uD(p||q)\end{aligned} \tag{4.18}$$

Similarly, it can be shown that,

$$\sum_{i=1}^{m} v_i \log\frac{v_i}{u_i} = v\log\frac{v}{u} + vD(q||p) \tag{4.19}$$

For completing the proof, first we use the definition of $LD(U||V)$ and then Equations 4.18 and 4.19.

$$\begin{aligned} LD(U||V) &= \sum_{i=1}^{m} u_i \log \frac{u_i}{v_i} + \sum_{i=1}^{m} v_i \log \frac{v_i}{u_i} \\ &= u \log \frac{u}{v} + uD(p||q) + v \log \frac{v}{u} + vD(q||p) \\ &= (u-v) \log \frac{u}{v} + uD(p||q) + vD(q||p) \end{aligned}$$

This completes the proof. □

Corollary 1. *(*Log-sum and KL-distance inequality*) For U, V, u, v, p, and q as defined earlier,*

$$LD(U||V) \geq uD(p||q) + vD(q||p). \tag{4.20}$$

Equality holds if and only if $u = v$.

Proof. Since for any two positive numbers u and v, $(u - v)\log(u/v) \geq 0$ (see proof of Theorem 3), the proof of the inequality follows immediately from Theorem 4. □

In the next section we present methods that utilize the results in the Section 4.2 , 4.3, and 4.4 for identification of human activities.

4.4.1 Log-Sum Distance and Bregman divergence [61]

Let $F : \mathcal{X} \to \mathbb{R}^+$ be a convex function, where $\mathcal{X} \subseteq \mathbb{R}^m$. Then Bregman divergence D_F between two points $U, V \in \mathcal{X}$ is defined by

$$D_F(U||V) = F(U) - F(V) - \langle U - V, \nabla F(V) \rangle. \tag{4.21}$$

Let, $f(x_i) = x_i \log x_i$; where $x \in \mathbb{R}^m$ and dom $f = (0, \infty)$

Again, if we define F as a sum of function $f(.)$, then we get $F(x) = \sum_{i=1}^{m} f_i(x_i)$ and the gradient of F at vector point x

$$\begin{aligned}
\nabla F(x) &= \left[\frac{\partial F(x)}{\partial x_1}, \frac{\partial F(x)}{\partial x_2}, \ldots, \frac{\partial F(x)}{\partial x_m}\right] \\
&= \left[\frac{\partial f_i(x_i)}{\partial x_1}, \frac{\partial f_i(x_i)}{\partial x_2}, \ldots, \frac{\partial f_i(x_i)}{\partial x_m}\right] \\
&= \left[\frac{\partial \sum_{i=1}^{m} x_i \log x_i}{\partial x_1}, \frac{\partial \sum_{i=1}^{m} x_i \log x_i}{\partial x_2}, \ldots, \frac{\partial \sum_{i=1}^{m} x_i \log x_i}{\partial x_m}\right] \\
&= \left[1 + \log x_1, 1 + \log x_2, \ldots, 1 + \log x_m\right] \qquad (4.22)
\end{aligned}$$

Let $U =< u_1, u_2, \ldots, u_m >$ and $V =< v_1, v_2, \ldots, v_m >$ are two vectors consist of positive numbers, then Bregman divergence from equation 4.21 can be re-written as

$$\begin{aligned}
D_F(U||V) &= \sum_{i=1}^{m} f(u_i) - \sum_{i=1}^{m} f(v_i) - \sum_{i=1}^{m} (u_i - v_i)(1 + \log v_i) \\
&= \sum_{i=1}^{m} u_i \log u_i - \sum_{i=1}^{m} v_i \log v_i - \sum_{i=1}^{m} u_i \log v_i - \sum_{i=1}^{m} u_i + \sum_{i=1}^{m} v_i \log v_i + \sum_{i=1}^{m} v_i \\
&= \sum_{i=1}^{m} u_i \log u_i - \sum_{i=1}^{m} u_i \log v_i - \sum_{i=1}^{m} u_i + \sum_{i=1}^{m} v_i \\
&= \sum_{i=1}^{m} u_i \log \frac{u_i}{v_i} - \sum_{i=1}^{m} u_i + \sum_{i=1}^{m} v_i \qquad (4.23)
\end{aligned}$$

where $\langle U - V, \nabla F(V) \rangle = \sum_{i=1}^{m} (u_i - v_i)(1 + \log v_i)$

Similarly, we can show that

$$D_F(V||U) = \sum_{i=1}^{m} v_i \log \frac{v_i}{u_i} - \sum_{i=1}^{m} v_i + \sum_{i=1}^{m} u_i \qquad (4.24)$$

Now adding equation 4.23 and 4.24 we get symmetric Bregman divergence

$$\begin{aligned}
D_F(U||V) + D_F(V||U) &= \sum_{i=1}^{m} u_i \log \frac{u_i}{v_i} - \sum_{i=1}^{m} u_i + \sum_{i=1}^{m} v_i + \sum_{i=1}^{m} v_i \log \frac{v_i}{u_i} - \sum_{i=1}^{m} v_i + \sum_{i=1}^{m} u_i \\
&= \sum_{i=1}^{m} u_i \log \frac{u_i}{v_i} + \sum_{i=1}^{m} v_i \log \frac{v_i}{u_i} \\
&= LD(U||V) \qquad (4.25)
\end{aligned}$$

Hence, we see that the symmetric Bregman divergence is Log-Sum Distance measure when the convex function F is defined as $F(x) = \sum_{i=1}^{m} f_i(x_i) = \sum_{i=1}^{m} x_i \log x_i$.

4.5 Activity Detection Algorithms

Let $A = \{1, 2, ..., n_a\}$ be a set of ADLs, and let the set $\{1, 2, \cdots, m\}$ be denoted by M. During each activity $k \in A$, *Wireless Sensing Units* (WSUs) with *Inertial Measurement Units* (IMUs) are attached to different locations on the body of a subject. These WSUs collect motion data and send the data to a computer for preprocessing. Let us assume that each WSU collects m time series of length n.

Initially datasets are collected for all n_a activities. These datasets are processed by feature extraction algorithms, which are used for training a device for identification of each activity. For correct identification of each activity, the set of features for the activity must have an unique signature; otherwise, correct identification of the activity may not be possible. During the monitoring phase datasets are collected continuously, their features are extracted, and these features are compared with learned features to identify activity being performed.

4.5.1 Data Preparation for Extraction of Features

For $k \in A$, let $\boldsymbol{X}^{(k)} = [X_1^{(k)}, X_2^{(k)}, \cdots, X_m^{(k)}]$ is the m-time series of an activity k. When $X_l^{(k)}$, for $l \in M$, is removed from $\boldsymbol{X}^{(k)}$ and a unit column vector $\mathbf{1}$ of appropriate size is appended as the first column, let the new matrix be denoted as $\boldsymbol{X}_{l'}^{(k)}$. That is,

$$\boldsymbol{X}_{l'}^{(k)} = [\mathbf{1}, X_1^{(k)}, \cdots, X_{l-1}^{(k)}, X_{l+1}^{(k)}, \cdots, X_m^{(k)}]. \tag{4.26}$$

In this study, we use the method described in Section 4.2 for estimating m linear models $\hat{f}_l^{(k)}(.)$, for each $k \in A$ and each $l \in M$, to relate $X_l^{(k)}$ with $\boldsymbol{X}_{l'}^{(k)}$. Thus,

$$\hat{X}_l^{(k)} = \hat{f}_l^{(k)}(\boldsymbol{X}_{l'}^{(k)}) = \boldsymbol{X}_{\boldsymbol{l'}}^{(\boldsymbol{k})} \hat{\beta}_l^{(k)}, \tag{4.27}$$

where

$$\hat{\beta}_l^{(k)} = ((\boldsymbol{X}_{l'}^{(k)})^T \boldsymbol{X}_{l'}^{(k)})^{-1} (\boldsymbol{X}_{l'}^{(k)})^T X_l^{(k)}. \tag{4.28}$$

Recall that $\hat{\beta}_l^{(k)}$ points in the steepest uphill direction and minimizes residual error $RSS(\hat{\beta}_l^{(k)})$. For each activity $k \in A$ we have m steepest uphill direction vectors $\hat{\beta}_l^{(k)} = [\hat{\beta}_{l0}^{(k)}, \hat{\beta}_{l1}^{(k)}, \cdots, \hat{\beta}_{lm}^{k}]^T$, for $l \in M$. Also, corresponding to each $\hat{\beta}_l^{(k)}$, we have a mean residual sum of squared error $\bar{e}_l^{(k)}(\hat{\beta}_l^{(k)})$ given by,

$$\bar{e}_l^{(k)}(\hat{\beta}_l^{(k)}) = RSS(\hat{\beta}_l^{(k)})/n = ||X_l^{(k)} - X_{l'}^{(k)}||/n. \tag{4.29}$$

In this study, we use these values to study several activity monitoring and recognition algorithms.

Algorithm 1 Calculate Parameters $\hat{\beta}_l^{(k)}$ for $\hat{f}_l^{(k)}$ and $\bar{e}_l^{(k)}(\hat{\beta}_l^{(k)})$

1: **procedure** ESTIMATEPARAMETERSANDERRORS
2: Input: $\boldsymbol{X}^{(k)}$
3: Output: $\hat{\beta}_l^{(k)}$ and $\bar{e}_l^{(k)}$ for all $l \in M$
4: for each $l \in M$
5: Construct $\boldsymbol{X}_{l'}^{(k)}$ from $\boldsymbol{X}^{(k)}$ using Equation (4.26)
6: Compute $\hat{\beta}_l^{(k)}$ using Equation (4.28)
7: Compute $\bar{e}_l^{k}(\hat{\beta}^{(k)})$ using Equation (4.29)
8: **end procedure**

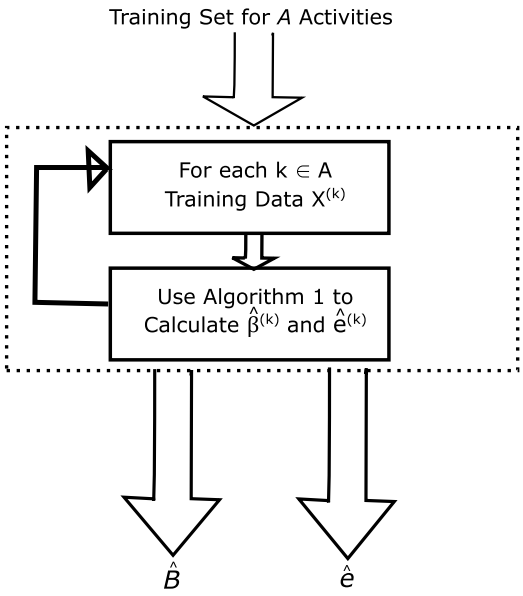


Figure 4.1: Block diagram of extraction of features for training.

4.5.2 Extraction of Features for Training

We use Algorithm 1 for extracting features from datasets for training (and monitoring or testing) (see Fig. 4.1). For each activity $k \in A$, input to the algorithm is a matrix of motion data $\boldsymbol{X}^{(k)}$. The algorithm computes m linear models and m mean residual error values for each activity. Let us denote the model parameters by $\hat{\boldsymbol{\beta}}^{(k)}$ as,

$$\hat{\boldsymbol{\beta}}^{(k)} = [\hat{\beta}_1^{(k)}, \hat{\beta}_2^{(k)}, \cdots, \hat{\beta}_m^{(k)}].$$

In the matrix notation this can be written as,

$$\hat{\boldsymbol{\beta}}^{(k)} = \begin{bmatrix} \hat{\beta}_{10}^{(k)} & \hat{\beta}_{20}^{(k)} & \cdots & \hat{\beta}_{m0}^{(k)} \\ \hat{\beta}_{11}^{(k)} & \hat{\beta}_{21}^{(k)} & \cdots & \hat{\beta}_{m1}^{(k)} \\ & \cdots\cdots\cdots & & \\ \hat{\beta}_{1m}^{(k)} & \hat{\beta}_{2m}^{(k)} & \cdots & \hat{\beta}_{mm}^{(k)} \end{bmatrix}.$$

Since there are n_a activities, we call Algorithm 1 for n_a times to get the set of training parameters $\hat{B}$.

$$\hat{B} =< \hat{\boldsymbol{\beta}}^{(1)}, \hat{\boldsymbol{\beta}}^{(2)}, \cdots, \hat{\boldsymbol{\beta}}^{(n_a)} >$$

We use all $\hat{\boldsymbol{\beta}}^{(k)}$ in $\hat{B}$ for our Angle Similarity Algorithm described in Section 4.5.3.

