Real-time physiological identification using incremental learning and semi-supervised learning

by

Shashank Shivarudrappa

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Computer and Information Science)
in the University of Michigan-Dearborn
2018

Master's Thesis Committee:

Assistant Professor Omid Dehzangi, Chair
Professor Kiumi Akingbehin
Associate Professor Brahim Medjahed

Acknowledgements

I would first like to thank my thesis advisor Prof. Omid Dehzangi of the Computer and Information Science Department at the University of Michigan-Dearborn. Prof. Dehzangi was always supportive whenever I needed guidance or suggestions about my research or writing. He consistently backed my work and steered me in the right direction throughout my thesis.

I would also like to thank my research mates, Shalini Bansal, Akash Pavate, Cayce Williams, Christian Hessler and Cheyenne Vasseli who contributed to my research through their inputs. Without their passionate contribution, I wouldn't have had enough support and motivation to see through my work. Also, the research wouldn't have been successful if not for my friends who volunteered to wear the sensors and collect the required data needed for my research. So, I would like to thank Vikas Rajendra, Muhammed Farooq, Omar Iftikhar, Likith Manjegowda, Kelsey Joseph, Selvamani, Adarsh Krishnan, Bharath Kotabagi and Arindam Banerjee.

Finally, I must express my very profound gratitude to my parents and my faculty for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Table of Contents

List of Tables

List of Figures

Abstract

The widespread usage of wearable sensors such as smart watches provide access to valuable objective physiological (such as Electrocardiogram(ECG)) signals ubiquitously. Healthcare domain has been tremendously benefited by the collection of physiological signals which can be used for health monitoring of patients. The signals from the wearable sensors enabled the researchers and data experts to process them and identify the human physiological state by classifying the human activities. This led to the growth and development of smart ecosystem in the healthcare domain.

In this thesis, ECG signals have been investigated as the physiological measure to detect human activities. Various measures are extracted from ECG, such as heart rate variability, average heart rate etc. and their relationships with different human activities are investigated. To build a comprehensive analytical machine learning model for ECG signals and to enable the continuous monitoring of humans, one would need access to real time streaming of continuous data. So, the data would be unsupervised most of the time and it would be very expensive (almost practically impossible) to label all the data streaming in real time. Also, it is highly probable that the data is collected from different sessions and varying situations. Therefore, the machine learning models need to be able to adapt to new sessions. This would be a major challenge in human state monitoring provided that the conventional predictive models work only on the stationary data. Also, these models would fail to work on the data from multiple sessions. To provide a practical solution to address above issues, two advanced methods in machine learning have been discussed in this research: Incremental learning and Semi supervised learning.

Incremental learning is a paradigm in Machine learning where the stream of input data is continuously used to extend the existing knowledge learnt by the model. The incremental learning module has been built in Apache Spark platform which provides a scalable cloud

infrastructure to apply machine learning algorithms on streaming data. Semi supervised learning is another solution implemented in this thesis where some out of all the data points are labelled. Different semi supervised algorithms have been studied and applied which learn the relationship between features and adapts the model to data from multiple sessions. Finally, the results are compared and the implementation ideas for the discussed solutions have been proposed.

Chapter 1: Introduction

1.1 Background

Technology is rapidly advancing and evolving every single day. There is essentially a race for the companies to lead in developing technologies that help with the active lifestyles. Wearable technologies that monitor health and daily activities have created a massive and quickly developing market and customer base. The growing popularity of wearable devices has opened wide prospects in health monitoring as part of smart connected health systems. The ensuing information from such systems can potentially result in ground-breaking developments in a huge array of applications such as healthcare, elderly care support, wellness, emergency response, fitness monitoring, long-term preventive chronic care, and other smart environments (Achten and Jeukendrup, 2003)(Syed and Guttag, 2011)(Salehizadeh *et al.*, 2015). Despite many interesting approaches in previous studies, there are many challenges in designing and developing smart pervasive monitoring systems which can scale large streams of behavioral and physiological signals (Banaee, Ahmed and Loutfi, 2013)(Kulkarni and Ade, 2014).

Physiological signals have been an important measure which depict the physiological process of human beings. There are various physiological measures such as heart beat rate (electrocardiogram(ECG)), respiratory rate (capnogram), skin conductance (electrodermal activity(EDA)), muscle current (electromyography(EMG)), brain electrical activity (electroencephalography(EEG)) which are popular in research activities to determine the human state or activity. Implementation of some of these techniques through portable devices could be expensive or intrusive. So, ECG has been considered for this research as it generates reliable signal which can recognize physiological changes consistently, at low cost, and with minimum intrusiveness.

An electrocardiogram (ECG) is a continuous recording of the electrical activity of the heart muscle or myocardium. During heartbeat, the cardiac muscles undergo contraction along with a sequence of depolarization and repolarization resulting in electrical waves. During depolarization, a cardiac cell generates an electrical impulse where different concentrations of ions such as sodium, potassium etc. cross the cell membrane and causes the action potential. Repolarization is the return of the ions to their previous resting state which corresponds with relaxation of the heart muscles. These changes in potential, summed over many cells, can be measured by electrodes placed on the surface of the body. For any pair of electrodes, a voltage is recorded whenever the direction of depolarization (or re-polarization) is aligned with the line connecting the two electrodes. The sign of the voltage indicates the direction of depolarization, and the axis of the electrode pair is called as the lead. Multiple electrodes along different axes can be used so that the average direction of depolarization, as a three-dimensional vector, can be reconstructed from the ECG tracings. In usual scenarios, portable ECG monitors wouldn't use multi-lead data so that the battery life can be maximized by reducing the number of electrodes used. However, obtaining the data from multiple leads could lead to better learning of the ECG patterns in terms of user's physiological state. Therefore, this study includes the analysis of data from all electrodes in ECG sensor but only for 7 hours of duration.

ECG has the advantage of being easy to acquire. The electrical activity of the heart can be measured on the surface of the body in an inexpensive manner. In real scenarios, where ECG is generally important to diagnose health parameters, the ECG is typically captured by bedside monitors, in an in-patient setting. However, in an out-patient setting, a portable ECG device worn by patients can record data continuously over long hours. This research involves a hypothetical out-patient scenario and collects data using a mobile phone. This can be considered as the personal mobile phone of the subject, where in the data streaming app is installed and the subjects is required to carry it all along.

In this thesis, data are collected from portable ECG sensors and are investigated to recognize the human activities using the physiological changes collected from the sensors. So, the subjects were made to do a series of pre-defined activities and the machine learning techniques have been implemented on the physiological data to recognize the activities. The challenges have been

discussed with respect to conventional machine learning techniques, and new methodologies have been tried out and adapted to support the hypothesis.

Already popular in health-related applications, the wearables sensors are being extended to other domains. The analysis on valuable data from wearables could be employed by industries and business and used in variety of applications like for example, building a smart environment for the users.

1.2 Problem Statement

The signals collected from the wearables could be either streaming(free-form) or isolated. The isolated data are collected under experimenter's supervision, so the data would be clearly labelled. Whereas the streaming data is collected in real time and requires continuous queries to process them. This type of data is so continuous in a way that, understanding the physiological changes in human while transitioning from one activity to another becomes important. Additionally, the analytics on streaming data adds real-time insight to decision making models. Such real-time insights would be valuable for the industries and could generate greater potential benefits for the businesses.

However, there are a lot of challenges while collecting and analyzing the streaming data. The app which collects the data should be stable and shouldn't crash during the process of data collection. Battery constraints of the mobile phone and sensors limit the duration of data collection. Also, the analytics on streaming data is a developing technology and not a lot of experts are found in this field. So, new machine learning methodologies are being implemented to work on the streaming data.

Another challenging aspect in modern machine learning applications is session to session variability. It would seem highly unlikeable that all the streaming data could come from one single session because of the constraints mentioned earlier. Additionally, there are possibilities of session disconnectivity or session crash due to multiple reasons like hardware failure, out of range between wearable sensor and data receiving equipment etc. So, in practical situations, the data would be collected in multiple sessions. Also, the isolated data and free-form data collection

represent the data from 2 different sessions. Conventional machine learning algorithms would fail to make good predictions on the data from new session which was never trained before. This drawback calls for the need of handling the variabilities between multiple sessions which is a major challenge in modern applications. The solution would be to build a model that can adapt to new sessions by transferring the knowledge from the learned session. So, the model adaptability becomes an important criterion for the development of machine learning solution to handle streaming data from multiple sessions which has been discussed in this thesis.

1.3 Related work

1.3.1 Wearable Sensors

The wearable sensing devices is making the healthcare system go through a transformation which makes the continuous monitoring of inhabitants possible even without hospitalization (Mukhopadhyay, 2014). This work talks about how advancements in sensing technologies along with necessary networking technologies and applied sciences is enabling the smart systems to monitor human activities continuously. In paper (Patel *et al.*, 2012), the authors discuss the recent developments in the field of wearable sensors and systems that relate to rehabilitation. Also, they discuss the health monitoring applications of wearable systems that employ multiple sensors integrated into a sensor network either only body worn sensor network or integration of body worn sensors with ambient sensors. However, wired sensors cannot be used for long term health monitoring, hence the evolution on wireless wearable communication technology is on rise.

Another work (Yilmaz, Foster and Hao, 2010) shows how chronic disease management has been achieved by detecting vital signs from wearable sensing devices. Such monitoring systems could be employed for efficient disease management and to prevent diseases and thereby to reduce health care costs. In a recent paper (Ravi *et al.*, 2017), Ravi et al. introduce a deep learning approach in data analytics for low-power mobile wearable devices. They discussed the constant and rapid growth in the wearable devices applications and how the segmentation of the raw signals could make machine learning more practical.

1.3.2 Human Activity Recognition

Authors from the journal paper (Bulling, Blanke and Schiele, 2014) talk about how human activity recognition has been a main research interest in the last 20 years and the progress in the development of matured activity recognition systems. Also, the importance of sensor-based human-activity recognition in the field of artificial intelligence and ubiquitous computing has been discussed and presented in (Yin, Yang and Pan, 2008). Authors in this paper have employed physiological signals for activity detection but the focus was on detecting abnormal activities from normal activities. Some other works talk about the usage of physiological monitoring human emotion recognition (Kim, Bang and Kim, 2004).

