A thesis submitted in fulfillment of the requirements for the degree of Master by Research (Computer Science)

Rongjun XIE
B.Sc Software Engineering
Guangdong University of Technology (GDUT)

School of Science
College of Science, Engineering, and Health
RMIT University
Melbourne, Victoria, Australia
July, 2018
Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Rongjun XIE
January 6, 2019
Acknowledgement

All the praises and thanks be to my parents who devote their lives to supporting my dream. This honor is shared with my father and mother. I would also like to express my sincere gratitude and appreciation to my supervisors, colleagues and friends. It is the greatest gift that God gives me to meet all of you.

First, I would like to thank my supervisors, Ibrahim and Marc, who guided and supported me to accomplish my research goals. Their effective supervision, endless encouragement and continuous support always motivate me especially when I felt depressed and confused.

In addition, I am thankful to my colleagues and friends, Forkan, Vu, Shahriar, Ali, Jesse, Simon, Stray, Sonya, Carol, Ruohan, etc. I do appreciate their kindness and patient in their support, which helped me overcome the worst hardship I have ever had to suffer in my life. I would also like to express my gratitude to the support of RMIT CS&IT for providing a good research environment.

Once again, I praise and thank everyone that directly or indirectly supported me throughout this time. Wish all the best to you and your family. I also hope this piece of work would benefit the research community and help patients with better healthcare services.
Credits

Portions of the materials used in this thesis have previously appeared or under consideration in the following scientific publications:


Contents

Declaration ii

Acknowledgement iii

Credits iv

Contents v

List of Figures ix

List of Tables xi

Abstract 2

1 Introduction 3
  1.1 Backgrounds and Motivation ........................................ 4
  1.2 Research Challenges ................................................ 9
  1.3 Research Questions ............................................... 12
  1.4 Research Contributions ......................................... 13
  1.5 Thesis Organization ............................................. 15

2 Preliminaries 19
  2.1 Extreme Learning Machine (ELM) ................................. 19
  2.2 Incremental learning with ELM .................................. 22
    2.2.1 One-Class ELM ........................................... 23
3 Fast and P2P Learning Healthcare System

3.1 Motivation and Contribution .......................................................... 27
3.2 Related Work .................................................................................. 28
3.3 Methodology .................................................................................... 31
  3.3.1 System scenario and overview ....................................................... 31
    3.3.1.1 Data collection and prediction in a single peer ....................... 32
    3.3.1.2 Data sharing among the peers ............................................ 33
  3.3.2 ELM-SM .................................................................................... 33
    3.3.2.1 The generation of semi-model ............................................ 35
    3.3.2.2 Semi-model version based on MD5 ...................................... 36
    3.3.2.3 Update process of ELM-SM in a peer ................................ 37
3.4 Experimental evaluations and results ................................................. 39
  3.4.1 Accuracy of ELM-SM ................................................................. 40
  3.4.2 Comparison between traditional ELM and ELM-SM in terms of training time ................................................................. 41
3.4.3 Performance of P2P learning ......................................................... 42
    3.4.3.1 Experimental Setup for the proposed P2P learning system ... 42
    3.4.3.2 Varying the size of the network ......................................... 46
    3.4.3.3 Varying the number of initial learning peers ....................... 47
    3.4.3.4 Communication cost in terms of exchanging the semi-models 48
3.4.4 Comparison of data size between the semi-model and raw data .... 49
3.5 Summary ......................................................................................... 51

4 Collaborative ELM (CELM) with A Confidence Interval (CI)

4.1 Motivation and Contribution .......................................................... 54
  4.1.1 Motivations .............................................................................. 54
CONTENTS

4.1.2 Contributions ........................................... 55
4.2 Related Work ............................................... 56
4.3 System model ............................................... 58
4.4 Overview of our proposed system ......................... 60
  4.4.1 Overview of our proposed classifier ................... 61
4.5 Proposed Collaborative CELM with a CI .................... 64
  4.5.1 Function definitions in our algorithms ............... 65
  4.5.2 Data Label Domain Transformation (DLDT) ............ 66
  4.5.3 Knowledge-parameter generation of a dataset ........ 68
    4.5.3.1 Generating the first component ($C_1$) in a knowledge-parameter 68
    4.5.3.2 Generating the second component ($C_2$) in a knowledge-parameter 69
  4.5.4 The calculation of a Confidence Interval (CI) ........ 71
  4.5.5 Learning process of our proposed model .............. 73
  4.5.6 Prediction process for our proposed model .......... 76
  4.5.7 Time complexity ..................................... 77
4.6 Experimental results ...................................... 78
  4.6.1 Data preprocessing .................................. 79
    4.6.1.1 Data cleaning .................................. 79
    4.6.1.2 Data grouping .................................. 79
  4.6.2 Identifying optimal parameters for CELM initialization 80
  4.6.3 Evaluation of our proposed model in a peer during P2P learning . 81
    4.6.3.1 Experimental workflow ......................... 81
    4.6.3.2 Accuracy of one-class classification during P2P learning .... 82
    4.6.3.3 General performance during P2P learning (from one-class to multiclass) ........ 84
  4.6.4 Analytic comparison with existing works ............. 87
4.7 Summary .................................................. 88