A call to the Algorithm 1 also generates m mean residual sum of squared errors,

$$\bar{e}^{(k)} = \{\bar{e}_1^{(k)}, \bar{e}_2^{(k)}, \cdots, \bar{e}_m^{(k)}\}.$$

These mean error values are used in our other algorithms presented in Sections 4.5.4 and 4.5.5.

4.5.3 Angle Similarity Algorithm for Identification of Activities

As the name suggest, this algorithm computes angles between linear model parameters of the training sets and testing sets. Let a testing datasets be denoted by $\boldsymbol{X}$. First call Algorithm 1 with $\boldsymbol{X}$ as input. Let output parameters from the algorithm be denoted by

$$\hat{\boldsymbol{\beta}} = [\hat{\beta}_1, \hat{\beta}_2, \cdots, \hat{\beta}_m].$$

Now call Algorithm 2 for n_a times, once for each activity $k \in A$. Each call calculates m angles $[\theta_1^{(k)}, \theta_2^{(k)}, \cdots, \theta_m^{(k)}]$ for an activity k. Let us represent these set of angles as matrix $\boldsymbol{\theta}$ of n_a rows and m columns, each row represents angles for a given activity. Next we describe how these angles are used to identify activities. We have evaluated two methods.

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1^{(1)} & \theta_2^{(1)} & \cdots & \theta_m^{(1)} \\ \theta_1^{(2)} & \theta_2^{(2)} & \cdots & \theta_m^{(2)} \\ & \cdots\cdots\cdots & & \\ \theta_1^{(n_a)} & \theta_2^{(n_a)} & \cdots & \theta_m^{(n_a)} \end{bmatrix}. \tag{4.30}$$

In an ideal case, if the testing set is from activity k, the angle between the parameters of the activity and the parameters of the training activity k should be zero. However, in reality the value of these angle are not zero. Our algorithm assumes that, if the testing data came from activity k, then the angle would be minimum with training parameters for activity k and the testing parameters (see Fig. 4.2).

4.5.3.1 Minimum Sum Method

In this method, sum of all angles in a row are computed. The activity corresponding to the row that has the minimum sum is identified as the activity. The problem with this method is that there is no obvious or natural way to avoid a misclassification. The method described next overcomes this deficiency.

4.5.3.2 Voting Method

In this method, first we find the minimum angle at each column. Let us consider each minimum value as a vote, that is, if the minimum value is at row number k, then it is a vote for activity k. Since we have m columns, an activity may receive as many as m votes. After completing column-wise voting, the votes for each activity is counted. The activity that has more than half the votes, represents the activity. For example, if m is six then the winning activity (row number in our notation) must have four or more votes. If no row has more than half of the votes, then the input is not classified. Thus, in this activity recognition method we have a natural way of setting a threshold for recognition.

Algorithm 2 Calculate Angles $\theta_l^{(k)}$

1: **procedure** ANGLESBETTRAININGANDTESTDATASETS
2: Input: $\hat{\boldsymbol{\beta}}^{(k)}$ and $\hat{\boldsymbol{\beta}}$
3: Output: $\theta_l^{(k)}$ for all $l \in M$
4: for each $l \in M$
5: Calculate Angle $\theta_l^{(k)}$ between $\hat{\beta}_l^{(k)}$ and $\hat{\beta}_l$
6: **end procedure**

4.5.4 Symmetric KL-distance Algorithm

In this section we present an algorithm for identification of ADLs using symmetric KL-distance measures (introduced in Section 4.3. Recall that it is defined as $KL(p||q) = D(p||q) + D(q||p)$ for two probability mass distributions p and q. *It is important to remind the reader that this algorithm as well as the* log-sum distance *algorithm use* mean residual sum of square errors.

Let $\bar{e}^{(k)} = \{\bar{e}_1^{(k)}, \bar{e}_2^{(k)}, \cdots, \bar{e}_m^{(k)}\}$ be the *mean residual sum of square errors* for activity k. Since $\bar{e}^{(k)}$ is not a probability mass distribution, before computing KL-distance matrix $\boldsymbol{D_{KL}}$, each set of errors for an activity is normalized such that the sum of the normalized values adds to 1. Let us denote the normalized values by

$$q^{(k)} =< q_1^{(k)}, q_2^{(k)}, \cdots, q_m^{(k)} > \tag{4.31}$$

where

$$q_l^{(k)} = \frac{\hat{e}_l^{(k)}}{\sum_{i=1}^{m} \hat{e}_i^{(k)}}. \tag{4.32}$$

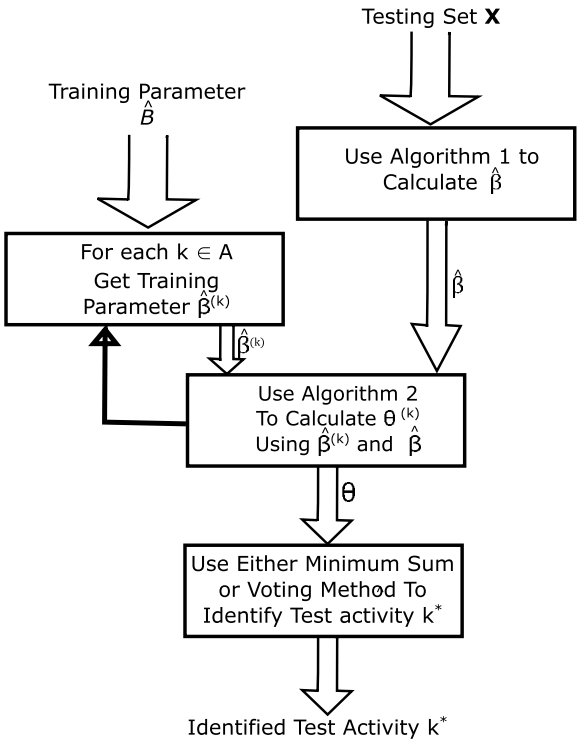


Figure 4.2: Block diagram of activity identification system using angle similarity.

Let us denote these normalized values as a matrix Q.

$$\boldsymbol{Q} = \begin{bmatrix} q_1^{(1)} & q_2^{(1)} & \cdots & q_m^{(1)} \\ q_1^{(2)} & q_2^{(2)} & \cdots & q_m^{(2)} \\ & \cdots\cdots\cdots & & \\ q_1^{(n_a)} & q_2^{(n_a)} & \cdots & q_m^{(n_a)} \end{bmatrix} \tag{4.33}$$

Then, the sets of normalized error values in the matrix Q are used to define a $(n_a \times n_a)$ distance matrix $\boldsymbol{D_{KL}}$, and it is used for identification of the activities as described below.

$$\boldsymbol{D_{KL}} = \begin{bmatrix} d_{KL}^{(1,1)} & d_{KL}^{(1,2)} & \cdots & d_{KL}^{(1,n_a)} \\ d_{KL}^{(2,1)} & d_{KL}^{(2,2)} & \cdots & d_{KL}^{(2,n_a)} \\ \cdots & \cdots & \cdots & \cdots \\ d_{KL}^{(n_a,1)} & d_{KL}^{(n_a,2)} & \cdots & d_{KL}^{(n_a,n_a)} \end{bmatrix} \tag{4.34}$$

where,

$$d_{KL}^{(i,j)} = \sum_{k=1}^{m} q_k^{(i)} \log_2 \left(\frac{q_k^{(i)}}{q_k^{(j)}} \right) + \sum_{k=1}^{m} q_k^{(j)} \log_2 \left(\frac{q_k^{(j)}}{q_k^{(i)}} \right).$$

Note that $d_{KL}^{(k,k)} = 0$ for all $k \in A$. Also, the matrix $\boldsymbol{D_{KL}}$ is symmetric, but for ease of description and programming we have used complete matrix.

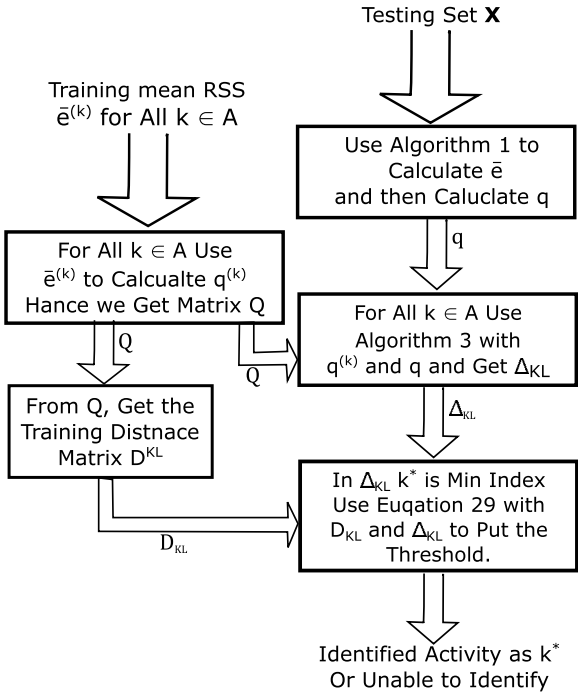


Figure 4.3: Block diagram of activity identification system using KL-Distance algorithm.

4.5.4.1 Symmetric KL-distance Algorithm

Let a testing datasets be denoted by $\boldsymbol{X}$. First call Algorithm 1 with $\boldsymbol{X}$ as input. Let output mean residual sum of squares from the algorithm be denoted by $\bar{e} = \{\bar{e}_1, \bar{e}_2, \cdots, \bar{e}_m\}$, and the corresponding normalized values be denoted by $q =< q_1, q_2, \cdots, q_m >$.

Algorithm 3 Calculate Distances $d_{KL}^{(k)}$

1: **procedure** DISTBETTRAININGANDTESTDATASETS
2: Input: $q^{(k)}$ and q
3: Output: $\Delta_{KL}^{(k)}$
4: $\Delta_{KL}^{(k)} = \sum_{l=1}^{m} \left(q_l^{(k)} \log_2 \frac{q_l^{(k)}}{q_l} + q_l \log_2 \frac{q_l}{q_l^{(k)}} \right)$
5: **end procedure**

Now call Algorithm 3 for n_a times, once for each activity $k \in A$. Each call for activity k calculates one distance value $\Delta_{KL}^{(k)}$. Let us represent these set of distances as $\Delta_{KL} = \{\Delta_{KL}^{(1)}, \Delta_{KL}^{(2)}, \cdots, \Delta_{KL}^{(n_a)}\}$. We use the set of distances in Δ_{KL} for identification of the activity from which we got the testing datasets $\boldsymbol{X}$. If k^* be the index such that, $\Delta_{DL}^{(k^*)}$ is the minimum distance in Δ_{KL}, one may select k^* as the activity. But, as shown in Section 4.7.2.2, the KL-distances among activities may vary greatly and selection of the activity may be wrong.

Threshold For increasing correct activity-detection rate, it is desirable to have a method to decrease or eliminate effect of distance variations. Our proposed method for this is described next. An implementation of the proposed method is described in Section 4.6.

We compare each element $D_{DL}^{(k^*,l)}$ in the row k^* of $\boldsymbol{D_{KL}}$ with corresponding element $\Delta_{DL}^{(l)}$ in Δ_{KL}, for all $l \neq k^*$; we accept k^* as the activity if

$$\alpha_{k^*} \cdot d_{KL}^{(k^*,l)} < \Delta_{KL}^{(l)} \tag{4.35}$$

for some predetermined threshold $\alpha_{k^*} \in (0, 1)$. One may use a simplified version of CMA-ES algorithm [62] to computer a 'good' values of α_{k^*}. However, we define an optimization function for computing optimal values of α (see Section 4.6.1). A block diagram of KL-distance and threshold based activity identification method is shown in Fig. 4.3.

Next we present an algorithm using our Log-Sum Distance measure.