However, majority of human activity recognition systems employ behavioral data such as triaxial accelerometer signals from inertial sensors, smartwatches and smartphones. Inertial sensors could be combined with other wearable sensors to propose multi modal analysis which has wider applicability in many domains. One such work has been discussed in paper (Bulling, Ward and Gellersen, 2012) where eye and body movements were jointly studied to recognize the reading events.

1.4 Experimental Setup

This thesis focuses on analyzing the data collected over an extended period of time. Figure 1 illustrates the overview of the experimental setup. The data is collected using SHIMMER (Sensing Health with Intelligence, Modularity, Mobility and Experimental Re-usability) ECG Sensors (Mehmood *et al.*, 2016).

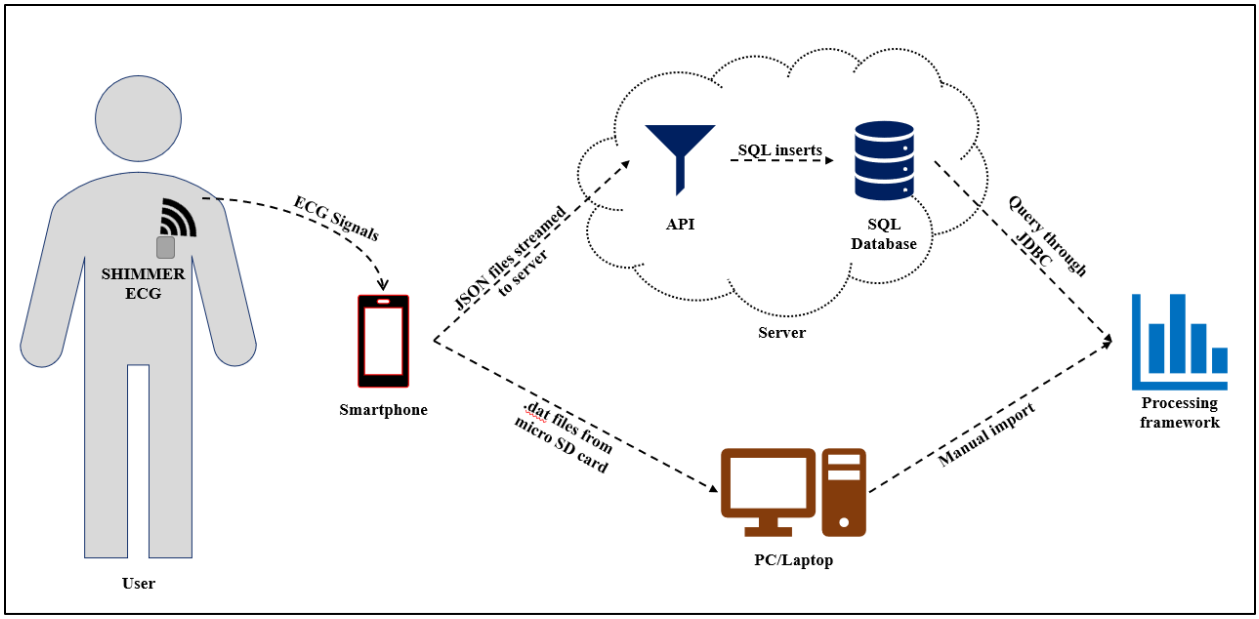

Figure 1: Illustration of Experimental setup

The Shimmer ECG unit is worn on a subject's chest with electrodes attached to different parts of the chest as shown in Figure 2. Shimmer follows the configuration of four-lead setup and it records the pathway of electrical impulses through the heart muscle. A lead refers to the signal of the voltage difference between 2 electrodes. In this work, the collection and storage of data have been demonstrated in 2 ways:

1. An android app called MultiShimmerSync is used to collect the data streaming from Shimmer sensor and stores the data in .dat format which is similar to an excel file. This datafile is stored in Micro SD card of the android device. The data is then manually loaded into the processing framework i.e., MatLab installed in the processing system.
2. There is a custom app built which can send the data to the server in the form of JSON (JavaScript Object Notation) arrays. The data is packaged for every 5 minutes and sent to the server. Custom API (Application Programming Interface) has been built which reads the data packets streaming from the android app, parses it and loads into the SQL server. The database and API are hosted in the same server to provide vicinity to each other thereby avoiding any network delays. This is crucial for continuous loading of the streaming real time data. The API provides additional functionalities such as token based authentication and entity framework for user data. Users can register and login through their credentials and start sending data packets using a token generated for them. So, the

API would automatically assign the correct user id based on the token submitted by the users. This user id would be tagged along all the physiological data before they get loaded into the database. Such continuous data can be retrieved by MatLab by using its inbuilt Database explorer app and a relevant SQL JDBC driver.

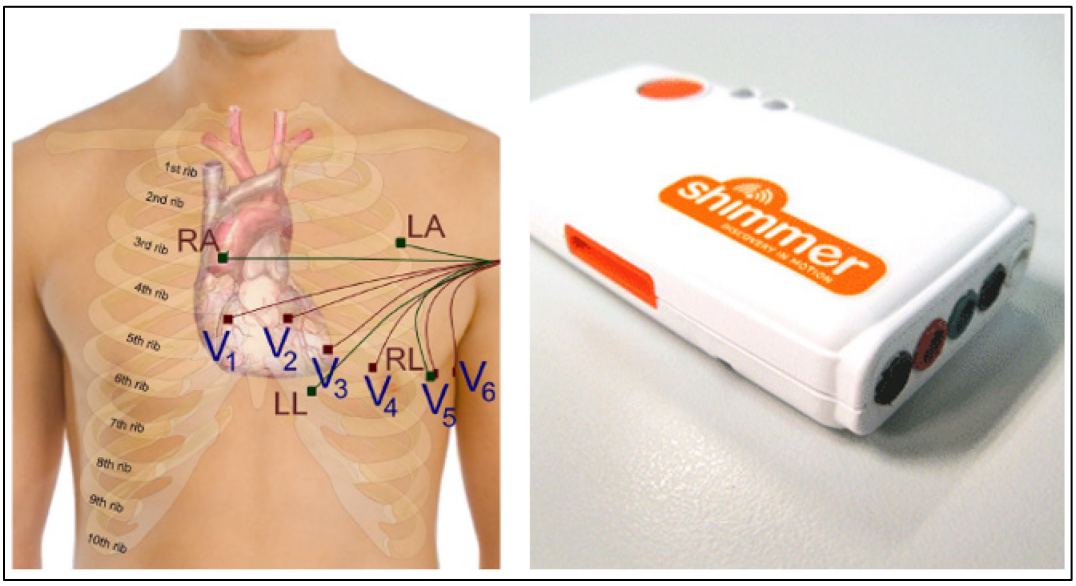

Figure 2: Shimmer ECG electrodes placement

1.5 Processing framework

Figure 3 represents the overview of the processing framework. As represented in the block diagram, there are 6 major steps:

1. Data collection
2. Data segmentation
3. Data filtering and labeling R-R peaks
4. Feature extraction
5. Feature selection
6. Model generation

Each of the modules is discussed in detail in the subsequent sections.

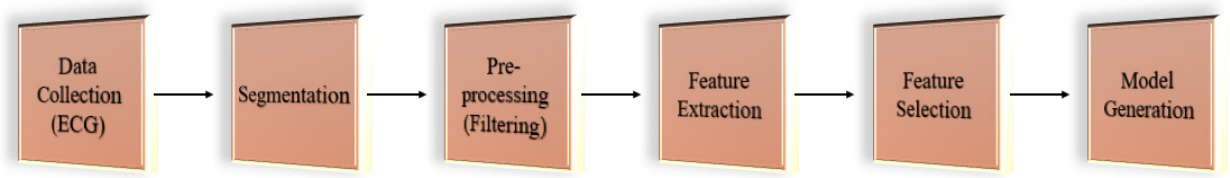

Figure 3: Overview of the data processing steps

1.5.1 Data Collection

The data is collected using Shimmer ECG Sensors (Mehmood *et al.*, 2016). Data acquisition has been conducted in two modes i.e. supervised and unsupervised. Physiological signals have been collected from 10 different subjects and the results are presented in this thesis. The subjects were made to perform a set of tasks during both supervised and unsupervised sessions. During supervised session, all the activities were done in a controlled environment and were labelled accordingly. Whereas in unsupervised session, the data was collected in free-form style which aligns with the daily routine of the subject. The data has been collected at 51.2 Hz sampling frequency. Table 1 lists all the monitored signals used in this study.

Table 1: Acquired ECG signals from the Shimmer device

Signal	Calibration	Unit
ECG LA-RA	CAL	mV
ECG LL-LA	CAL	mV
ECG LL-RA	CAL	mV
ECG Vx-RL	CAL	mV

The signals in the Table 1 represents the four-lead ECG solution configured in the Shimmer device. The leads are described as below. The first 3 leads are bipolar leads which represents the voltage difference between 2 limb electrodes.

- Lead 1 (LA-RA) is the ECG vector signal measured from the RA (right arm) position to the LA (left arm) position.

- Lead II (LL-RA) is the ECG vector signal measured from the RA (right arm) position to the LL (left leg) position
- Lead III (LL-LA) is the ECG vector signal measured from the LA (left arm) position to the LL (left leg) position. This is derived by subtracting Lead I from Lead II.
- Unipolar lead (Vx-RL) is the ECG vector signal measured from the Wilson's Central Terminal (WCT) voltage to the Vx position. The Wilson's Central Terminal (WCT) is a voltage that represents the average potential of the body and acts as a reference point, with respect to which the voltage difference for the unipolar leads is measured. It is calculated by averaging the voltage measured at the RA, LA and LL electrodes. RL electrode has been used to drive the inverted WCT voltage where RL can be placed anywhere on the body as long as it is outside of the triangle formed by the other 3 limb electrodes (LA, RA and LL).