5 Data Priority for P2P Healthcare System .................... 89
  5.1 Motivation and Contribution ............................. 90
5.2 Related Work ................................................................. 91
5.3 Overview of our proposed system ................................. 92
5.4 Methodology ................................................................. 96
  5.4.1 Vital biosignals and clinical events ......................... 96
  5.4.2 Criteria for data assessment ................................. 97
  5.4.3 Time-Window Based Processor (TWBP) ...................... 98
    5.4.3.1 Time-Window Data Chunk (TWDC) and its priority ...... 99
    5.4.3.2 Data collection with the filter ..................... 100
  5.4.4 Priority processor ................................................. 101
5.5 Experimental evaluations and results ............................ 103
  5.5.1 Data preprocessing ............................................. 103
  5.5.2 Priority classification ........................................ 104
  5.5.3 Data collection efficiency .................................. 105
  5.5.4 The average waiting time ..................................... 107
5.6 Summary ................................................................. 109

6 Conclusion ................................................................. 111
  6.1 Limitations and Future Directions of Research ............ 114

Bibliography ............................................................... 117
List of Figures

1.1 The typical architecture of remote healthcare systems ........................................ 5
1.2 Collaborative healthcare community .............................................................. 7
1.3 The comparison between the centralized-based learning model and P2P learning model .......................................................... 8
1.4 The example of the P2P learning system ......................................................... 9
2.1 The structure of ELM ................................................................. 20
2.2 An example of a sample in 60 minutes ......................................................... 24
3.1 Proposed healthcare system overview ............................................................. 32
3.2 Example of data sharing ................................................................. 34
3.3 The process of proposed semi-model generation based on ELM .................... 34
3.4 Example of a semi-model and its MD5 value .............................................. 37
3.5 The process of updating the semi-model in a peer ........................................ 38
3.6 Comparison of training time between ELM and ELM-SM ......................... 42
3.7 Example of P2P learning process of the proposed system in the 3D network topology .... 44
3.8 Example of step calculation in a peer ....................................................... 46
3.9 Required steps (shown in log scale) in terms of varying the size of the P2P network .... 47
3.10 Required steps in terms of varying the number of initial learning peers .......... 48
3.11 The maximum number of semi-model packets sent and received in a peer during the learning process ........................................ 49
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.12</td>
<td>Comparison of data size between a semi-model and its raw data in the P2P learning process.</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison between a OS-learning classifier and a non-OS-learning classifier.</td>
</tr>
<tr>
<td>4.2</td>
<td>Proposed healthcare system overview.</td>
</tr>
<tr>
<td>4.3</td>
<td>The P2P learning process and prediction process in a peer.</td>
</tr>
<tr>
<td>4.4</td>
<td>Learning process based on a knowledge-parameter $C_1$ and $C_2$.</td>
</tr>
<tr>
<td>4.5</td>
<td>Examples of prediction process of the proposed model with one, two and three outputs.</td>
</tr>
<tr>
<td>4.6</td>
<td>Overall accuracy of the CELM without the CI for all training samples (D0-4).</td>
</tr>
<tr>
<td>4.7</td>
<td>Accuracy of one-class prediction in a peer during the P2P learning process.</td>
</tr>
<tr>
<td>4.8</td>
<td>Accuracy in a peer during the P2P learning process.</td>
</tr>
<tr>
<td>4.9</td>
<td>The confidence upper bound in a peer during the P2P learning process.</td>
</tr>
<tr>
<td>4.10</td>
<td>Filter rate in a peer during the P2P learning process.</td>
</tr>
<tr>
<td>5.1</td>
<td>The proposed remote healthcare system.</td>
</tr>
<tr>
<td>5.2</td>
<td>The workflow of the time-window based processor.</td>
</tr>
<tr>
<td>5.3</td>
<td>An example of data collection.</td>
</tr>
<tr>
<td>5.4</td>
<td>The workflow of the priority processor.</td>
</tr>
<tr>
<td>5.5</td>
<td>The ratio of data sent to the server with different abnormal data ratios.</td>
</tr>
<tr>
<td>5.6</td>
<td>The ratio of requests of sending data with different abnormal data ratios.</td>
</tr>
<tr>
<td>5.7</td>
<td>The waiting time of records with and without priority. Every record is processed in Time $t$.</td>
</tr>
</tbody>
</table>
List of Tables