4.5.5 Log-Sum Distance Algorithm

This version of the algorithm considers $DL(U||V)$ for activity identification. The algorithm is similar to the KL-distance algorithm. Instead of $\boldsymbol{D_{KL}}$, it uses $\boldsymbol{D_{LD}}$ values as defined next.

$$\boldsymbol{D_{LD}} = \begin{bmatrix} d_{LD}^{(1,1)} & d_{LD}^{(1,2)} & \cdots & d_{LD}^{(1,n_a)} \\ \\ d_{LD}^{(2,1)} & d_{LD}^{(2,2)} & \cdots & d_{LD}^{(2,n_a)} \\ & \cdots\cdots\cdots\cdots & & \\ d_{LD}^{(n_a,1)} & d_{LD}^{(n_a,2)} & \cdots & d_{LD}^{(n_a,n_a)} \end{bmatrix}.$$

where

$$d_{LD}^{(i,j)} = \sum_{k=1}^{m} \hat{e}_k^{(i)} \log_2 \left(\frac{\hat{e}_k^{(i)}}{\hat{e}_k^{(j)}} \right) + \sum_{k=1}^{m} \hat{e}_k^{(j)} \log_2 \left(\frac{\hat{e}_k^{(j)}}{\hat{e}_k^{(i)}} \right).$$

The results from empirical evaluation of these algorithms are presented in Section 4.7.2.3. In the next section we describe how a set of 'good' threshold values are selected for enhancement of performance of the KL-distance and Log-Sum Distance algorithms.

4.5.6 Hybrid Method: Log-Sum Distance with Angle Similarity

This proposed hybrid activity detection method is based on both Log-Sum distance and Angle Similarity methods. To train our recognition system we extract training mean RSS errors $\bar{e}^{(k)}$ for all $k \in A$ and parameter $\hat{B}$ from the training data. Then using training mean RSS errors $\bar{e}^{(k)}$ for all $k \in A$ we calculate training Log-Sum distance matrix $\boldsymbol{D_{LD}}$. To identify an activity from a given dataset $\boldsymbol{X}$, we extract features mean RSS errors $\bar{e}$ and parameter $\hat{\boldsymbol{\beta}}$ from the dataset. We use mean RSS errors $\bar{e}$ of $\boldsymbol{X}$ and training mean RSS errors $\bar{e}^{(k)}$ with Log-Sum distance algorithm and thresholding to identify the activity k^*. If we are able to identify the activity k^*, we stop there and select k^* as the activity for the dataset $\boldsymbol{X}$.

If we are unable to identify the activity using Log-Sum distance algorithm, we apply the Angle Similarity method with voting using parameter $\hat{\boldsymbol{\beta}}$ of $\boldsymbol{X}$ and training parameter $\hat{B}$. If Angle Similarity method can identify the activity, we select k^* as the testing activity. If both methods result the test dataset as unable to identify, we select that activity testing dataset $\boldsymbol{X}$ as unclassifiable. The Hybrid activity detection process is shown in Fig. 4.4.

4.5.7 Complexity Analysis of Activity Detection Algorithm

It can be observed from Equation 4.14 that the execution time of the Log-Sum distance algorithm grows linearly with the number of features m. Our feature extraction process use linear regression. And for m number of features and n number of data points Equation 4.8 has time complexity of $O(m^2n)$. Hence, feature extraction for n_a activities using Algorithm 1 has time complexity of $O(n_am^2n)$. In all practical situations, $n >> n_a$ and $n >> m$. Angle calculation from two vectors are linear time operation. Our Algorithm 2 calculates m angles and total time complexity for this algorithm is $O(m^2)$. In practical situ-

ations, number of features m and number of activities n_a are very small. In our experiment maximum value we have used for m and n_a are 25 and 13, respectively.

In the next section we present method we used to determine threshold values for decreasing misclassification.

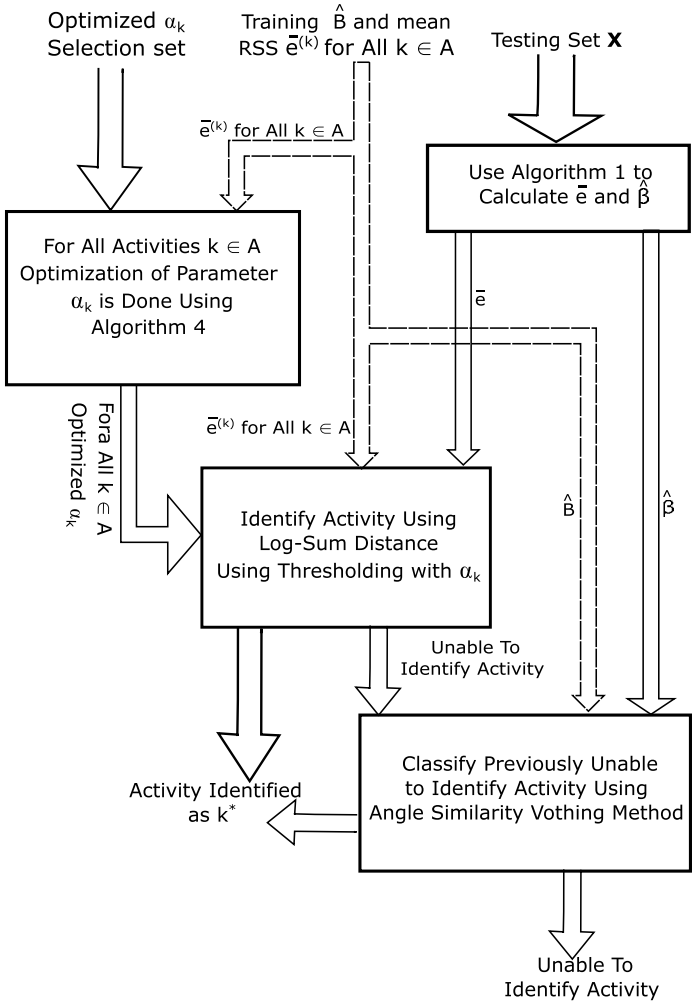


Figure 4.4: Block diagram of proposed hybrid activity classification method.

4.6 Computation of Optimal Threshold Values

The algorithms we have described in Section 4.5 can identify activities with high accuracy, but still incorrect classification is not eliminated. To eliminate or reduce incorrect classification, we proposed to use a threshold value α_k, for activity $k \in A$, in Section 4.5.4 (Equation 4.35). During classification an input that does not meet the threshold criterion, we label it unclassified. We believe, in most cases failure to classify an activity is preferable over misclassification. For setting an automatic threshold value of the parameter α_k in Equation 4.35 we first define an optimization function. The value of the function is minimum when the value of the parameter is optimal. Thus, if we start with a value of zero for α_k and its value is increased, the function will monotonically decrease until the optimal point is reached. From the optimal point, the value of the function monotonically increases as the value of α_k is increased.

4.6.1 Optimization Function

In this section we define an optimization function for searching an optimal parameter value α_k for activity k. For $\alpha_k = 0$, let us assume that C and M be the number of correctly and incorrectly classified inputs, respectively. Now for $0 < \alpha_k < 1$, let $C(\alpha_k)$, $M(\alpha_k)$, and $U(\alpha_k)$ be the number of correctly, incorrectly and unclassified inputs, respectively. Note that $(C - C(\alpha_k))$ is the number of correct classification reduction, and similarly $(M - M(\alpha_k))$ is the number of misclassification reduction. Both the reductions contribute to increase in the number of unclassified inputs. Let us define a function $f(\alpha_k)$,

$$f(\alpha_k) = \gamma_c(C - C(\alpha_k)) + \gamma_m(M - M(\alpha_k)) - \gamma_u * U(\alpha_k) \tag{4.36}$$

where γ_c, γ_m, and γ_u are suitable positive constants.

Setting appropriate values of these constants is very important. Let us examine what happens when a non-zero value for α_k is introduced. Suppose when α_k equal to zero and an input $\boldsymbol{X}$ was misclassified, but for a non-zero value of α_k, $\boldsymbol{X}$ is not classified; thus, a decrease in misclassification group has resulted into an equal increase in unclassified group. For this shifting of an input from misclassified group to unclassified group we want to reduce the value of $f(\alpha_k)$; for this to happen we must have $\gamma_m < \gamma_u$.

On the other hand, any reduction of correct classifications must increase the value of the function. With similar argument it can be established that $\gamma_c > \gamma_u$. From these two conditions we get $\gamma_c > \gamma_u > \gamma_m$. For the experiments reported in Section 4.7, we have used $\gamma_c = 3, \gamma_u = 2$, and $\gamma_m = 1$.

4.6.2 Algorithm for Computation of a Threshold

A block diagram of the proposed optimal value computation algorithm using optimization function defined earlier is shown in Fig. 4.5. We use 50% of the data for computing a set of optimal value of α_k for $k \in A$ (the other 50% of the data is used for performance evaluation). Let $P^{(k)}$ be the set of datasets to be used for computing optimal value of α_k for the activity k.

Algorithm 4 shows high-level pseudo code for computation of optimal value of the parameter α_k. Input to the algorithm is the trained distance matrix, D_{KL}, and the datasets for the activity k, $P^{(k)}$. Output from the algorithm is the optimal value α_k for the k. For ensuring at least one iteration set $f^{(last)}$, the last computed value to = ∞. Initial value of α_k and the $\alpha^{(last)}$ is set to zero. Now α_k is used for computing number of correct and incorrect classification C and M, respectively. Then $f(\alpha_k = 0)$ is calculated. The `while` loop is

continued until the last computed value of the optimization function $f(\alpha_k) < f(\alpha^{(last)}$, that is, the value of the optimization function continues to decease. This algorithm is called n_a times for computing optimal thresholds α_k for all n_a activities. Each of the KL-Distance and Log-Sum Distance methods requires one set of threshold values. Here we describe the method for KL-Distance method only, because the procedure for computing that for the Log-Sum Distance method is almost identical.

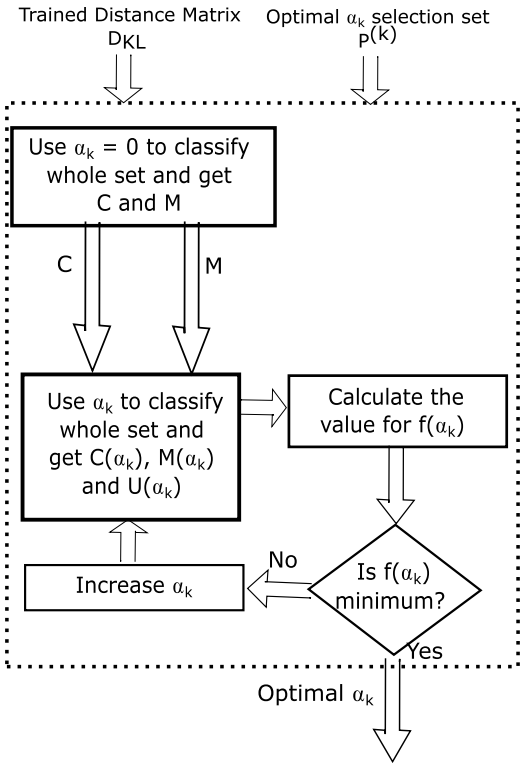


Figure 4.5: A block diagram of the proposed parameter optimization algorithm.

In the next section we report typical results from empirical evaluate the proposed Log-Sum Distance algorithm and other algorithms.

Algorithm 4 Optimize Parameter α_k for the Activity k

procedure PERAMETEROPTIMIZATION
    Input: D_{KL} - Trained distance matrix, and $P^{(k)}$
    Output: α_k
    Set $f^{(last)} = \infty$ and $\alpha^{(last)} = \alpha_k = 0$
    Using all $\boldsymbol{X}^{(\boldsymbol{k})} \in P^{(k)}$ compute C and M
    Compute $f(\alpha_k)$
    while $f(\alpha_k) < f^{(last)}$
        Increase α_k, $f^{(last)} = f(\alpha_k)$
        Using all $\boldsymbol{X}^{(\boldsymbol{k})} \in P^{(k)}$
            compute $C(\alpha_k)$ and $M(\alpha_k)$
        Compute $f(\alpha_k)$
    $\alpha_k = \alpha^{(last)}$
end procedure

4.7 Empirical Evaluation of Proposed Algorithms

In this section we report the empirical evaluation of our proposed algorithms and methods. First, we describe the experimental setup and data sources for empirical evaluations. Next, we discuss observations from the experimental evaluations, and compare and contrast performances of the proposed algorithms and methods with the KL-distance method.

To be more specific, we present results for two different angle similarity methods, symmetric KL-distance method, and Log-Sum Distance methods. Both KL-distance method and Log-Sum Distance method have been evaluated with two different variations for each: one variation uses no threshold values for reducing incorrect classifications, and the other

utilizes threshold values to reduce incorrect classifications. As discussed, the reduction of incorrect classification is very significant.