1.5.1.1 Supervised Data Collection

Subjects were required to do five different activities in exclusion with each other under the supervision of the experimenter. Table 2 summarizes the supervised data, which is collected for five activities: sitting, walking, standing, eating and driving a car. Each activity was performed for 10 minutes each and the data files corresponding to each activity were labeled straightaway. These data files represent the source for building baseline solution for the proposed methodologies discussed in subsequent chapters. The details about the activities performed by the subjects are as follows:

- Sitting: Users were made to sit on a chair or a couch. This represents sedentary activity where minimal movement was recorded.
- Standing: Users were made to stand still without much movement.
- Walking: Users were made to walk continuously either indoors or outdoors.
- Eating: Users were made to sit and have a meal or snacks.
- Driving: Users were required to drive a Car.

Table 2: Supervised data summary

SN	Duration (min)	Activity
1	10:00	Sit
2	10:00	Stand
3	10:00	Walk
4	10:00	Eat
5	10:00	Drive a car

1.5.1.2 Unsupervised Data Collection

The unsupervised long span data was collected for 7-8 hours continuously, taking into considerations the battery life of the shimmer device. Table 3 provides a detailed description of one of the sessions of unsupervised data collection. The timestamps presented in the Table 3 are recorded and logged manually by the participant. The activities are performed in sync with daily routine of the subject. However, the subjects were made to manually log the activities during unsupervised session for evaluation purposes. For each activity, the log includes the start time of the activity, name of the activity and end time of the activity recorded by the subjects. Many third-party apps are available to help manual logging, out of which an app called TimeStamper has been employed by the subjects. With free-form data continuously collected for 7-8 hours at 51.2 Hz sampling frequency, the size of the data files generated were turned out to be ~800-900 MB.

Table 3: Unsupervised data logging

Start Time	End Time	Duration (hh:mm:ss)	Activity
9:25:08	9:25:20	0:00:12	walk
9:25:20	9:44:17	0:18:57	sit
9:44:17	9:45:59	0:01:42	walk
9:45:59	10:09:57	0:23:58	sit
10:09:57	10:11:43	0:01:46	walk
10:11:43	10:16:16	0:04:33	stand
10:16:16	10:17:41	0:01:25	walk
10:17:41	10:39:17	0:21:36	drive
10:39:17	10:53:35	0:14:18	walk
10:53:35	10:56:49	0:03:14	stand
10:56:49	10:58:55	0:02:06	walk
10:58:55	11:06:08	0:07:13	drive
11:06:08	11:19:33	0:13:25	walk
11:19:33	11:40:47	0:21:14	eat
11:40:47	12:54:19	1:13:32	sit
12:54:19	13:23:35	0:29:16	walk
13:23:35	13:47:17	0:23:42	stand
13:47:17	13:49:36	0:02:19	walk
13:49:36	14:07:19	0:17:43	sit
14:07:19	14:19:43	0:12:24	walk
14:19:43	14:32:18	0:12:35	stand
14:32:18	14:45:53	0:13:35	eat
14:45:53	14:52:23	0:06:29	sit
14:52:23	14:53:03	0:00:41	walk
14:53:03	14:55:10	0:02:07	sit
14:55:10	15:03:23	0:08:13	stand
15:03:23	15:39:28	0:36:05	sit
15:39:28	15:42:51	0:03:24	walk
15:42:51	15:55:30	0:12:39	eat
15:55:30	16:03:07	0:07:37	sit
16:03:07	16:09:32	0:06:24	walk
16:09:32	16:14:04	0:04:33	stand
16:14:04	16:25:03	0:10:59	eat

1.5.2 Data Segmentation

As suggested in one of the related works (Ravi *et al.*, 2017), the method adopted for data pre-processing was to segment the data signal by overlapping in such a way that each segment (also

called as window) comprises of 200 data points and increments were made for every 50 data points. So, each window had data points collected for 4 seconds and there was 3 seconds overlap in every subsequent window. Then each window was considered a unit of ECG data for the subsequent processing. Figure 4 shows the illustration of the segmentation procedure employed in this work.

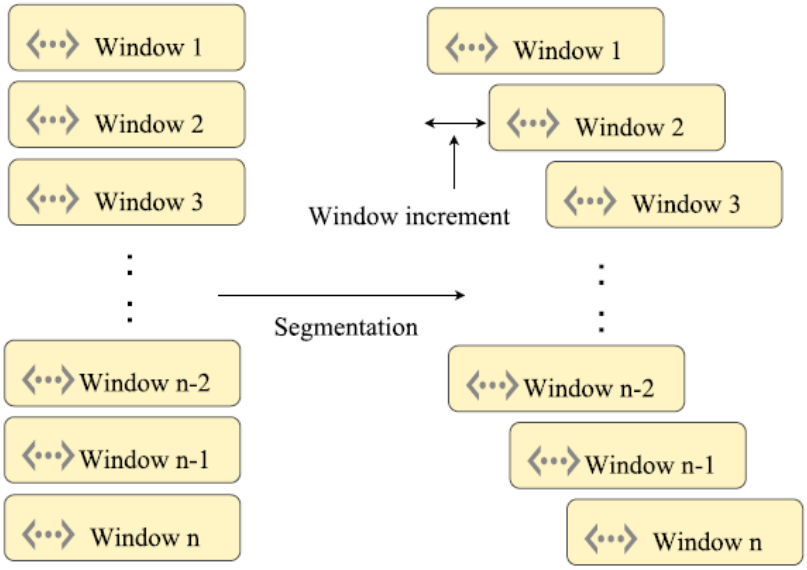

Figure 4: Segmentation procedure

1.5.3 Data filtering

The segments were made to undergo signal processing using low-pass and high-pass filters having cut-off frequencies of 0.5 Hz to 4 Hz. In this way, the power line noise and high frequency noise were removed. Along with the noise removal, other pre-processing tasks were also carried out such as locating local maxima through windowed filtering, and labeling R peaks in the recorded ECG data. This was implemented in MatLab as per the package provided by authors Burns et al. in their comprehensive work (Burns *et al.*, 2010).

1.5.4 Feature Extraction

ECG data features were extracted using MATLAB from the signals and are shown in Table 4. After segmentation of the signal, various time domain and frequency domain features were extracted from each window. Time domain features are mostly based on R-R peaks and are derived by using the preprocessing package written by author Sergey Chernenko. Feature name, unit of measurement and a brief description regarding each feature is given in the Table 4.

Table 4: Extracted ECG features

Domain	Feature	Unit	Description
Time – Domain	HRV	-	Heart Rate Variability
	AvgHR	bpm	Average Heart Rate
	MeanRR	ms	Mean of selected R-R series
	NN50	count	No. of consecutive R-R intervals that differs more than 50 milliseconds
	SD_HR	1/min	Standard Deviation of Heart Rate
	SD_RR	1/min	Standard Deviation of R-R interval
	RMSSD	ms	Root Mean Square of the differences of selected R-R interval series
	SE	-	Sample Entropy
Frequency- Domain	PSE	-	Power Spectral Entropy

The distribution of data points could be visualized using scatter plot. The relationship between Average heart rate and standard deviation of heart rate is shown in the Figure 5.

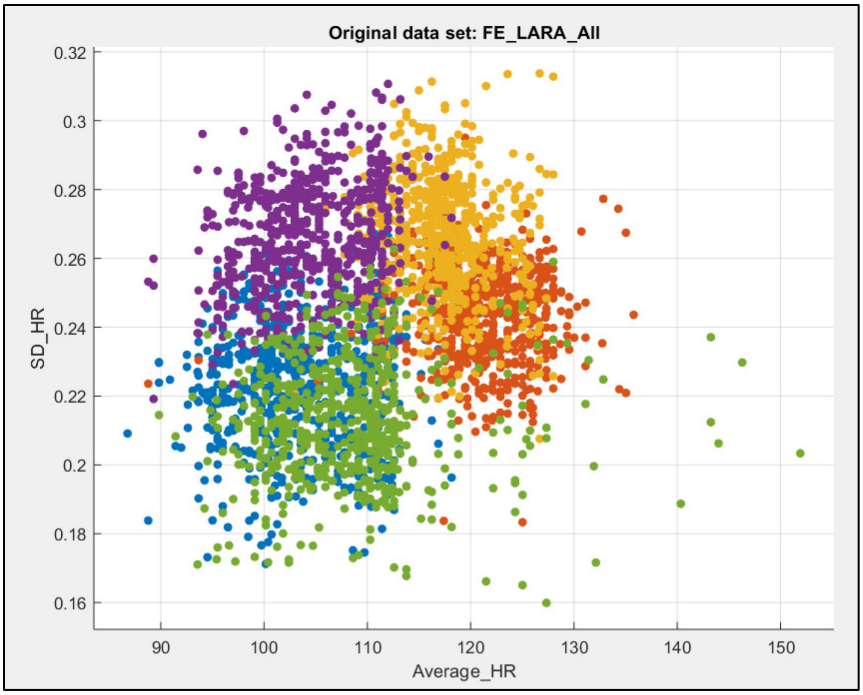

Figure 5: Scatter plot of lead1 data from subject 1

1.5.5 Feature Selection

Feature Selection is the automatic selection of attributes in the data that are most relevant to the current predictive modeling problem. It helps overcome the curse of dimensionality as well as improve the accuracy of the model. Two methods of feature selection have been implemented in this work.

1.5.5.1 Correlation

One of the cleanest method for understanding a feature relation to the target variable is Pearson's correlation coefficient which computes linear correlation between two vectors or variables. The resulting value lies in [-1;1], with -1 indicating perfect negative correlation (as one variable increases, the other decreases), +1 indicating perfect positive correlation and 0 meaning no linear correlation between the two entities.

In this method, correlation of each feature is calculated with label and is stored in a vector. By observing the values of correlation coefficients, an absolute threshold value of 0.2 was set

and only those features which had correlation coefficient value above the threshold were considered. If we have 2 datasets, the correlation coefficient between two variables can be determined by equation (1) given below.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (1)$$

where:

n is the sample size

x_i, y_i are the single samples indexed with i

$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean for one dataset $\{x_1,\ldots,x_n\}$

$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the sample mean for another dataset $\{y_1,\ldots,y_n\}$

1.5.5.2 SequentialFS:

SequentialFS is a wrapper method of Feature selection where it selects features sequentially and trains an algorithm using cross validation. It repeats this process until it has good subset of features which performed best during learning phase. The algorithm used in this wrapper method is Linear Discriminant Analysis(LDA) which works well with SequentialFS and 10-fold cross-validation has been used. The output is a logical vector indicating which features are finally chosen.