2.1 The targeted events for evaluation, their labels and generalized normal value of vital biosignals .......................................................... 25
3.1 Time consumption of related works .................................................. 29
3.2 Recent research works in healthcare based on machine-learning algorithms ................................................................. 30
3.3 The number of training samples in five events ........................................ 40
3.4 The number of testing samples in five events ........................................ 40
3.5 The performance measure of each event using ELM-SM with 500 hidden neurons and the sigmoid activation function. Here 95.47% is obtained .................................................. 41
4.1 Summary of notations. ................................................................. 65
4.2 The time complexity of each proposed algorithm ...................................... 78
4.3 The training dataset ................................................................. 79
4.4 The testing set ................................................................. 80
4.5 Comparison among our model and similar works in terms of results and features 88
5.1 Vital biosignals and their normal range. ................................................ 96
5.2 Vital biosignals defining the triage. .................................................... 98
5.3 Priority definition by score of vital biosignals .......................................... 98
5.4 The preprocessed data with 4 data priorities. ......................................... 104
5.5 The accuracy comparison among different learning algorithms. ............... 104
5.6 The confusion matrix after performing the classification. Here 80% accuracy is obtained ................................................................. 105
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.7</td>
<td>The performance measure of each priority using ELM.</td>
<td>105</td>
</tr>
<tr>
<td>5.8</td>
<td>Waiting time comparison</td>
<td>109</td>
</tr>
</tbody>
</table>
Abstract

Modern e-healthcare systems are prevalent in many medical institutions to reduce physicians’ workload and enhance diagnostic accuracy, which leverages affordable wearable devices and Machine-Learning (ML) techniques. The healthcare systems collect various vital biosignals (e.g., heart rate and blood pressure) from wearable devices of users (e.g., chronic patients living alone at home) and analyze these patients’ data in real-time by different ML classifiers (e.g., Support Vector Machine (SVM) or Hidden Markov Model (HMM)). The automatic diagnosis effectively improves the physicians’ performance in terms of diagnostic efficiency and accuracy. There are three challenges impacting these healthcare systems – the increasing number of patients, new diseases and the changes of existing disease patterns, which are caused by population aging as well as the alteration of environment and lifestyle. This research is intended to explore a novel healthcare system with advanced ML solutions that can solve the challenges and exhibit high accuracy and efficiency.

We begin by designing a Peer-to-Peer (P2P) learning system, which provides efficient and robust processes of sharing and learning knowledge generated from raw patients’ data. Nowadays medical institutions have collected enormous volumes of patients’ data for knowledge discovery, but they rarely consider sharing their knowledge with other medical institutions, which can increase the diagnostic accuracy efficiently. In order to break the isolation, our P2P system enhances the healthcare system with the sharing ability. Since centralized models have huge communication costs for sharing massive amounts of personal vital biosignal data among the institutions for the training purpose, the decentralized structure (P2P technique) is adopted in our solution. This model builds a foundation for sharing medical knowledge in a distributed way.
In the next step, to enhance the model with the incremental learning ability of new unknown data labels, we build Collaborative Extreme Learning Machine (CELM) with a Confidence Interval (CI). More specifically, when encountering samples with new data labels, the enhanced model can provide a fast learning process and a correct prediction process. In order to improve the learning efficiency with new data labels, our proposed solution eliminates redundant calculation in the learning process. In addition, our model improves the prediction accuracy by considering where true predictions are likely to lie. The extensive experimental analysis shows that the proposed model is efficient and achieves high accuracy in diagnosing clinical events by analyzing patients’ vital biosignal data. The efficient learning and accurate prediction schemes are proposed in our healthcare system, however, the process of data collection is not efficient.

Finally, we optimized our P2P healthcare system by introducing data priority which can represent patients’ urgency. Since enormous amounts of vital biosignal data from patients’ wearable sensors are collected and sent to the system for analysis, our algorithms of assessing data priority can effectively reduce the volume of data transmitted in the data collection process. In addition, the waiting time of data before prediction can be optimized because data with higher priority are processed in front of those with lower priority, which helps our system to provide diagnostic decisions in a proper time according to patients’ urgency.