4.7.1 Experimental Settings

A list of activities used in our study is shown in Table 4.1. As can be seen from the table, one group of datasets has data for six activities and the other group of datasets has data for 13 activities. The table introduces the abbreviations used for reporting results as well as identifies activities in datasets in each group. The first group of datasets were collected in our lab and it is referred to as *UM data* in the discussions. The other group of datasets has been obtained from Berkley *WARD* database [30] and it is referred to as Wearable Action Recognition Database *WARD* in the discussions. The *WARD* database has datasets for 13 activities. The set of activities includes three resting activities —1. standing still (SS), 2. sitting still (ST), and 3. laying down on back (LD) — and ten active daily life activities — 1. walk forward (WF), 2. walk forward left-circle (WL), 3. walk forward right-circle (WR), 4. turn left (TL), 5. turn right (TR), 6. go upstairs (UP), 7. go downstairs (DN), 8. jog (JG), 9. jump (JP), 10. push wheelchair (PW).

4.7.1.1 Experimental Device Setup and Data sources

For collecting *UM data* we assembled a wireless motion sensor device with a 9DOF Razor IMU (that captures 9-axis motion data), a SparkFun Bluetooth Mate Gold transreceiver for wireless networking, SparkFun USB LiPoly Charger and battery pack, and a SparkFun FTDI Basic Breakout (for programming the IMU). To collect data, a we placed the sensor device on the ankle of the subject. We also used a USB bluetooth transreceiver to log sensor readings in a laptop computer. Each 9DOF Razor IMU via its bluetooth transre-

Table 4.1: Activities, their abbreviations, and types of activities included in the datasets.

Activity	**Abbreviation**	**UM Data**	**WARD**
Standing Still	SS	x	x
Sitting Still	ST	x	x
Laying Down on Back	LD	x	x
Walk Forward	WF	x	x
Walk Forward Left-Circle	WL		x
Walk Forward Right-Circle	WR		x
Turn Left	TL		x
Turn Right	TR		x
Go Upstairs	UP	x	x
Go Downstairs	DN	x	x
Jog	JG		x
Jump	JP		x
Push Wheelchair	PW		x

ceiver connects with the USB bluetooth transreceiver and creates a wireless sensor network (WSN). For the experiment we use only one wireless sensor node but the system we have developed can connect as many as 10 sensor nodes.

Also, we have developed a software tool on the data receiving computer. The software tool runs on Microsoft windows operating system and is capable of simultaneously collecting data from multiple wearable sensor devices. The software keeps track of the data sending and receiving times and records these times.

For the results reported here, we setup and enable the motion-sensing device to sample at $20ms$, which is equivalent to $50Hz$ sampling rate. The device records nine motion data per sample; three-axis accelerometer readings are in meter per square second (m/s^2), 3-axis gyroscope readings are in radian per square second ($rads/s^2$), and 3-axis magnetometer readings are in Tesla (T). For results reported here, we exclude the magnetometer readings. Therefore, our input data vectors are in $\mathbb{R}^6$ and they are consist of accelerometer and gyroscope readings.

4.7.1.2 Berkeley WARD Database

University of California at Berkeley has a Wearable Action Recognition Database [30]. This benchmark database contains datasets for 13 activities of daily living. Five motion sensing devices are placed on different location on the body of a subjects while the subject performs a designated activity. Each device has two sensors: (i) one 3-axis accelerometer and (ii) one 2-axis gyroscope. So, their input data vectors are in $\mathbb{R}^5$. It is a stable data set for a quantitative comparison of algorithms for human activity recognition using wearable motion sensors.

4.7.1.3 Length of Time Series for Evaluations

Since recognition should be near real-time, it is important that the number of samples to be used for testing should be small. We evaluated all algorithms with different lengths of the time series. We observed that a minimum of 250 samples are necessary for consistent high recognition rates. In general, the recognition rates increases with increase of the length of the time series. It worth noting that for a given datasets to test and train as the length of the sequence is increased, the number of testing samples decreases. We found that if

450 samples are used, the there are enough samples in both UM and WARD datasets for testing and training. Unless otherwise stated, all the results reported here use 450 samples for training and identification of activities.

We selected a location from the time series at random and used 450 samples for training. For testing, we use over lapping windows of 450 samples for each test.

4.7.2 Performance Evaluation with UM datasets

4.7.2.1 Angle Similarity

We use angle between training and testing vectors to identify activity as discussed in Section 4.5.3. We report results for the *Minimum Sum Method* in Table 4.2. As we can see from the result table this method has no option for avoiding wrong identification and overall performance of this method is not satisfactory. Only going downstairs (DN) can be identified with 100% accuracy.

Table 4.2: Performance of Angle Similarity algorithm with *Minimum Sum Method* for UM data.

	SS	ST	LD	UP	DN	WF
SS	51.7	27.8	11.2	0	0	9.3
ST	8.4	58.2	33.4	0	0	0
LD	6.9	6.28	86.82	0	0	0
UP	0	0	0	95.7	0	4.3
DN	0	0	0	0	100	0
WF	0	0	15.1	0	6.1	78.8

The results for the *Voting Method* is shown in Table 4.3. The threshold for classification was set to 50%, i.e., an activity must receive more than 50% votes to be successfully clas-

sified. For a given input data if none of the activities received more than 50% votes, then the data was not classified and placed in the unclassified group. From the table we observe that for going upstairs (UP) and going downstairs (DN) 100% of the testing samples are correctly classified. For lying down (LD) 95.66% of the samples are classified and 4.34% of the samples are placed in the unclassified group. Also out of the 95.66% samples, only 5.22% samples are incorrectly classified. Other three activities have lower classification rates but using voting threshold of 50% incorrect classifications are decreased.

Evaluations of both the methods reveal that the *Voting Method* identifies activities more accurately than the *Minimum Sum Method*.

Table 4.3: Performance of Angle Similarity algorithm with *Voting Method* for UM data.

	SS	ST	LD	UP	DN	WF	unclassified
SS	48.96	0	8.33	0	0	9.38	33.33
ST	0	59.2	34.4	0	0	0	6.4
LD	0	5.22	90.44	0	0	0	4.34
UP	0	0	0	100	0	0	0
DN	0	0	0	0	100	0	0
WF	0	0	5.08	0	0	77.97	16.95

4.7.2.2 Symmetric KL-distance

Table 4.4 shows the KL-distance between six activities in the UM data. It is interesting to note that mutual distances among three sedentary activities (Standing Still (SS), Seating Still (ST), and Laying Down (LD)) are greater than their distances with three mobile activities (Go Upstairs (UP), Go Downstairs (DN) and Walk Forward (WF). Also, three

mobile activities have relatively smaller distance among themselves. However, as we see next, classification performances for all activities are quite good.

Activity classification for the UM data is presented in Table 4.5. For three activities —sitting still (ST), laying down (LD) and going upstairs (UP) — all testing examples are classified correctly. For standing still (SS) and walking forward (WF) correct classification rates are 98.8% and 90.8%, respectively. But for going down (DN) incorrect classification is very high about 65%. Thus, for any practical application this classification rate needs improvement. Fortunately, as we present later, the classification rates can be improved with thresholding technique, if values of parameters are selected wisely.

Table 4.4: KL-distance matrix for UM data.

	SS	ST	LD	UP	DN	WF
SS	0	1.0421	3.3658	0.3639	0.5743	0.6710
ST	1.0421	0	4.1064	1.9071	2.8489	3.1006
LD	3.3658	4.1064	0	2.5690	5.3749	5.3472
UP	0.3639	1.9071	2.5690	0	0.5737	0.5832
DN	0.5743	2.8489	5.3749	0.5737	0	0.0066
WF	0.6710	3.1006	5.3472	0.5832	0.0066	0

4.7.2.3 Log-Sum Distance

The Log-Sum Distance between six activities are shown in the Table 4.6. All the distances have a multiplicative factor of 10^6. From the distance matrix we see that it has two different patterns: sedentary activities have relatively smaller distances among themselves,

Table 4.5: Performance of KL-distance from UM data without threshold

	SS	ST	LD	UP	DN	WF
SS	98.8	1.2	0	0	0	0
ST	0	100	0	0	0	0
LD	0	0	100	0	0	0
UP	0	0	0	100	0	0
DN	0	0	0	0	45.44	64.56
WF	0	0	0	0	9.2	90.8

but they have higher distances from mobile activities. Similarly, mobile activities have relatively smaller distance among themselves.

Table 4.6: Log-Sum distance matrix from UM data. The numbers have a multiplicative constant of 10^6.

	SS	ST	LD	UP	DN	WF
SS	0	0.000035	0.000056	18.875321	35.008325	81.066996
ST	0.000035	0	0.000061	21.857380	40.797971	93.753411
LD	0.000056	0.000061	0	20.451555	40.031396	91.896320
UP	18.875321	21.857380	20.451555	0	1.599367	7.824626
DN	35.008325	40.797971	40.031396	1.599367	0	2.698625
WF	81.066996	93.753411	91.896320	7.824626	2.698625	0

Table 4.7 shows testing results for the proposed Log-Sum distance method. Classification rates for five of the six activities are perfect. For standing still, 91.67% of the testing samples are correctly classified. This result demonstrates that the proposed Log-Sum method has much better classification rates than the KL-distance method.

Table 4.7: Performance of proposed Log-Sum Distance algorithm from UM data without threshold.

	SS	ST	LD	UP	DN	WF
SS	91.67	8.33	0	0	0	0
ST	0	100	0	0	0	0
LD	0	0	100	0	0	0
UP	0	0	0	100	0	0
DN	0	0	0	0	100	0
WF	0	0	0	0	0	100

4.7.2.4 Symmetric KL-Distance with Optimized Threshold Value for Each Activity

The incorrect classification rates reported in Section 4.7.2.2 can be reduced by using a threshold value for each activity. Using the optimization method described in Section 4.6, we computed an optimal parameter value for each activity. Values of the optimal parameters and classification performances are reported in Table 4.8. Optimal threshold parameter values correctly classified all testing samples for four of the six activities. Also, it decreased incorrect classification rates for other two activities.

4.7.2.5 Log-Sum Distance with Optimized Threshold Value for Each Activity

Table 4.9 shows the testing results for the proposed Log-Sum Distance method when individual threshold value for each activity is used. The method correctly classified all testing samples for all 6 activities without any misclassification. To be more specific, except standing still, 100% testing samples are classified correctly. Standing still (SS) has about 8.33% testing samples unclassified. But the most important observation is that no testing sample is classified incorrectly.

Table 4.8: Performance of KL-Distance from UM data for training and testing size of 450 and using optimized α_k

	SS	ST	LD	UP	DN	WF	unclassified
SS	98.8	0	0	0	0	0	1.2
ST	0	99.8	0	0	0	0	0.2
LD	0	0	100	0	0	0	0
UP	0	0	0	100	0	0	0
DN	0	0	0	0	40.2	23.4	36.4
WF	0	0	0	0	1.8	64.67	33.53
α_k	0.28	0.65	0.29	0.33	0.68	0.71	

4.7.2.6 Hybrid Method: Log-Sum Distance with Angle Similarity

Results from combination of Log-Sum Distance method and Angle Similarity is presented in Table 4.10. In this experiment we have individual threshold value (α_k) for each activity and voting method for angle similarity. The hybrid method improves result significantly; especially, unable to classify for standing still (SS) is reduced from 8.33% to only 1.18%.

4.7.3 Performance Evaluation with WARD datasets

In this section we report performances of the proposed methods with Berkeley WARD database. *Recall that the data was collected with five sensor devices, and each device had five IMUs. Thus, while reading this section one must not directly compare observations from WARD data with that from UM data.* We report results only for KL-distance and

Table 4.9: Performance of proposed Log-Sum Distance algorithm from UM data for optimized α_k

	SS	ST	LD	UP	DN	WF	unclassified
SS	91.67	0	0	0	0	0	8.33
ST	0	100	0	0	0	0	0
LD	0	0	100	0	0	0	0
UP	0	0	0	100	0	0	0
DN	0	0	0	0	100	0	0
WF	0	0	0	0	0	100	0
α_k	0.25	0.65	0.29	0.31	0.35	0.32	

Log-Sum Distance methods, since they provide better classifications than angle similarity method.

4.7.3.1 Performance of Symmetric KL-Distance with WARD datasets

Table 4.11 demonstrates the testing results for the KL-distance method without threshold. It is clear that correct classification rates are quite high. Although we have some misclassifications for different activities, overall average classification accuracy is over 96%. Table 4.12 demonstrates the testing results using optimal α_k for each activities. All activities are detected with very high accuracy and misclassification rate is 0.8% or lower.

4.7.3.2 Performance of Log-Sum Distance with WARD datasets

Table 4.13 demonstrates the testing results for our proposed Log-Sum Distance method without threshold for WARD dataset. The performance is better than the KL-distance method. To be more specific, Log-Sum Distance method has incorrect classifications at 21

Table 4.10: Performance of proposed Hybrid method that utilizes Log-Sum Distance algorithm with optimized α_k and Angle Similarity methods.