1.5.6 Model Generation

This is the core step in the process of learning from the data. Various Machine learning solutions has been studied and applied in this thesis. Since features are extracted from 3 different bipolar leads, the models have been built using each of those leads and from combination of features from multiple leads. More information about the methodologies are discussed in subsequent chapters.

Chapter 2: Proposed Methodology

2.1 System Architecture

The work undertaken in this thesis follows 3 methodologies out of which first one is conventional machine learning techniques. The challenges and drawbacks associated with this method have been discussed in this chapter. The other 2 methodologies – incremental learning and semi supervised learning techniques represent the solutions which have been discussed in subsequent chapters. Figure 6 depicts the block diagram of the whole architecture.

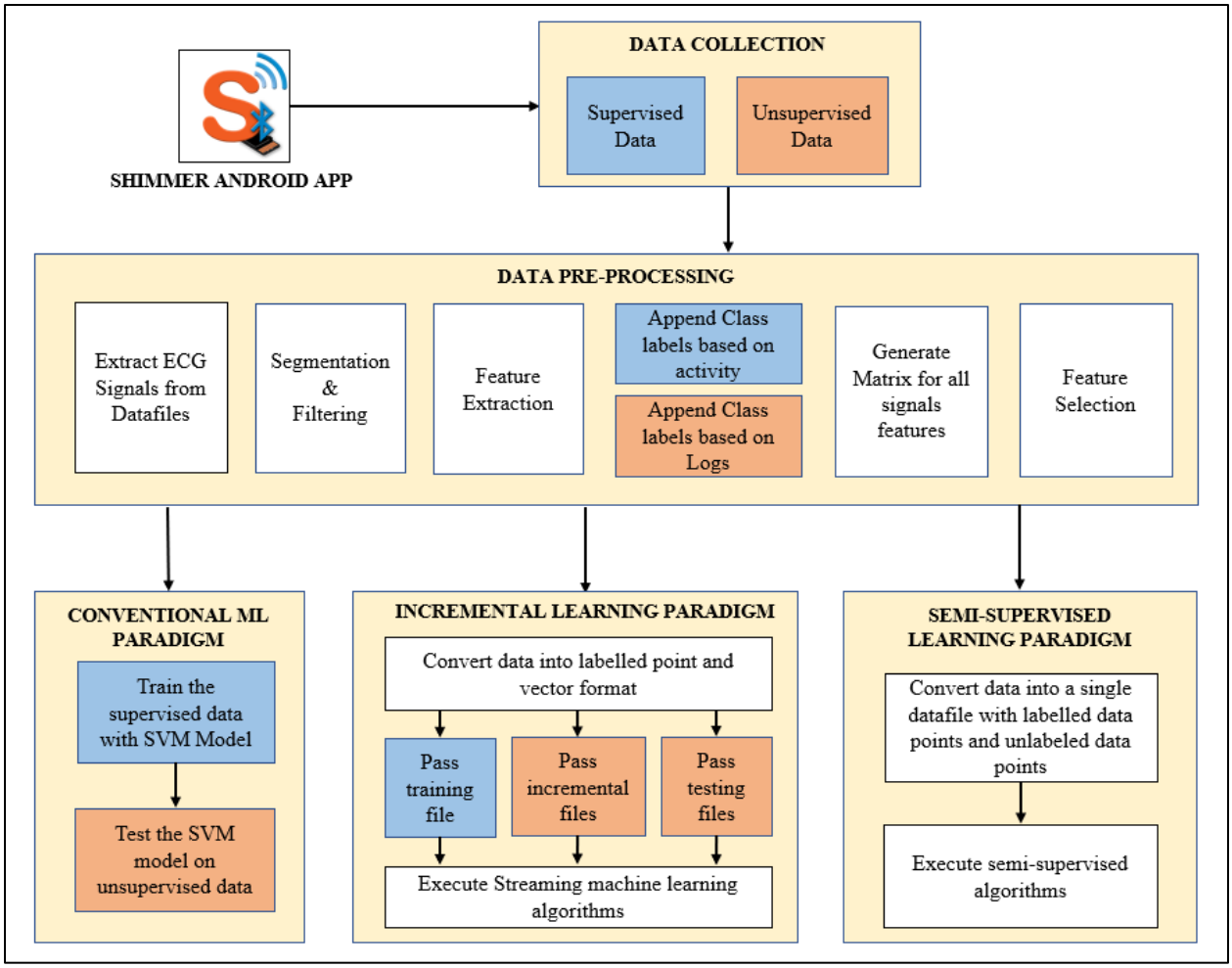

Figure 6: Block diagram of the system architecture

2.2 Conventional machine learning

In previous chapter, all the steps in data pre-processing were covered. This section is focused on model generation part. Conventional Machine learning paradigm considers data in 2 phases – training phase and testing phase. So, supervised data has been used in training phase whereas unsupervised data has been used in testing phase. Logs manually logged by subjects were used to label the unsupervised data. Linear SVM has been used to build baseline models in MatLab. The models were generated on supervised data (during training phase) and used to predict labels on unsupervised data (during testing phase). This experiment gave a chance to see how the conventional algorithms work on data from different sessions.

Linear SVM offer an effective classification strategy to separate input vector into a 2-class problem by evaluating a score on the input vector. The score is represented in terms of a scalar function f(x) and is calculated as shown in (2).

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b \qquad (2)$$

where,

x_i represent the support vectors,

N is the number of support vectors

$a_i > 0$ are adjustable weights

y_i can be either -1 or +1

b is the bias term

and $K(x_i, x)$ is the kernel function.

Binary SVM has been extended to multi class classifier which uses built in MatLab function *fitcecoc.* Three different leads of ECG data measurements raised the opportunity to create multiple models and study the differences between them. Three models were built using the features extracted from each of the bipolar leads of ECG. Three other models were built by combining the features from multiple leads. Table 5 lists the models created in this study. The models were built using 10-fold cross validation method (10-CV). In 10-CV, the original dataset is partitioned into 10 equal size subsets. Of the 10 subsets, a single subset is retained as the validation data for testing the model, and the remaining nine subsamples are used as training data. The cross-validation process is then repeated 10 times (the folds), with each of the 10 subsets used exactly once as the validation data. The 10 results from the folds can then be averaged to produce a single estimation. The advantage of this method is that all observations are used for validation in exactly one out of the 10 iterations without being participated in the training process for that iteration.

Table 5: Evaluated models

Models	Description	No. of Features
L1	Lead1	9
L2	Lead2	9
L3	Lead3	9
L1 + L2	Lead1 + Lead2	18
L1 + L3	Lead1 + Lead3	18
L1 + L2 + L3	Lead1 + Lead2 + Lead3	27

2.3 Results

For each of these models, 10-fold cross validation was applied to obtain cross validated models. The accuracies of all the six models are shown in Table 6. These models were tested on the unsupervised data files which followed the same preprocessing steps as supervised data files and the prediction results are shown in Table 7.

Table 6: 10-CV Accuracies on supervised data

Models	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10
L1	89.8%	98.4%	86.1%	94.0%	80.2%	92.0%	83.7%	86.3%	84.8%	98.9%
L2	92.7%	91.4%	92.5%	79.7%	88.2%	75.8%	89.3%	95.3%	82.1%	97.8%
L3	91.4%	90.4%	98.8%	84.1%	100.0%	83.1%	86.9%	97.4%	88.0%	78.6%
L1 + L2	94.4%	99.4%	93.1%	98.6%	97.2%	97.8%	96.9%	97.4%	90.0%	99.3%
L1 + L3	96.0%	100.0%	99.3%	99.4%	100.0%	97.0%	98.3%	99.8%	93.3%	99.5%
L1 + L2 + L3	97.9%	100.0%	100.0%	99.2%	100.0%	98.7%	98.5%	100.0%	95.5%	99.6%

Table 7: Prediction accuracies on unsupervised data

Models	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5	Subject 6	Subject 7	Subject 8	Subject 9	Subject 10
L1	21.2%	51.0%	23.5%	47.9%	53.5%	52.7%	12.9%	20.1%	20.6%	12.4%
L2	18.0%	47.7%	0.1%	68.7%	39.3%	42.8%	35.0%	19.0%	50.6%	18.3%
L3	16.6%	10.1%	11.0%	57.1%	27.1%	22.2%	22.2%	54.1%	10.5%	24.5%
L1 + L2	23.3%	64.2%	6.7%	51.3%	37.6%	41.8%	27.6%	36.6%	20.1%	12.7%
L1 + L3	35.5%	39.2%	6.5%	56.2%	23.0%	8.7%	32.0%	41.3%	10.6%	8.3%
L1 + L2 + L3	31.0%	39.1%	12.7%	59.1%	22.9%	8.9%	28.9%	41.7%	10.6%	11.9%

2.4 Conclusion

Support Vector Machines have failed to apply the learned model to predict the labels on unsupervised data with good accuracy rate. This gives an idea about the drawbacks of conventional machine learning algorithms while dealing with the data that was never seen before. Though there is a plenty of opportunity to refine the models, by optimizing the parameters or by feature selection, there is still not much improvement in the accuracy rate because of the variabilities between sessions. Subsequent chapters talk about the methodologies and techniques that can be applied to overcome this problem.

Chapter 3: Incremental learning

3.1 Introduction

Incremental learning has recently attracted growing attention from both academia and industry. Incremental learning is a machine learning paradigm where the learning process takes place whenever new examples emerge and adjusts what has been learned according to the new examples. From the computational intelligence point of view, there are at least two main reasons why incremental learning is important. First, from data mining perspective, many of today's data-intensive computing applications require the learning algorithm to be capable of incremental learning from large-scale dynamic stream data, and to build up the knowledge base over time to benefit future learning and decision-making process. Second, from the machine intelligence perspective, biological intelligent systems are able to learn information incrementally throughout their lifetimes, accumulate experience, develop associations, and coordinate sensory-motor pathways to accomplish goals.

In this thesis, analyzing the data collected over long period of time has been the main focus. To overcome the drawbacks of the conventional machine learning techniques, incremental learning provides a needed solution from streaming analytics. In this chapter, a hypothesis to construct an incremental learning module using streaming logistic regression and streaming k means algorithms has been developed and supported through empirically generated models.