	SS	ST	LD	UP	DN	WF	unclassified
SS	98.8	0	0	0	0	0	1.2
ST	0	100	0	0	0	0	0
LD	0	0	100	0	0	0	0
UP	0	0	0	100	0	0	0
DN	0	0	0	0	100	0	0
WF	0	0	0	0	0	100	0
α_k	0.25	0.65	0.29	0.31	0.35	0.32	

locations compared with 38 locations for the KL-distance method (see Tables 4.11, and 4.13). The performance of the Log-Sum Distance method improves significantly when optimal threshold values are introduced to avoid misclassification (see Table 4.14). There are only 5 incorrect entries compared to 12 that for KL-distance method. Moreover, maximum rate of incorrect classification is 0.1% compared to that for KL-distance method is 0.8%.

4.8 Conclusion

Automatic identification of daily life activities can be used for promotion of healthier physical activities and lifestyle. There are many inexpensive wireless motion sensing devices or one can be assembled using off-the-shelf hardware components. These sensors can be used to make small wearable devices and collect motion data for monitoring regular human activities. For the experiment, we have assembled and programmed components to make a wearable motion sensing device with WSN to collect data. We have used those

Table 4.11: Performance of KL-Distance method without threshold values for WARD database.

	SS	ST	LD	WF	WL	WR	TL	TR	UP	DN	JG	JP	PW
SS	98.8	0	0.8	0	0.4	0	0	0	0	0	0	0	0
ST	0.1	91.5	0	0	0	0	0	1.4	0	0	6.4	0.6	0
LD	1.4	0	97.5	0	0	0	0	0	0	0	0	0	1.1
WF	2.7	0	0	95.4	0.3	1.4	0	0	0	0.2	0	0	0
WL	0	0.8	0	0.4	96.0	1.1	0	0	0.1	0	0	0	1.6
WR	0	0	0	0.3	0.2	96.5	0	0	0	0	0	2.2	0
TL	0	0	1.4	0	0.7	0	95.1	2.9	0	0	0	0	0
TR	0	2.7	0	0	0	0	0.8	94.2	0	0	0	2.3	0
UP	0	0	0	0	0	0	1.0	0	97.6	0.5	0.9	0	0
DN	0	0	0	0.3	0.6	0	0	0	0	97.7	0	0	1.4
JG	0	0	0	0	0	0	0	0.6	0	0	98.6	0.8	0
JP	0	0	0	0	0	0	0	0	0.9	0	0	99.1	0
PW	0	1.8	0	0	0.5	3.1	0	0	2.2	0	0	0	92.4

collected data and also one openly available benchmark data for performance analysis of our proposed method over other similar methods.

There are a number of human activities where corresponding motion sensor readings are very similar. Previous activity detection algorithms classified each input to one of the given set of activities, which resulted into incorrect classifications. It is believed that in real-life applications incorrect classification more harmful than avoiding classification. Here, we have proposed a threshold based method for decreasing incorrect classification rates. We have proposed a method for computing optimal threshold values that are used by the detection algorithm during classification.

We have proposed and implemented a new mathematical method for distance based classification. In the experiment, the new proposed Log-Sum Distance method has shown a promising performance over other similar distance based methods. We have tested this

Table 4.12: Performance of KL-Distance method with threshold values for WARD database

	SS	ST	LD	WF	WL	WR	TL	TR	UP	DN	JG	JP	PW	unclassified
SS	97.2	0	0	0	0	0	0	0	0	0	0	0	0	2.8
ST	0	91.5	0	0	0	0	0	0	0	0	0	0	0	8.5
LD	0.2	0	96.5	0	0	0	0	0	0	0	0	0	0	3.3
WF	0	0	0	95.2	0	0.1	0	0	0	0	0	0	0	4.7
WL	0	0.8	0	0	95	0	0	0	0	0	0	0	0	4.2
WR	0	0	0	0	0	96.1	0	0	0	0	0	0.3	0	3.6
TL	0	0	0	0	0	0	95	0.2	0	0	0	0	0	4.8
TR	0	0.6	0	0	0	0	0	93.6	0	0	0	0	0	5.8
UP	0	0	0	0	0	0	0.2	0	97.2	0	0	0	0	2.6
DN	0	0	0	0	0.6	0	0	0	0	97.4	0	0	0	2.0
JG	0	0	0	0	0	0	0	0	0	0	96.6	0	0	3.4
JP	0	0	0	0	0	0	0	0	0.1	0	0	99	0	1.0
PW	0	0.4	0	0	0.1	0.1	0	0	0	0	0	0	92.4	7.0
α_k	0.41	0.62	0.53	0.45	0.32	0.39	0.52	0.45	0.34	0.31	0.39	0.46	0.54	

new method for regular human activities classification and it shows superior classification capability over KL-Distance and Angle Similarity methods. This performance inspires us to develop algorithm in other research areas using Log-Sum Distance measure. In the next chapter we develop Log-Sum Distance based algorithm to identify repetitive muscle contractions using EMG recording.

Table 4.13: Performance of proposed Log-Sum Distance algorithm without threshold values for WARD database

	SS	ST	LD	WF	WL	WR	TL	TR	UP	DN	JG	JP	PW
SS	100	0	0	0	0	0	0	0	0	0	0	0	0
ST	8.75	91.25	0	0	0	0	0	0	0	0	0	0	0
LD	0	7.5	92.5	0	0	0	0	0	0	0	0	0	0
WF	0	0	0	95.8	2.9	1.3	0	0	0	0	0	0	0
WL	0	0	0	0	97.6	1.7	0.2	0	0	0	0	0	0.5
WR	0	0	0	0	1.4	98.4	0	0	0	0	0	0	0.2
TL	0	0	0	0	0	0	98.8	1.2	0	0	0	0	0
TR	0	0	0	0	0	0	2.1	97.9	0	0	0	0	0
UP	0	0	0	0	0.2	0	0	0	98.8	0.8	0	0	0.2
DN	0	0	0	1.2	0.1	0	0	0	0.1	98.6	0	0	0
JG	0	0	0	0	0	0	0	0	0	0	100	0	0
JP	0	0	0	0	0	0	0	0	1.3	0.3	0	98.4	0
PW	0	0	0	0	0	2.8	0	0.2	0	0	0	0	97.0

Table 4.14: Performance of proposed Log-Sum Distance algorithm with threshold values for WARD database

	SS	ST	LD	WF	WL	WR	TL	TR	UP	DN	JG	JP	PW	unclassified
SS	95.3	0	0	0	0	0	0	0	0	0	0	0	0	4.7
ST	0	90.2	0	0	0	0	0	0	0	0	0	0	0	9.8
LD	0	0.1	92.5	0	0	0	0	0	0	0	0	0	0	7.4
WF	0	0	0	94.8	0	0	0	0	0	0	0	0	0	5.2
WL	0	0	0	0	96.7	0	0	0	0	0	0	0	0	3.3
WR	0	0	0	0	0	95.4	0	0	0	0	0	0	0.1	4.5
TL	0	0	0	0	0	0	96.8	0	0	0	0	0	0	3.2
TR	0	0	0	0	0	0	0	95.9	0	0	0	0	0	4.1
UP	0	0	0	0	0	0	0	0	98.8	0	0	0	0	1.2
DN	0	0	0	0.1	0	0	0	0	0.1	96.5	0	0	0	3.3
JG	0	0	0	0	0	0	0	0	0	0	100	0	0	0
JP	0	0	0	0	0	0	0	0	0.1	0	0	97.8	0	2.1
PW	0	0	0	0	0	0	0	0	0	0	0	0	96.2	3.8
α_k	0.38	0.61	0.57	0.43	0.34	0.42	0.50	0.47	0.31	0.36	0.24	0.35	0.37	

CHAPTER 5

Muscle Contractions Detection from EMG Recordings

In the previous chapter, we presented Log-Sum Distance measure, ADL identification algorithms using Lo-Sum Distance, and extensive experimental evaluations of these algorithms using two datasets. In this chapter, we use Log-Sum Distance Measure for locating rhythmic repetitive muscle contractions in one or multiple muscles from EMG singnals. Also, we propose a method for identifying the muscle contracts before other muscles; the the muscle that contracts first, probably, triggers contractions of other muscles.

5.1 Introduction to the problem

Individual paralyzed because of spinal cord injury can be affected by different kind of involuntary muscle activities. Rhythmic repetitive muscle contractions within short period of time are one of the involuntary muscle activities that might interfere with the SCI individual's normal living. By analyzing long-term EMG recordings of the paralyzed muscle, it might be possible to measure the muscle contraction frequencies, durations, co-activity and also their severity.

Over the past few decades, a large number of studies have been dedicated to involuntary muscle activities. Many of them have used EMG recordings and for these they have

developed techniques for analysis of EMG signals [63]. EMG recordings have been used to identify spasms, motor unit activities, durations of spasms, number of contractions, start and end of contractions etc. However, these studies have used EMG-signals of a single channel. In the case of EMG data from channels, signal from each channel is analyzed at a time. We have review research on involuntary muscle activities using EMG recordings in Section 2.3.

To determine the characteristics of involuntary muscle activities, particularly how common they are, and the prevalence of different types of contractions, long-term (24-h) electromyographic recordings are effective. These long-term recordings generate large datasets, and multiple channels of EMG only compound the amount of data that must be processed.

5.1.1 Related Review of Involuntary Muscle Contractions Research

As described by Christine et. al in their research [10, 11, 64], involuntary muscle contractions or muscle spasms are quite common in SCI individuals. It may occur throughout the day and / or night, and can be uncomfortable and hamper regular normal living of the paralyzed individual. There are three types of muscle spasms: (i) Unit, (ii) Tonic, and (iii) Clonus [10]. Unit and tonic spasms are most common types of spasms, but clonus spasm also prevalent and interfere with the paralyzed person's normal activities. Fortunately, clonus spasms have different type of contraction patterns in the EMG recordings and require special processing to detect it. In this research, we focus on detection of the contraction patterns that are characteristics of clonus spasms. In the rest of the dissertation, repetitive contractions observed in clonus spasms are referred to as contractions.

Chaithanya et al [11] has defined clonus spasms as repetitive contractions followed by periods of relative muscle silence and in the EMG recording it appears as bursts of EMG.

They identify clonus spasms using Morlet wavelet filtering and then measures different statistical properties of those spasms. Among different types of filter they have used, non-linearly scaled wavelet filter of 74.8-193.9 Hz frequency band gave the best results. In their analysis they found that duration of one clonus spasm spans between 1.1s to 43.4s. They found that envelope of EMG signal frequencies range from $4Hz$ to $12Hz$ for a EMG bursts. Also, they found that each repetitive contractions or EMG burst has duration between $40ms$ to $90ms$. In our data preprocessing, we use non-linearly scaled wavelet filter before detection of EMG signal envelops, which is similar to the process described in [11].

In the EMG recording one burst of EMG represents one repetitive contraction region (see Fig. 5.1). So, in the EMG recording a clonus is represented by multiple bursts of EMG and there is a silent period between two consecutive EMG bursts. In this research we are interested in those locations where multiple bursts of EMG appear in the EMG recordings.

Figure 5.1: An example of the EMG bursts. Here we have 5 EMG bursts in this example. Each EMG burst is repetitive muscle contractions and between two EMG bursts there is a silent region.

5.1.2 Problem Statement

A Spinal cord injured individual can get affected by involuntary rhythmic repetitive muscle contractions in only one muscle or across multiple muscles at the same time. In EMG recording this rhythmic repetitive muscle contractions is shown as EMG bursts.

Given data from multiple muscles EMG recordings from an SCI individual, we address two problems: (i) How to detect presence of EMG bursts region in one or more channels of EMG recordings. (ii) If an identified region has EMG bursts in multiple channels, how to identify the channel that has the first EMG burst.

The proposed solution to first problem has three steps: (i) data preparation for feature extraction (Section 5.3.1), (ii) feature extraction (Section 5.3.2), and (iii) Log-Sum distance computation and its application to identify locations of EMG bursts (Section 5.3.4). Our algorithm for solving second problem — identification of EMG channel that has first contraction – utilizes envelops of EMG signals and the Log-Sum distance (Section 5.3.4).