3.2 Related Work

In (He *et al.*, 2011) He et al. put forward a universal adaptive incremental learning framework called ADAIN which is said to learn from continuous data and enhance trained model and its prediction performance with time. The input data is considered as raw data over which a baseline model and hypothesis was developed. Some of the previous works show modification of existing

algorithms to support incremental learning. One such paper (Polikar *et al.*, 2001) discusses the proposed algorithm called Learn++ which uses Neural Networks pattern classifiers for incremental training.

There have been some previous efforts on clustering of data streams. The exploration of the data stream over different time windows can provide the users with a much deeper understanding of the evolving behavior of the clusters (Aggarwal *et al.*, 2003). The authors in this work (Aggarwal *et al.*, 2003) have proposed an idea of using two components, one online component to store detailed summary statistics periodically and another off-line component which uses these statistical data to provide a quick understanding of the clusters in the data streams. In a similar work (Chen and Tu, 2007) the authors have proposed density approach to cluster the data streams in real time by studying relationship between data density, decay factor and cluster structure.

In many applications, the value of data decreases in proportion to the time that has passed since the data is produced (Maarala *et al.*, 2015). For example, in traffic scenarios, a delay counted in minutes is too long when decisions about the quickest driving routes in metropolitan areas have to be made immediately. In (Maarala *et al.*, 2015), the authors discuss that Apache Spark Streaming is an efficient method as the core component for real time analysis because it is efficient in iterative computing tasks, supports a variety of data sources and programming languages, and can be run on Hadoop, which is significant for Big Data processing.

3.3 Platform

Apache Spark is an open source big data processing framework built with sophisticated analytics. It provides a comprehensive framework to manage big data processing requisites with a diverse range of data sets. Spark supports many applications through its variety of components among which Spark's Machine learning component and Streaming component are of our interests. MLlib is Spark's machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. It consists of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction, and so on.

Spark streaming component receives the data streams and divides the data into batches which are then sent to spark engine for further processing.

3.4 Methodology

3.4.1 Preprocessing

For our analysis, we have used streaming logistic regression and streaming k means clustering algorithms to build an incremental learning module in Spark. Streaming logistic regression is intended to work only on binary classification. So, we had to build 5 models with One vs Rest criterion.

The streaming machine learning algorithms in Spark accept the data in labeled point format as shown in expression (3).

$$(\text{Class; [Feature1; Feature2; Feature3; …….]}) \quad (3)$$

Also, the Streaming k-means clustering algorithm requires training files in vector format which is shown in expression (4).

$$[\text{Feature1; Feature2; Feature3; …….}] \quad (4)$$

So, the data files were converted into the formats required by Spark before loading them to streaming algorithms.

3.4.2 Streaming Model generation

Streaming logistic regression creates a baseline model and updates it by learning from the new training files being streamed to the algorithm. Datasets from one lead (Lead 1) were considered for streaming logistic regression and 4 sessions of incremental training has been carried out for all the subjects.

Additionally, as clustering provides the best solution for unsupervised learning, Streaming k-means clustering has been used as part of the analysis. Streaming k-means clustering enhances

the functionality of k-means algorithm by learning from the streaming data continuously and updating the cluster centers during each interval as defined by an input parameter called batch duration. In this work, the effect of streaming data files on the clustering algorithm was studied by predicting the clusters for new data points, analyzing the shift in size of each cluster and analyzing the variation in the cluster centers. For each batch of data, all the data points were assigned to their nearest cluster, new cluster centers were computed by using equation (5).

$$c_{t+1} = \frac{c_t n_t \alpha + x_t m_t}{n_t \alpha + m_t} \quad (5)$$

where,

c_t represents old cluster center,

n_t represents number of data points present in the cluster thus far

α represents the decay factor which is the key element in incremental learning

x_t is the new cluster center from the current batch

m_t is the number of data points added to cluster in the current batch

3.5 Results

3.5.1 Streaming logistic regression

To build models for binary classification using streaming logistic regression, we converted the labels in our data files so that they can be either 0 or 1. Since, we have 5 different activities, we built 5 models with one vs rest criterion for each subject. For each of these models, we have prepared 4 training data files (1 baseline datafile + 3 incremental files) and a single test file. The baseline file has been created from Supervised data and the incremental files have been created from unsupervised data. Once the streaming context with logistic regression is started, the incremental files and test file are passed to streaming directories and accuracies are calculated.

Figure 7 shows the snapshot for one of the models designed to distinguish sitting activity from rest of the activities. It indicates the results shown after a baseline model is trained. The classes are distinguished as 1 and 0 where sit activity is labeled as 1.0 and rest of the activities

comprising stand, walk, eat and drive are labeled as 0.0. The output of streaming algorithm is shown in terms of (Actual class label, Predicted class label). Figure 8 shows the improved results after an incremental file is passed to the trained model. It can be observed that some of the data samples were correctly classified as 1.0(sit activity) after incremental file is passed.

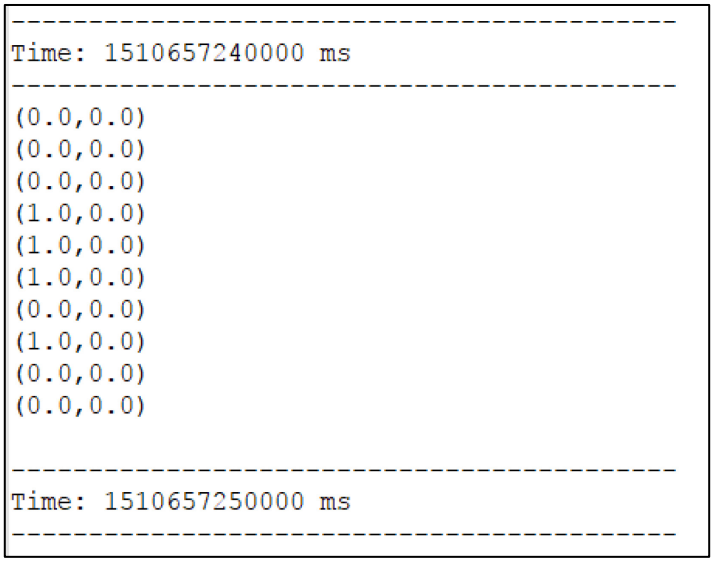

Figure 7: Results of a baseline logistic regression model at a timeframe

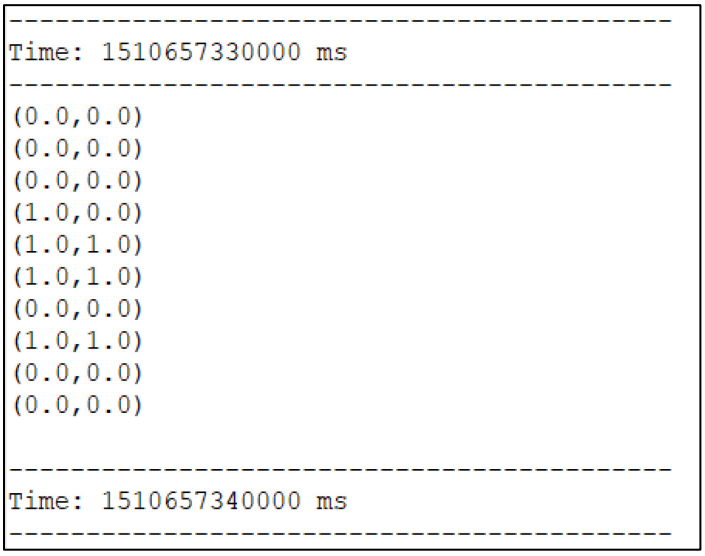

Figure 8: Results of the logistic regression model after incremental learning at a timeframe

The analysis using Streaming logistic regression has been extended to all subjects. Table 8 indicates the prediction accuracies on test files during 4 sessions of incremental learning. Each

subject's results are shown in terms of average of all binary classifier (one vs rest model) results. Session 1 indicates the results of a baseline model trained from supervised data. Sessions 2, 3 and 4 indicate results after new data files were passed to the streaming algorithm to update the model. It is evident from the Table 8 that the models were getting better and were able to predict the classes with higher accuracies during subsequent sessions of incremental learning.

Table 8: Accuracies of streaming logistic regression during incremental learning

Subjects	**Session 1**	**Session 2**	**Session 3**	**Session 4**
Subject 1	55%	65%	74%	76%
Subject 2	71%	76%	80%	87%
Subject 3	74%	72%	81%	84%
Subject 4	84%	86%	86%	90%
Subject 5	74%	71%	76%	84%
Subject 6	74%	65%	74%	76%
Subject 7	66%	74%	84%	83%
Subject 8	80%	82%	84%	87%
Subject 9	82%	79%	79%	85%
Subject 10	75%	77%	81%	82%
Average	73%	75%	80%	83%

3.5.2 Streaming k-means clustering

The streaming k-means clustering has been implemented to group the data points among different clusters incrementally. Once each incremental file is passed, the cluster labels are predicted and printed to a result file. Spark starts labeling the clusters from 0. Hence, we have cluster labels 0,1,2,3 and 4 for five activities. The number of data points assigned to each cluster are noted and consequently, the shift in number of data points from one cluster to another are analyzed. Figure 9 shows the distribution of data points in each cluster as and when different incremental files were passed to the streaming algorithm. The change in cluster counts has been analyzed over 5 sessions. There is a possibility of having low inter cluster distances between few activities which could be the reason for having less data points in few clusters like clusters 1 and

3. Irrespective of that, the model is being continually updated from the incremental files which is evident from the shifting of the data points from clusters 0 and 2 to cluster 4.

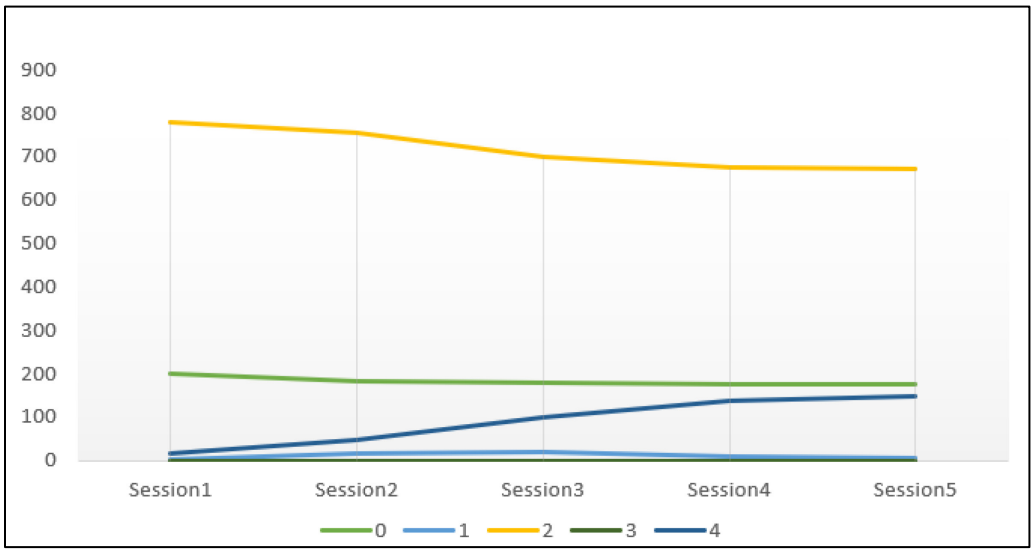

Figure 9: Trend in the change of number of data points in each cluster as a result of incremental learning

Additionally, work has been extended in Apache Spark to read the cluster centers of the 5 clusters produced using streaming k-means algorithm. Figures 10-13 demonstrate the change of cluster centers as and when new data files were added to streaming algorithm. Cluster centers were extracted from the trained model and were written to a file. Since streaming algorithms are being dealt with, the algorithm writes the outputs in multiple files as per the batch duration which was set to 15 seconds. The cluster centers from the output files were exported to MatLab and were analyzed by drawing scatter plots.

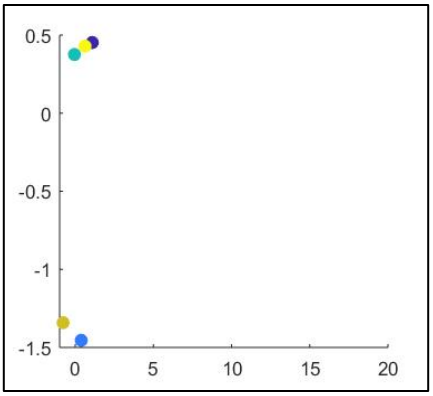

Figure 10: Scatter plots of cluster center during baseline model

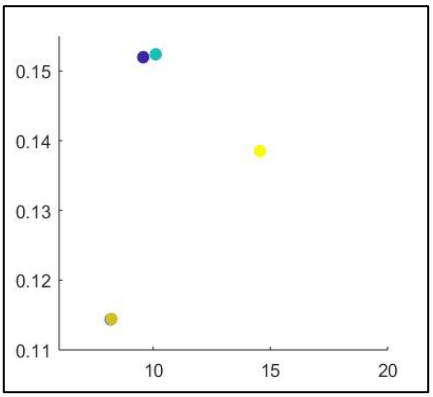

Figure 11: Cluster centers during incremental learning (1)

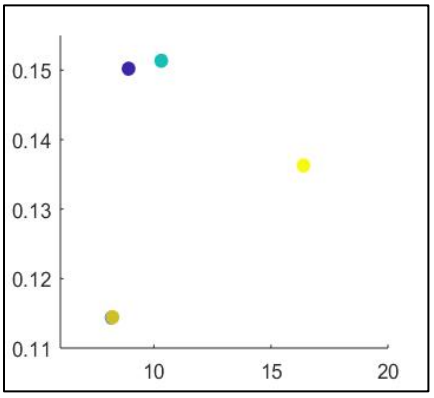

Figure 12: Cluster centers during incremental learning (2)

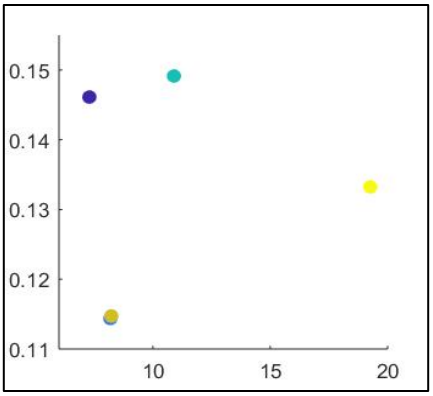

Figure 13: Cluster centers during incremental learning (3)

The scatter plots of cluster centers have been shown in only 2-dimensions for representational purpose. The first plot shown in Figure 10 indicates random cluster assignments at the beginning of the algorithm. So, the cluster centers were set around the values of 0 randomly. The second plot shown in Figure 11 indicates changes in cluster centers after k-means algorithm starts clustering the new data points from the first data file. As and when new data files were added, these cluster centers appear to change as shown in Figure 12. It can be observed that some of the clusters are closely grouped because of some of the identical physiological features between few activities. However, we can see a small separation between 2 very closely located cluster centers in the last plot shown in Figure 13 which we couldn't see in previous 2 plots. But the other two cluster centers shown in variants of blue colors show better separation when the model was getting updated continuously in the process of incremental learning. If the plots could be visualized in multi-dimensions we might be able to see the well separated cluster centers between all the activities.

3.6 Conclusion

In this chapter, different aspects of an incremental learning of a model have been introduced in order to identify a set of human states physiologically using the participants ECG signals. ECG based time and frequency domain features have been used in Spark platform to do streaming analytics and results have been presented. Streaming logistic regression have been used to do the task of online learning which adapts the model continuously by learning from streaming data. The spark platform provides a flexibility to read data at time intervals as defined by the user

efficiently. This is an added advantage as the system will keep tapping for any incoming data and as soon as data is available, the computations would initiate. Various observations have been made on the datasets from 10 subjects which showed the trend towards more accurate classification using incremental method. The addition of more incremental data files would help achieve optimal results.

Chapter 4: Semi Supervised Learning

4.1 Introduction

Semi Supervised Learning is a classical machine learning setup that assumes only limited samples of labelled data for training purposes. This methodology is becoming widely popular as the researchers want better performance for less cost and unlabeled data is cheaper compared to labelled data. The continuous streaming data under the study could be a real nudnik to be labelled. Also, labelling such large streams of data would need experts' advice along with the subject's knowledge. This led to the raise of a paradigm called Semi-supervised learning where both labelled and unlabeled data were used to build better learners, than using each alone. Ideally both labelled and unlabeled data would be combined together in a single feature matrix and semi-supervised algorithms learn from the gaussian distribution and decision boundary shift.

The general principle underlying semi-supervised learning is that the marginal distribution, which can be estimated from the data alone, may suggest an appropriate way to adjust and adapt the target function. The structure of such marginal distributions can be assumed based on the closeness of the data points to each other (continuity assumption), or formation of discrete clusters (cluster assumption), or the location of underlying data points on a manifold of much lower dimension (manifold assumption). Transductive inference has been a common method in semi supervised learning that is focused on learning the unlabeled data and focused on deducing the right labels for them (Sindhwani, Niyogi and Belkin, 2005). This work discusses the implementation of LapSVM (Laplacian Support Vector Machines) algorithm along with a semi-supervised dimensionality reduction technique which have performed well as state of the art methods for Semi supervised learning.

4.2 Related Work

Various methods have been designed and implemented for Semi supervised learning and transductive learning tasks. Authors in paper (Olivier, Schölkopf and Zien, 2006) have showed that how LapSVM has proven to improve the accuracy while detecting the diabetes prediction as compared to the fully supervised version of the classifier. Another work (Li, Du and Zhang, 2012) discusses the detection of abnormalities in ECG in the aim to identify the irregular cardiac activities by using the transductive transfer learning framework. Various transductive and cluster transfer learnings have been studied and compared. In (Sigdel *et al.*, 2014), Sigdel et al, have evaluated the performance of two wrapper methods for semi supervised learning algorithms for classification of protein crystallization images. Self-training algorithms were used as a baseline and compared with another method called Yet Another Two Stage Idea (YATSI) semi supervised learning.

Semi supervised classification algorithms can be combined with some of the state of the art semi supervised dimensionality reduction techniques. One such work has been proposed to diagnose bearing faults in induction motors (Razavi-Far *et al.*, 2017). Different semi supervised dimensionality reduction techniques have been applied and combined with semi supervised classification algorithms and different combinations were evaluated.

Similar to semi supervised learning, there are other directions to build generalized models that complement the goals of semi supervised learning. Related works on such new techniques have been studied and discussed in (Luo *et al.*, 2017) where the authors have studied how the reusable and general purpose adaptation models can be enabled to quickly learn future tasks without much human intervention.

4.3 Methodologies

4.3.1 Semi-Supervised Discriminant Analysis(SDA)

Semi supervised Discriminant Analysis(SDA) is a dimensionality reduction technique which extracts the features from higher dimensional data and make use of both labeled and unlabeled

data points. SDA could be considered as an extended version of Linear Discriminant Analysis(LDA) where the class discriminatory function is preserved as much as possible. LDA uses a linear projection matrix which maximizes the inter-class covariance and minimizes the within-class covariance simultaneously. However, it would be improbable that the covariance matrix could be estimated accurately for each class if there are no sufficient training samples. So, SDA has been proposed by Deng et al. (Cai, He and Han, 2007) which uses the intrinsic features of LDA in finding optimum projections based on class separability and extends it by learning the geometric structure of data from unlabeled data points. Specifically, the labeled data points along with unlabeled data points are jointly used to build a graph incorporating locality information of the data set. The graph provides a discrete estimation to the local geometry of the data manifold. SDA also extends a regularization term, which is normally used to avoid overfitting in regularization algorithms, to incorporate the data manifold structure. Finally, the eigenvector is computed based on the non-zero eigenvalues and used to determine the extracted features in lower dimensional space.