5.2 Eigenvalue Decomposition for Feature Extraction

Let $X = [X_1, X_2, ...X_m]$ be a matrix of n rows and m columns, where each $X_j = [x_{1j}, x_{2j}, \cdots, x_{nj}]^T$, for $1 \leq j \leq m$, is a column of n elements. Let A is a square symmetric matrix.

$$A = X^T X; A \in \mathbb{R}^{m \times m} \tag{5.1}$$

A nonzero eigenvector z of a square matrix A is defined by the following property:

$$Az = \lambda z \tag{5.2}$$

Every square symmetric matrix is orthogonally diagonalizable. And since $A^T = (X^T X)^T = (X)^T (X^T)^T = X^T X = A$. Then when we decompose A for eigenvalues we get the following:

$$A = Z \Lambda Z^T \tag{5.3}$$

Where Z and Λ represent orthogonal and diagonal matrices respectively. As A is positive symmetric matrix, their eigenvalues are nonnegative and ordered from high to low, and first eigenvalue is the largest value.

Now the singular value decomposition of $X(n \times m)$ can be written as:

$$X = USV^T \tag{5.4}$$

Here, when $n > m$ then U $(n \times m)$ and V $(m \times m)$ are orthogonal matrices and S $(m \times m)$ is a diagonal matrix. Now if we consider the equation 5.1 using equation 5.3 and 5.4, we get the following relationship between singular values and eigenvalues [65].

$$A = X^T X = VS^T U^T USV^T = VS^2 V^T = Z\Lambda Z^T \tag{5.5}$$

The above eigenvalue is the alternative of the specific component. It is same as the squared singular value of the specific component. In the next section, we use these eigenvalues as the feature vector in our proposed Log-Sum Distance based EMG bursts algorithm.

5.3 Algorithm to Detect EMG Bursts

5.3.1 Data Preparation for Feature Extraction

In the data preparation phase we apply short-term Fourier transforms (STFTs) and wavelet filter on the each individual channel of EMG recordings. First, we use SIFTs to filter out high and low frequencies from our input EMG signals. Then we use a non-linearly scaled Morlet wavelet filter to obtain the EMG envelope. For wavelet we have used the pass-band filter of 74 - 194 Hz [11]. Figure 5.2 shows an example of the enveloped EMG data. We use EMG recording envelops for extracting features as discussed the next section.

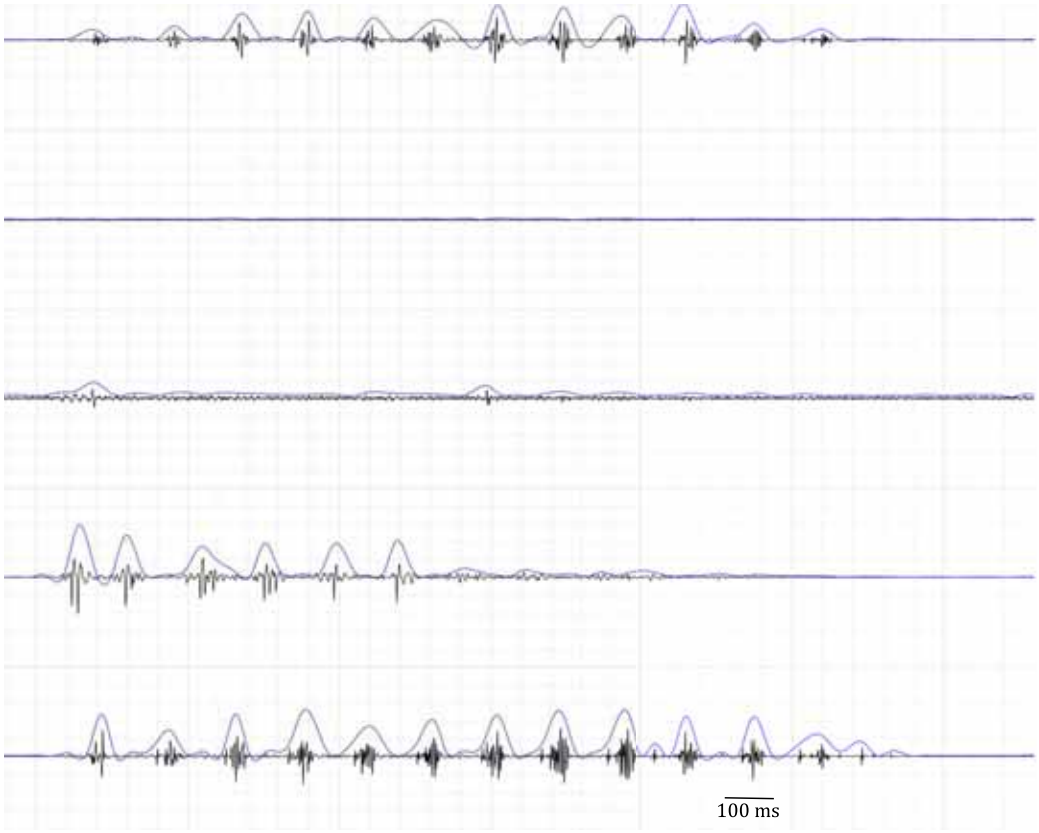


Figure 5.2: An example of EMG envelope.

5.3.2 Feature Extraction from EMG-Signal Envelops

We extract features from EMG data envelop. From given EMG recording envelops from m channels, n consecutive samples are selected from each channel to create a $n \times m$ matrix for. For our case $n >> m$. We use an overlap window of w to get the next matrix.

We use Algorithm 5 to extract features from EMG data envelop. Input to the algorithm is $X =< X^{(1)}, X^{(2)}, \cdots, X^{(p)} >$, where p is the number of matrices of size $n \times m$ from EMG recording envelops. We use equation (5.1) to get the square symmetric matrix $A^{(i)}$ and then using equation (5.5) we compute the diagonal matrix Λ and from there we get the eigenvector $z^{(i)}$. Values of eigenvector $z^{(i)}$ are positive and ordered from high to low, and

first eigenvalue is the largest value. And z represents the sequence of all eigenvectors

$$z =< z^{(1)}, z^{(2)}, \ldots, z^{(i)}, \ldots, z^{(p)} > . \tag{5.6}$$

Algorithm 5 Calculate Eigenvectors (z)

1: **procedure** EXTRACTFEATURES
2: Input: $\boldsymbol{X} =< \boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \ldots, \boldsymbol{X}^{(p)} >$
3: Output: $z = \{z^{(i)}, z^{(i)}, \ldots, z^{(i)}\}$
4: for each $\boldsymbol{X}^{(i)} \in \boldsymbol{X}$
5: Compute $A^{(i)}$ from $X^{(i)}$ using Equation (5.1)
6: Compute $z^{(i)}$ from Equation (5.5)
7: **end procedure**

Algorithm 6 Compute Log-Sum Distance (D_{LD})

1: **procedure** COMPUTELOGSUMDISTANCE
2: Input: $z = \{z^{(1)}, z^{(2)}, \ldots, z^{(p)}\}$
3: Output: $D_{LD} =< d_{LD}^{(1)}, d_{LD}^{(2)}, \ldots, d_{LD}^{(p-1)} >$
4: for $i = 1 : p - 1$
5: Compute $d_{LD}^{(i)}$ using Log-Sum Distance Equation (4.14) from $z^{(i)}$ and $z^{(i+1)}$
6: **end procedure**

5.3.3 Log-Sum Distance Algorithm to Detect EMG Bursts Locations

In the first phase of EMG Bursts location detection, we use Algorithm 6 to get the Log-Sum Distance between two adjacent matrix's eigenvectors. It takes eigenvector sequence z as an input and outputs sequence of distances D_{LD}. Two adjacent eigenvectors $z^{(i)}$ and

$z^{(i+1)}$ are used to compute their Log-Sum distance $d_{LD}^{(i)}$. Thus, a sequence of Log-Sum distances, D_{LD}, are obtained.

Figure 5.3 displays an example of 8 seconds of EMG recordings from five EMG channels, envelops of the signals, and corresponding Log-Sum distances. All the readings in the figure are normalized for the ease of presentation. Green plotted line indicates the Log-Sum Distances. It shows that Log-Sum distances higher where EMG bursts occur. In the next phase, we search through all the Log-Sum Distances, D_{LD}, to find these higher Log-Sum distance regions to locate the EMG bursts.

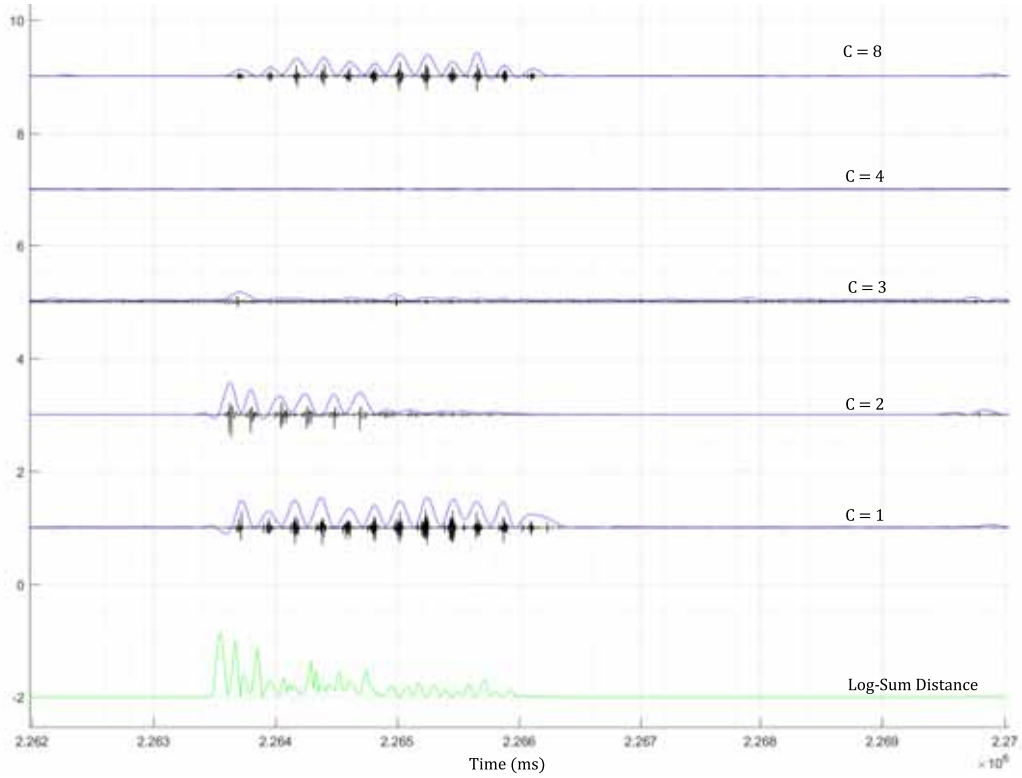


Figure 5.3: Log-Sum distance for the EMG recordings

In the second and final phase, we search though each $d_{LD}^{(i)}$ from the start position to detect EMG bursts locations. All the steps involve to detect EMG bursts locations from Log-Sum distance values D_{LD} are shown in Fig 5.4. For this purpose, we take Log-Sum

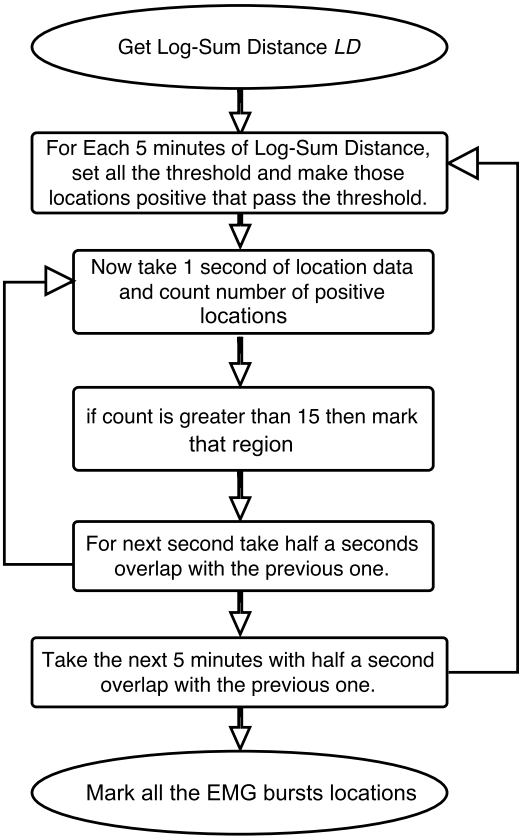


Figure 5.4: EMG bursts detection flowchart from Log-Sum Distance values

distance for each 5 minutes window of data and put thresholds for the Log-Sum Distance values. All the values $d_{LD}^{(i)}$ above the threshold value are the regions where we have active EMG signals, motor unit, or muscle contractions. So, we mark above the threshold value locations as positive and bellow the threshold value locations as zero. In the case of EMG bursts, all the higher Log-Sum Distance values are closer to each other, because lots of EMG activities happen around EMG bursts (see Fig. 5.3). Now, among those values that have passed the threshold, we search for the consecutiveness around them. For this, we go over one second of Log-Sum Distance value locations at a time to find the clusters of positive locations. If one second of Log-Sum Distance values have at least 15 positive locations, we mark that region as EMG bursts region. Also, for the next consecutiveness

search, we keep an overlap of half seconds. This increases the resolution of the search and help us to detect long-duration EMG bursts. Figure 5.5 shows a detected EMG bursts region and it is bounded inside the red curve.

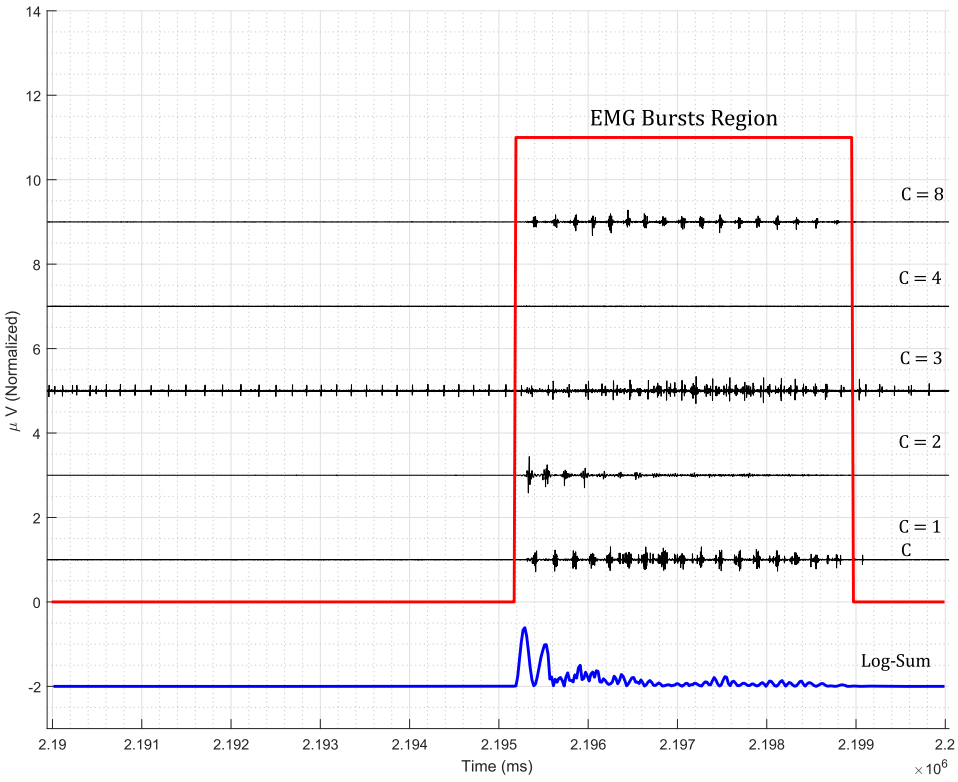


Figure 5.5: A region with EMG bursts from 5^{th} hour between 36.58 - 36.65 minutes

5.3.4 Identifying the Channel That has the First EMG Burst

In the previous Section, we identify the EMG bursts locations in one or more channels of EMG recordings. Now an algorithm is presented to identify the channel that has the first EMG burst. We consider one identified EMG bursts region at a time. To identify start of the EMG bursts, we take one identified EMG bursts region at a time and then process each channel's EMG data separately. First, we pinpoint the start of the EMG bursts in each

individual channel. For this we use the EMG data envelop (Section 5.3.1). If a channel has EMG activities and the largest peak is within the area where frequency range is from 4 to 12 Hz we identify that channel has EMG bursts. We select the start time of the first EMG burst as the EMG bursts start time. Thus, we mark all the start time in all the channels where we have EMG bursts in that identified region. Finally, we identify the channel that has the lowest start time as the channel that has the first EMG burst in the region. Figure 5.6 shows the marked start positions of a EMG bursts region. Here the *channel 2* has the first EMG burst in that region.

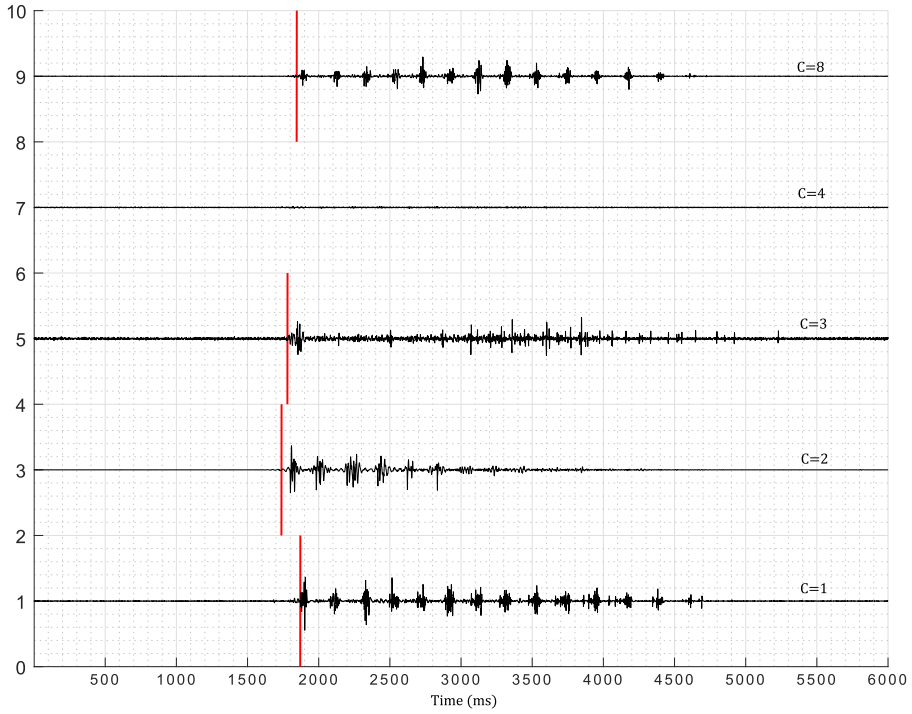


Figure 5.6: A EMG bursts with start locations marked in each channel. It is taken from 5^{th} hour between 21.23 - 21.29 minutes

5.4 Empirical Evaluation

5.4.1 Data Collection from Subjects

In our experiment we use the data that has been collected by Chaithanya and this data collection process is described in his thesis [66]. This data has been collected from spinal cord injured individual. EMG data from leg muscles was collected from seven different individuals. EMG recording was done from eight different leg muscles over 24-hours period for each individual. This data has the sampling rate of $1000Hz$ per channel. For the evaluation of our proposed algorithm and methods we only used part of the EMG data from five different leg muscles. We have used data from channels $1, 2, 3, 4$, and 8.

5.4.2 Performance Evaluation

Accuracy of our algorithm is calculated using the Equation (5.7). A person has gone over the data and manually counted the number of EMG bursts regions in one or multiple channels together and number of EMG bursts region in each channel. When our algorithm detect a EMG bursts location correctly we call it *True Positive*. *False Positive* happens when algorithm identify a EMG bursts but there is no EMG bursts in that region. When algorithm fails to identify a true EMG bursts location we denote that as *Missed*.

$$Accuracy = N_{TruePositive}/(N_{TruePositive} + N_{FalsePositive} + N_{Missed}) \tag{5.7}$$

Total EMG Bursts Regions: Table 5.1 shows the number of total EMG bursts regions identified in one or more channels using our algorithm for the five hours of data (from hour 3 to 7 from subject number 6). It also demonstrates that our method has accuracy of 79.3%.

we missed 11 EMG bursts regions out of 279. Our method has detected 59 false positive EMG bursts regions.

Table 5.1: Total EMG bursts region count and accuracy

Manual Count	Detected by Log-Sum Method	Missed by Log-Sum Method	Log-Sum Method's Accuracy(%)
279	327	11	79.3

Identification of the Channel that Starts the First EMG Burst: Table 5.2 shows number of times each of the channel has the EMG bursts region. It also shows the number of times each channels has first EMG burst in the identified regions. From the first row we observe that channel 8 has highest number of EMG bursts regions, 243 times and from the second row we note that it also has highest number of the first EMG burst, 148. Channel 2 has 106 first EMG burst among its 124 EMG bursts regions. First EMG burst has never been in channel 4 though it has 6 EMG bursts regions.

Table 5.2: Identify the muscle (channel) that triggers contractions in other muscles

	C = 1	**C = 2**	**C = 3**	**C = 4**	**C = 8**
Number of Contractions Regions	152	124	95	6	243
Starts First Contraction in the Regions	12	106	61	0	148

5.4.3 Execution Time

In this experiment we use computer that has a Intel Core $i7$-$6800K$ CPU that has $3.40GHz$ clock, $64GB$ of memory, and running 64-bit Windows 7 Enterprise operating

system. We use MATLAB $R2016b$ for programming. We process one hour of data from each of the five channels at a time, which is five hours of data. It takes us on an average 2.43 ± 0.3 minutes to process this 5 hours of data. If we extrapolate this for 24 hours of data for each of the five channels, it is will be around 63.13 ± 7.2 minutes (24×5 hours of data).

5.5 Conclusion

Identification algorithm of EMG bursts regions in one or more channels gave us a very promising results. Here we have used the Log-Sum Distance measure, presented in Chapter 4, to develop our detection algorithm. It shows a great accuracy to detect EMG bursts across multiple channels of EMG recording from SCI individuals. In this research we were also able to identify accurately those channels that has the first EMG burst in the rhythmic repetitive contractions regions.

This EMG bursts or repetitive muscle contractions, also known as clonus spasm, can hamper everyday living and activities for paralyzed SCI individuals. Knowing how common they are, their occurring time as well as their duration is expected to help the researcher and/or physicians to mitigate its negative influences on the SCI individuals.

This identification of channel that has the first muscle contractions is expected to be a new perspective for investigating into the involuntary muscle activity problem for SCI individuals. It may help those who want to understand how involuntary activities in one muscle might take a role for involuntary activities in other muscles.

CHAPTER 6

Conclusion

In this dissertation, we have developed novel algorithms to identify human fall and other activities, and to detect repetitive muscle contractions. We have presented novel learning models as well as some existing state-of-the art learning models for classification and identifying patterns from different types of sensor datasets. We have processed motion sensors data and EMG recordings from muscles to identify different events and muscle contraction locations, respectively. Extensive evaluation of our proposed algorithms and methods demonstrated that they are very effective.

First, we presented Human fall detection using motion sensors data in Chapter 3. We are able to detect fall events with 100% accuracy using semi-automatic feature extraction technique and two layer classification networks. Our one and two layers of classification networks are composed of neural network and softmax regression. We also use those to monitor activities of daily livings. ADLs were identified with two layered network with 100% accuracy.

Our success with fall and ADL detection with existing state-of-the art learning methods motivated us to develop new learning models. We presented our Log-Sum Distance measures in Chapter 4. We use the proposed Log-Sum Distance measure to develop algorithms for recognition of human activities from motion data. The sequences of m positive

numbers we used are residual sum of squares errors produced from modeling m motion time-series with multiple linear regression method. To reduce incorrect classification, we define a threshold test and use it in our proposed novel algorithm. We have defined an optimization function and used it for computing optimal threshold values. Extensive evaluation of our activity detection algorithm with two different sets of datasets show increased activity recognition rates and decreased incorrect classification rates compared to other existing methods. In one dataset, proposed algorithm detects all activities with 100% accuracy and in the another dataset, it detects all activities with 99% or higher accuracy.

In Chapter 4, we have achieved great accuracy in ADLs identification with Log-Sum Distance based learning methods. This inspired us to develop algorithm to detect repetitive muscle contractions from long-term EMG recordings using Log-Sum Distance measure. We presented the novel method to identify and analyze repetitive muscle contractions in one or more channels using EMG datasets of SCI individuals in Chapter 5. Our empirical evaluation has shown high accuracy in detection of repetitive contraction regions from EMG datasets. We also detect the start of contraction events on each channel for each detected repetitive contractions region. This way we were able to identify those channels (muscles) that are triggering first muscle contraction in the identified regions. The muscle contractions region and co-activity finding problem and its solution expected to opens a new opportunity to identify effect and/or influence spasm one channel to other channels.