4.3.2 Semi-supervised Classifier – LapSVMs

In machine learning, Manifold regularization is a technique for using the shape of a dataset to constrain the functions that should be learned on that dataset. In many machine learning problems, the data to be learned will be of marginal distribution and do not cover the entire input space. The technique of manifold learning assumes that the relevant subset of data comes from a manifold, a lower dimensional mathematical structure with useful properties. The technique also assumes that there is a good separation between data with different labels so that the function to be learned is smooth, and so the labeling function should not change quickly in areas where there are likely to be many data points. Manifold regularization algorithms can extend supervised learning algorithms in semi-supervised learning and transductive learning settings, where unlabeled data are available. Manifold regularization is a type of regularization which ensures that a problem is well-posed by penalizing complex solutions. Manifold regularization adds a second regularization term, the ‘intrinsic regularizer’, to the ‘ambient regularizer’ used in standard regularization methods which are aimed at overcoming the problem of overfitting.

Laplacian Support Vector Machines (LapSVMs) are the natural extensions to Support Vector Machines and have performed well as a state of the art method of semi supervised learning. LapSVMs follow the principles behind manifold regularization where the loss function is the linear hinge loss which is mainly used for Support Vector Machines. The loss function also has a smoothening factor which is the weight of the norm of the function in the low dimensional manifold (or intrinsic norm), that enforces smoothness along the sample.

Primal problems (not limited to the linear case) are best useful in the case of LapSVM. Melacci et, al have proposed two methods for solving the primal LapSVM problem: Newton's method and Preconjugate Gradient Descent(Melacci and Belkin, 2011). There are two primary reasons why such a solution may be preferable. First, it allows us to efficiently solve a single problem without the need of a two-step solution. Second, it allows us to very quickly compute good approximate solutions, while the exact relation between approximate solutions of the dual and original problems may be involved.

4.3.2.1 Newton's Method:

Solving the primal problem using the Newton's method has the same complexity of the original LapSVM. The only benefit of solving the primal problem with Newton's method relies on the compact and simple formulation that does not requires the "two step" approach and a quadratic SVM solver. Newton's method appears a natural choice for an efficient minimization, since it builds a quadratic approximation of the function which thus makes the function differentiable.

4.3.2.2 Early stopped Preconditioned Conjugate Gradient (PCG):

Instead of performing a costly Newton's step, the solution of the system can be computed by conjugate gradient descent. In detail, when the decision function becomes quite stable between consecutive iterations or when the error rate on is not decreasing anymore, then the PCG algorithm should be stopped. Due to their heuristic nature, it is generally better to compare the predictions every few iterations and within a certain tolerance. In this method of training the classifier, various parameters have been included such as:

MaxIter- Maximum number of iterations = 2000

CGstopType – The stopping criterion for CG iterations is ‘Stability stop’
CGStopParam – The parameter for the selected CG stop type which would be the percentage of tolerated different decisions between two consecutive checks = 1.5%
CGStopIter = 3 (which checks for stability after every three iterations)

Due to the high amount of unlabeled training points in the semi–supervised learning framework, the stability of the decision, can be used as a reference to early stop the gradient descent (stability check). By implementing this algorithm, the error rates are considerably reduced, and the time required to train the classifier are drastically improved.

4.4 Results

The semi-supervised learning is implemented on the LapSVMs algorithms ‘Newton’s method’ and ‘Early stopped PCG’ which show a considerable difference in the performance. Training by PCG with the proposed early stopping conditions shows an appreciable reduction of the training time and the error rate on all datasets. Three models for each subject have been considered for the study. Since LapSVM is a binary classifier, transformation techniques were applied to convert into a multiclass classifier using one vs rest strategy. It can be observed that PCG method has edged Newton method by providing better prediction accuracies on unlabeled data at around 86% and also at minimal training time. The accuracies and training times for each model before employing semi-supervised discriminant analysis are shown in Table 9.

The employment of Semi supervised discriminant analysis has transformed the original feature space into 5-dimensional feature space. However, the complex geographical separations of unlabeled data points have resulted in better accuracies in higher dimensional space (before SDA) than on transformed lower dimensional space (after SDA). The results for LapSVM on reduced dimensionality set have been shown in Table 10.

Table 9: LapSVM accuracies and training times before SDA

Subject	Model	Newton Method		Early Stopped PCG method	
		Accuracy	Training time (secs)	Accuracy	Training time (secs)
Subject1	L1	77.2%	2.72	86.0%	0.23
	L1 + L2	76.3%	2.20	85.9%	0.22
	L1 + L2 + L3	76.8%	1.85	86.5%	0.22
Subject2	L1	89.3%	21.06	86.7%	0.21
	L1 + L2	89.1%	4.17	86.5%	0.22
	L1 + L2 + L3	86.8%	1.76	87.0%	0.19
Subject3	L1	83.7%	23.52	87.5%	0.16
	L1 + L2	80.5%	1.11	88.0%	0.14
	L1 + L2 + L3	86.8%	1.02	88.9%	0.14
Subject4	L1	88.2%	2.83	90.1%	0.20
	L1 + L2	90.9%	1.10	91.3%	0.21
	L1 + L2 + L3	93.2%	0.90	91.6%	0.17
Subject5	L1	86.4%	9.69	84.5%	0.13
	L1 + L2	86.8%	4.45	86.0%	0.13
	L1 + L2 + L3	82.5%	1.00	82.7%	0.18
Subject6	L1	83.6%	8.48	86.6%	0.15
	L1 + L2	80.5%	1.66	86.4%	0.15
	L1 + L2 + L3	80.1%	1.45	86.2%	0.19
Subject7	L1	79.0%	24.66	80.8%	0.44
	L1 + L2	79.2%	2.96	85.7%	0.19
	L1 + L2 + L3	80.1%	1.75	85.9%	0.20
Subject8	L1	78.8%	36.18	86.2%	0.23
	L1 + L2	78.8%	15.56	86.4%	0.29
	L1 + L2 + L3	78.4%	2.96	86.6%	0.24
Subject9	L1	84.8%	11.71	86.0%	0.19
	L1 + L2	87.2%	3.22	84.9%	0.26
	L1 + L2 + L3	76.7%	1.95	87.0%	0.19
Subject10	L1	79.4%	55.79	84.8%	0.43
	L1 + L2	79.0%	3.91	85.1%	0.31
	L1 + L2 + L3	82.4%	3.55	85.2%	0.32
Average		**82.8%**	**8.51**	**86.4%**	**0.22**

Table 10: LapSVM accuracies and training times after SDA

Subject	Model	Newton Method		Early Stopped PCG method	
		Accuracy	Training time (secs)	Accuracy	Training time (secs)
Subject1	L1	75.0%	145.99	74.3%	0.82
	L1 + L2	78.2%	122.50	71.5%	0.86
	L1 + L2 + L3	34.2%	132.02	33.8%	0.60
Subject2	L1	72.8%	115.28	76.4%	0.34
	L1 + L2	86.9%	91.92	77.2%	0.25
	L1 + L2 + L3	77.4%	113.41	65.1%	0.24
Subject3	L1	80.3%	53.44	81.0%	0.19
	L1 + L2	77.9%	73.89	79.6%	0.31
	L1 + L2 + L3	81.0%	67.14	75.9%	0.59
Subject4	L1	87.3%	53.67	86.6%	0.58
	L1 + L2	89.0%	45.19	89.4%	0.42
	L1 + L2 + L3	90.7%	45.77	93.5%	0.22
Subject5	L1	84.6%	47.96	79.5%	0.28
	L1 + L2	82.4%	37.22	80.7%	0.49
	L1 + L2 + L3	82.5%	55.86	82.6%	0.29
Subject6	L1	82.2%	73.17	81.6%	1.07
	L1 + L2	85.4%	67.45	81.8%	0.42
	L1 + L2 + L3	76.3%	65.03	77.2%	0.39
Subject7	L1	77.8%	160.20	82.6%	0.54
	L1 + L2	75.8%	145.62	75.9%	0.56
	L1 + L2 + L3	80.3%	125.04	79.7%	0.75
Subject8	L1	75.5%	139.30	73.9%	1.05
	L1 + L2	77.4%	135.93	77.2%	0.43
	L1 + L2 + L3	78.0%	127.61	76.9%	1.16
Subject9	L1	83.4%	111.52	76.9%	0.64
	L1 + L2	84.5%	124.96	83.9%	0.73
	L1 + L2 + L3	77.7%	112.85	77.1%	0.64
Subject10	L1	76.3%	1174.68	76.4%	2.32
	L1 + L2	74.7%	238.17	75.1%	0.92
	L1 + L2 + L3	77.1%	316.67	64.1%	1.33
Average		**78.7%**	**143.98**	**76.9%**	**0.65**

4.5 Comparison and Conclusion

In this thesis different state of the art methodologies have been applied to classify different activities on large amount of continuous data. Since the data was collected in two different

modes: supervised and continuous, the baseline model failed to predict the labels on continuous data with good accuracies. So, the methods applied using incremental learning and Semi supervised learning have provided better results and overcome many challenges in machine learning such as session to session variability, regularization and continuous adaptability. The comparison between all the solutions for Lead 1 ECG data have been shown in Figure 14.