Log-Sum Distance measure showed a great promise for different classification and identification problems. It is robust, easy to use, and computationally efficient to work on different sensors data. It adds a new dimension to process sensor data in a efficient way. It opens possibility to use Log-Sum Distance measure to identify valuable patterns and information in other types of sensor data.

Bibliography

[1] L. M. Bregman, "The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 3, pp. 200–217, 1967.

[2] S. Abeyruwan, F. Sikder, U. Visser, and D. Sarkar, "Activity monitoring and prediction for humans and NAO humanoid robots using wearable sensors," in *FLAIRS Conference*, 2015, pp. 342–347.

[3] J. Andreu-Perez, D. R. Leff, H. M. Ip, and G.-Z. Yang, "From wearable sensors to smart implants—toward pervasive and personalized healthcare," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 12, pp. 2750–2762, 2015.

[4] E. Sazonov and M. R. Neuman, *Wearable Sensors: Fundamentals, implementation and applications*. Elsevier, 2014.

[5] S. Mukhopadhyay, "Wearable sensors for human activity monitoring: A review," *Sensors Journal, IEEE*, vol. 15, no. 3, pp. 1321–1330, March 2015.

[6] M. Adams and A. Hicks, "Spasticity after spinal cord injury," *Spinal cord*, vol. 43, no. 10, pp. 577–586, 2005.

[7] M. Dimitrijevic, P. Nathan, and A. Sherwood, "Clonus: the role of central mechanisms." *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 43, no. 4, pp. 321–332, 1980.

[8] W. A. Cook, "Antagonistic muscles in the production of clonus in man," *Neurology*, vol. 17, no. 8, pp. 779–779, 1967.

[9] J. A. Beres-Jones, T. D. Johnson, and S. J. Harkema, "Clonus after human spinal cord injury cannot be attributed solely to recurrent muscle-tendon stretch," *Experimental brain research*, vol. 149, no. 2, pp. 222–236, 2003.

[10] J. Winslow, A. Martinez, and C. K. Thomas, "Automatic identification and classification of muscle spasms in long-term emg recordings," *IEEE journal of biomedical and health informatics*, vol. 19, no. 2, pp. 464–470, 2015.

[11] C. K. Mummidisetty, J. Bohórquez, and C. K. Thomas, "Automatic analysis of emg during clonus," *Journal of neuroscience methods*, vol. 204, no. 1, pp. 35–43, 2012.

[12] A. Merlo, D. Farina, and R. Merletti, "A fast and reliable technique for muscle activity detection from surface emg signals," *IEEE Transactions on Biomedical Engineering*, vol. 50, no. 3, pp. 316–323, 2003.

[13] K. Takada and K. Yashiro, "Automatic measurement of on/off periods of emg activity." *Medinfo. MEDINFO*, vol. 8, pp. 751–754, 1994.

[14] G. Vannozzi, S. Conforto, and T. DÁlessio, "Automatic detection of surface emg activation timing using a wavelet transform based method," *Journal of Electromyography and Kinesiology*, vol. 20, no. 4, pp. 767–772, 2010.

[15] D. M. Wallace, B. H. Ross, and C. K. Thomas, "Characteristics of lower extremity clonus after human cervical spinal cord injury," *Journal of neurotrauma*, vol. 29, no. 5, pp. 915–924, 2012.

[16] O. D. Lara and M. A. Labrador, "A survey on human activity recognition using wearable sensors," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1192–1209, 2013.

[17] L. Z. Rubenstein, "Falls in older people: epidemiology, risk factors and strategies for prevention," *Age and Aging*, vol. 35-S2, pp. 37–41, 2006.

[18] "Costs of Falls Among Older Adults," http://www.cdc.gov/ homeandrecreationalsafety/falls/fallcost.html, 2014.

[19] O. Ojetola, E. I. Gaura, and J. Brusey, "Fall detection with wearable sensors–safe (smart fall detection)," in *Intelligent Environments (IE), 2011 7th International Conference on*. IEEE, 2011, pp. 318–321.

[20] A. Leone, G. Rescio, and P. Siciliano, "Supervised wearable wireless system for fall detection," in *Measurements and Networking Proceedings (M&N), 2013 IEEE International Workshop on*. IEEE, 2013, pp. 200–205.

[21] W.-S. Baek, D.-M. Kim, F. Bashir, and J.-Y. Pyun, "Real life applicable fall detection system based on wireless body area network," in *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*. IEEE, 2013, pp. 62–67.

[22] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *PARVASIVE 2004, LNCS 3001*, A. Ferscha and F. Matten, Ed. Springer-Verlag, 2004, pp. 1–17.

[23] M. Dumitrache and S. Pasca, "Fall detection algorithm based on triaxial accelerometer data," in *E-Health and Bioengineering Conference (EHB), 2013*. IEEE, 2013, pp. 1–4.

[24] P. Kumar and P. C. Pandey, "A wearable inertial sensing device for fall detection and motion tracking," in *Proc. Annu. IEEE Conf. INDICON, Mumbai, India*, 2013, pp. 1–6.

[25] N. C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Pervasive and Mobile Computing*, vol. 10, pp. 138–154, 2014.

[26] L. Gao, A. Bourke, and J. Nelson, "Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems," *Medical Engineering & Physics*, vol. 36, no. 6, pp. 779–785, 2014.

[27] J. A. Álvarez-García, L. M. S. Morillo, M. Á. Á. de La Concepción, A. Fernández-Montes, and J. A. O. Ramírez, "Evaluating wearable activity recognition and fall detection systems," in *6th European Conference of the International Federation for Medical and Biological Engineering*. Springer, 2015, pp. 653–656.

[28] V. R. Shen, H.-Y. Lai, and A.-F. Lai, "The implementation of a smartphone-based fall detection system using a high-level fuzzy petri net," *Applied Soft Computing*, vol. 26, no. 0, pp. 390 – 400, 2015.

[29] J. A. Ward, P. Lukowicz, G. Troster, and T. E. Starner, "Activity recognition of assembly tasks using body-worn microphones and accelerometers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1553–1567, Oct 2006.

[30] A. Y. Yang, R. Jafari, S. S. Sastry, and R. Bajcsy, "Distributed recognition of human actions using wearable motion sensor networks," *Journal of Ambient Intelligence and Smart Environments*, vol. 1, no. 2, pp. 103–115, 2009.

[31] C. Strohrmann, H. Harms, C. Kappeler-Setz, and G. Troster, "Monitoring kinematic changes with fatigue in running using body-worn sensors," *IEEE Transactions on Information Technology in Biomedicine*, vol. 16, no. 5, pp. 983–990, 2012.

[32] B. Mariani, M. C. Jiménez, F. J. Vingerhoets, and K. Aminian, "On-shoe wearable sensors for gait and turning assessment of patients with parkinson's disease," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 1, pp. 155–158, 2013.

[33] D. D. Mehta, M. Zañartu, S. W. Feng, H. A. Cheyne, and R. E. Hillman, "Mobile voice health monitoring using a wearable accelerometer sensor and a smartphone platform," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 11, pp. 3090–3096, 2012.

[34] Y.-W. Bai, S.-C. Wu, and C. H. Yu, "Recognition of direction of fall by smartphone," in *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on*. IEEE, 2013, pp. 1–6.

[35] S. Steidl, C. Schneider, and M. Hufnagl, "Fall detection by recognizing patterns in direction changes of constraining forces," in *Proceedings of the eHealth 2012*. OCG, 2012, pp. 27–32.

[36] S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas, and D. J. Cook, "Simple and complex activity recognition through smart phones," in *Intelligent Environments*. IEEE, 2012, pp. 214–221.

[37] P. Maziewski, A. Kupryjanow, K. Kaszuba, and A. Czyzewski, "Accelerometer signal pre-processing influence on human activity recognition," in *Conference Proceedings of Signal Processing Algorithms, Architectures, Arrangements, and Applications (SPA)*, 2009, pp. 95–99.

[38] D. Curone, G. M. Bertolotti, A. Cristiani, E. L. Secco, and G. Magenes, "A real-time and self-calibrating algorithm based on triaxial accelerometer signals for the detection of human posture and activity," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 4, pp. 1098–1105, 2010.

[39] E. M. Tapia, S. S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *Wearable Computers, 2007 11th IEEE International Symposium on*. IEEE, 2007, pp. 37–40.

[40] Z.-Y. He and L.-W. Jin, "Activity recognition from acceleration data using ar model representation and svm," in *Machine Learning and Cybernetics, 2008 International Conference on*, vol. 4. IEEE, 2008, pp. 2245–2250.

[41] T. Huynh and B. Schiele, "Analyzing features for activity recognition," in *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies*. ACM, 2005, pp. 159–163.

[42] S. W. Abeyruwan, D. Sarkar, F. Sikder, and U. Visser, "Semi-automatic extraction of training examples from sensor readings for fall detection and posture monitoring," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5406–5415, 2016.

[43] D. Riboni and C. Bettini, "COSAR: hybrid reasoning for context-aware activity recognition," *Personal and Ubiquitous Computing*, vol. 15, no. 3, pp. 271–289, 2011.

[44] L. C. Jatoba, U. Grossmann, C. Kunze, J. Ottenbacher, and W. Stork, "Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*. IEEE, 2008, pp. 5250–5253.

[45] A. Kuznietsov and D. Neubauer, "A wireless framework for movement activity monitoring of sprinters," in *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*. IEEE, 2012, pp. 1–6.

[46] Y.-C. Kan and C.-K. Chen, "A wearable inertial sensor node for body motion analysis," *IEEE Sensors Journal*, vol. 12, no. 3, pp. 651– 657, 2012.

[47] T. Van Kasteren, G. Englebienne, and B. J. Kröse, "An activity monitoring system for elderly care using generative and discriminative models," *Personal and Ubiquitous Computing*, vol. 14, no. 6, pp. 489–498, 2010.

[48] O. D. Lara, A. J. Perez, M. A. Labrador, and J. D. Posada, "Centinela: A human activity recognition system based on acceleration and vital sign data," *Pervasive and Mobile Computing*, vol. 8, no. 5, pp. 717–729, 2012.

[49] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *IJCAI*, vol. 14, no. 2, 1995, pp. 1137–1145.

[50] P. Rack, H. Ross, and A. Thilmann, "The ankle stretch reflexes in normal and spastic subjects," *Brain*, vol. 107, no. 2, pp. 637–654, 1984.

[51] J. M. Hidler and W. Z. Rymer, "Limit cycle behavior in spasticity: analysis and evaluation," *IEEE Transactions on Biomedical engineering*, vol. 47, no. 12, pp. 1565–1575, 2000.

[52] V. von Tscharner, "Intensity analysis in time-frequency space of surface myoelectric signals by wavelets of specified resolution," *Journal of Electromyography and Kinesiology*, vol. 10, no. 6, pp. 433–445, 2000.

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

[54] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[55] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, L. Getoor and T. Scheffer, Eds. Omnipress, 2011, pp. 265–272.

[56] D. Sarkar, "Methods to speed up error back-propagation learning algorithm," *ACM Comput. Surv.*, vol. 27, no. 4, pp. 519–544, Dec. 1995. [Online]. Available: http://doi.acm.org/10.1145/234782.234785

[57] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer, 2008.

[58] G. Stewart, *Introduction to Matrix Computations*, 1st ed. Academic Press, 1973.

[59] O. Kallenberg, *Foundations of Modern Probability*. Springer Science & Business Media, 2006.

[60] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley, 2006.

[61] L. Bregman, "A relaxation method of finding a common point of convex sets and its application to problems of optimization," in *Soviet Mathematics Doklady*, vol. 7, 1966, pp. 1578–1581.

[62] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evol. Comput.*, vol. 10, no. 4, pp. 371–395, Dec. 2002. [Online]. Available: http://dx.doi.org/10.1162/106365602760972767

[63] O. D. Lara and M. A. Labrador, "Techniques of emg signal analysis: detection, processing, classification and applications," *Biological Procedures Online*.

[64] C. K. Thomas, M. Dididze, A. Martinez, and R. Morris, "Identification and classification of involuntary leg muscle contractions in electromyographic records from individuals with spinal cord injury," *Journal of Electromyography and Kinesiology*, vol. 24, no. 5, pp. 747–754, 2014.

[65] R. Bro and A. K. Smilde, "Principal component analysis," *Analytical Methods*, vol. 6, no. 9, pp. 2812–2831, 2014.

[66] C. K. Mummidisetty, "Analysis of emg during clonus using wavelets," Ph.D. dissertation, University of Miami, 2009.