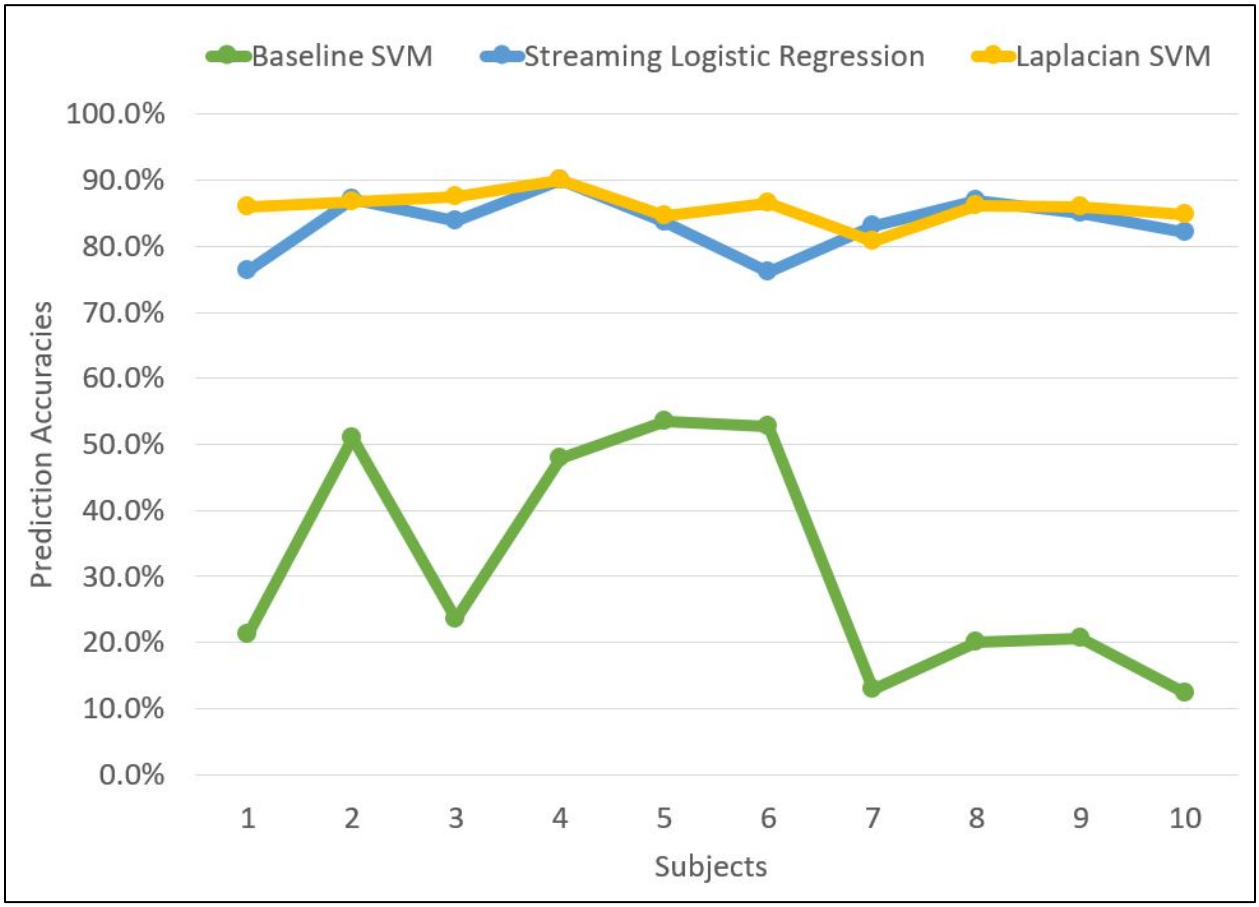

Figure 14: Comparison chart of different machine learning solutions applied in this thesis

It can be observed that both Incremental learning using Streaming logistic regression and Semi supervised learning using Laplacian SVM have yielded good results overcoming all the challenges discussed earlier. Though, the LapSVM trained on original feature set using early stopping PCG method has fared better than the incremental learning results. Semi-supervised analysis has shown that more customization on the optimization and regularization parameters can help us achieve even better results.

4.6 Future Scope

As future work, additional technologies that complement online learning and semi supervised learning such as domain adaptation methods, transfer learning methods, zero shot learning, multi task learning etc. can be implemented and compared. The development of these state of the art methods can deal with many hardest challenges in Machine learning.

The machine learning models which can predict user's state in real time can be incorporated in many applications in the various domains. Other physiological measures such as GSR (Galvanic Skin Response) and behavioral data such as motion could be included to support multi modal analysis. Many solutions can be built than can address the development of Smart homes, Smart cities, Smart mobility etc. One such project has already kicked off with University of Michigan Dearborn - Ford Motor Company Alliance, where the project deals with the usage of IOT technologies and Machine learning technologies implemented in this thesis to create smart environments. Some of the functionalities that are planned to be implemented are seamless home automation, customer state monitoring, adaptive environmental customization in vehicles and so on.

Additionally, this work could be extended to do provide multi modal sensing, and prediction and segregation functions can be improved by optimizing all the parameters and additional modalities in addition to ECG. Also, as Spark is open source and supported by almost all cloud technologies, the streaming implementations could be achieved seamlessly, and real time predictions can be done which would make creating smart environments achievable at affordable cost.

Bibliography

Achten, J. and Jeukendrup, A. E. (2003) 'Heart rate monitoring: applications and limitations', *Sports Med*, 33(7), pp. 517–538. doi: 10.2165/00007256-200333070-00004.

Aggarwal, C. C. *et al.* (2003) 'A Framework for Clustering Evolving Data Streams', *Proc. of the 29th int. conf. on Very large data bases*, pp. 81–92. doi: 10.1.1.13.8650.

Banaee, H., Ahmed, M. U. and Loutfi, A. (2013) 'Data mining for wearable sensors in health monitoring systems: a review of recent trends and challenges.', *Sensors (Basel, Switzerland)*, 13(12), pp. 17472–17500. doi: 10.3390/s131217472.

Bulling, A., Blanke, U. and Schiele, B. (2014) 'A tutorial on human activity recognition using body-worn inertial sensors', *ACM Computing Surveys*, 46(3), pp. 1–33. doi: 10.1145/2499621.

Bulling, A., Ward, J. A. and Gellersen, H. (2012) 'Multimodal recognition of reading activity in transit using body-worn sensors', *ACM Transactions on Applied Perception*, 9(1), pp. 1–21. doi: 10.1145/2134203.2134205.

Burns, A. *et al.* (2010) 'SHIMMER^{TM}: An extensible platform for physiological signal capture', in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC'10*, pp. 3759–3762. doi: 10.1109/IEMBS.2010.5627535.

Cai, D., He, X. and Han, J. (2007) 'Semi-supervised discriminant analysis', *Proceedings of the IEEE International Conference on Computer Vision*. doi: 10.1109/ICCV.2007.4408856.

Chen, Y. and Tu, L. (2007) 'Density-based clustering for real-time stream data', in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, p. 133. doi: 10.1145/1281192.1281210.

He, H. *et al.* (2011) 'Incremental learning from stream data', *IEEE Transactions on Neural Networks*, 22(12 PART 1), pp. 1901–1914. doi: 10.1109/TNN.2011.2171713.

Kim, K. H., Bang, S. W. and Kim, S. R. (2004) 'Emotion recognition system using short term monitoring of physiological signals', *Medical Biological Engineering and computing*, 42(Journal Article), pp. 419–427. doi: 10.1007/BF02344719.

Kulkarni, P. and Ade, R. (2014) 'Incremental Learning From Unbalanced Data With Concept Class, Concept Drift and Missing Features: a Review', *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 4(6), pp. 15–29. doi: 10.5121/ijdkp.2014.4602.

Li, K., Du, N. and Zhang, A. (2012) 'Detecting ECG abnormalities via transductive transfer learning', *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine - BCB '12*, pp. 210–217. doi: 10.1145/2382936.2382963.

Luo, Z. *et al.* (2017) 'Label Efficient Learning of Transferable Representations across Domains and Tasks', (Nips). Available at: http://arxiv.org/abs/1712.00123.

Maarala, A. I. *et al.* (2015) 'Low latency analytics for streaming traffic data with Apache Spark', in *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, pp. 2855–2858. doi: 10.1109/BigData.2015.7364101.

Mehmood, A. *et al.* (2016) 'Study of Multi-Classification of Advanced Daily Life Activities on SHIMMER Sensor Dataset', 8(2), pp. 86–92.

Melacci, S. and Belkin, M. (2011) 'Laplacian Support Vector Machines Trained in the Primal', *Journal of Machine Learning Research*, 12, pp. 1149–1184.

Mukhopadhyay, S. C. (2014) 'Wearable sensors for human activity monitoring: A review', *IEEE Sensors Journal*, 15(3), pp. 1321–1330. doi: 10.1109/JSEN.2014.2370945.

Olivier, C., Schölkopf, B. and Zien, A. (2006) *Semi-Supervised Learning*, *Interdisciplinary sciences computational life sciences*. doi: 10.1007/s12539-009-0016-2.

Patel, S. *et al.* (2012) 'A review of wearable sensors and systems with application in rehabilitation', *Journal of NeuroEngineering and Rehabilitation*, 9(1), p. 21. doi: 10.1186/1743-0003-9-21.

Polikar, R. *et al.* (2001) 'Learn++: An incremental learning algorithm for supervised neural networks', *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 31(4), pp. 497–508. doi: 10.1109/5326.983933.

Ravi, D. *et al.* (2017) 'A Deep Learning Approach to on-Node Sensor Data Analytics for Mobile or Wearable Devices', *IEEE Journal of Biomedical and Health Informatics*, 21(1), pp. 56–64. doi: 10.1109/JBHI.2016.2633287.

Razavi-Far, R. *et al.* (2017) 'A Hybrid Scheme for Fault Diagnosis with Partially Labeled Sets of Observations', *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 61–67. doi: 10.1109/ICMLA.2017.0-177.

Salehizadeh, S. M. A. *et al.* (2015) 'A novel time-varying spectral filtering algorithm for reconstruction of motion artifact corrupted heart rate signals during intense physical activities using a wearable photoplethysmogram sensor', *Sensors (Switzerland)*, 16(1). doi: 10.3390/s16010010.

Sigdel, M. *et al.* (2014) 'Evaluation of Semi-supervised Learning for Classification of Protein Crystallization Imagery.', *Proceedings of IEEE Southeastcon / IEEE Southeastcon. IEEE*

Southeastcon, 2014, pp. 1–6. doi: 10.1109/SECON.2014.6950649.

Sindhwani, V., Niyogi, P. and Belkin, M. (2005) 'Beyond the point cloud: from transductive to semi-supervised learning', *Proceedings of the 22nd international conference on Machine learning*, 1, pp. 824–831. doi: 10.1145/1102351.1102455.

Syed, Z. and Guttag, J. (2011) 'Unsupervised Similarity-Based Risk Stratification for Cardiovascular Events Using Long-Term Time-Series Data', *Journal of Machine Learning Research*, 12, pp. 999–1024.

Yilmaz, T., Foster, R. and Hao, Y. (2010) 'Detecting Vital Signs with Wearable Wireless Sensors', *Sensors*, 10(12), pp. 10837–10862. doi: 10.3390/s101210837.

Yin, J., Yang, Q. and Pan, J. J. (2008) 'Sensor-Based Abnormal Human-Activity Detection', *Knowledge and Data Engineering, IEEE Transactions on*, 20(8), pp. 1082–1090. doi: 10.1109/TKDE.2007.1042.