In a nutshell, this thesis provides a novel healthcare system which is robust, efficient, practical and optimal. This research introduced various new algorithms built on various technologies related to machine-learning, distributed computing and computer networking. These solutions addressed the efficiency issues in the processes of data collection, learning and prediction. Therefore, the research contributions in this thesis present P2P learning and prediction schemes as well as data assessment algorithms to provide a more efficient and reliable healthcare system. We believe that this research introduces a new learning scheme which can be introduced into different healthcare systems (ambient assisted living or healthcare monitoring systems).
Chapter 1

Introduction

Investment in the global health information technology market is expected to increase tremendously from US $96.8 billion in 2013 to US $210.3 billion by 2020 [53], making healthcare systems more and more popular in many different medical institutions across the world [53]. Such a significant increase clearly reflects how the public and government agencies are focusing on improving healthcare systems, which contributes to longer life expectancy and less healthcare expenditure [53][17].

Different kinds of healthcare systems have been explored with the rapidly changing landscape of health information technology and Machine-Learning (ML) technology. These healthcare systems collect a variety of health information from patients and use these data to guide medical decisions for clinicians [7]. These data are used to train those smart healthcare systems, which make the systems smarter. As one of the common healthcare systems, the Clinical Decision Support System (CDSS) is used to provide reliable and automatic diagnoses, which thereby can ensure care quality and safety in different medical institutions [19] and can reduce the workload of physicians with large numbers of patients.

Moreover, recent advances in Internet of Things (IoT) accelerate development of smart healthcare systems by providing data with broader types [12]. With approximately 90 million wearable devices in different fields including healthcare in 2014 [60], a new segment of IoT has emerged as “Wearable IoT” (WIoT). Healthcare systems with WIoTs lead a gradual shift from hospital-centered systems to a person-centered environment [30][23][34], which can monitor
Backgrounds and Motivation

Consequently, with significant advances in body sensors, IoT devices and ML technology, remote healthcare monitoring systems embedded with CDSS are proposed by different studies [23][39] and many healthcare IoT solutions are provided by different companies (Microsoft\(^1\), Alter Calsoft Labs\(^2\) and KAA\(^3\)) to help patients and hospitals. These intelligent systems provide reliable solutions to remotely monitor the elderly with chronic diseases [75], which are the main reason of many deaths in Australia and other western countries [23]. These diseases are usually caused by irregular lifestyle, improper diet, and congenital genetic problems [77]. In these healthcare systems, the patients’ data are collected from the IoT devices continuously and are transmitted to medical institutions for further analysis. After analyzing the data, a proper clinical decision is sent, which is related to the patients’ health status in real time.

By taking the leverage of various advanced technologies, smart healthcare systems are playing an important role in improving medical services. But now these intelligent healthcare systems are challenged by the increasing volume of patients’ data, new diseases and the changes of existing disease patterns, which require an advanced intelligent system to provide fast processing speed with good diagnostic accuracy.

1.1 Backgrounds and Motivation

Various research works have been providing promising solutions on smart healthcare systems. CDSS is one of the common smart healthcare systems, which can diagnose different medical conditions using patients’ medical records by different ML classifiers. For example, the study in [42] uses ECG data to detect patients’ heart issues. Multi-biosignals are used to predict 5 clinical events in [22]. The solution in [38] identifies liver cancer by magnetic resonance imaging (MRI).

\(^{2}\)https://www.altencalsoftlabs.com/healthcare-iot-platform/
\(^{3}\)https://www.kaaproject.org/healthcare/
Figure 1.1: The typical architecture of remote healthcare systems. Portable smart devices collect multiple biological signals of a patient with body sensors and send these data to the server. The clinical institution can host the server in the Cloud or locally. A CDSS is used to do the smart prediction which supports the diagnosis from clinicians. If an abnormal event is detected, the system can notify the patient properly.

Except for the CDSS, there are different kinds of smart systems serving patients. Among them, healthcare monitoring systems are proposed in many studies to help users for whom it is difficult to access medical services. As an example, Ambient Assisted Living (AAL) has been explored by different researchers [35][44]. AAL is a technology-based approach from information and communication technology (ICT), which can be used to support patient or elderly people at home. AAL aims to secure the health of different users, allowing the users living independently in their own home. It simplifies the activities of daily living with home automation and provides reliable intelligent solutions to monitor and care for the users.

Figure 1.1 shows a typical architecture of healthcare monitoring systems, which includes two main components: the data collection module and a data processing platform. As an example, the smart IoT device continually collects the bio-signal data from patients by their wearable sensors and then sends the data to CDSS. The CDSS can be hosted in a Cloud-based
platform or a local server employed by a medical institution. The system not only provides smart decisions (classification or prediction) to help clinicians (e.g., doctors or nurses), but also sends a proper notification to the patient once an abnormal clinical event is detected. The medical data from patients are also used as training datasets for updating the CDSS. The updated CDSS can have better diagnostic accuracy and detect more diseases in case of future events.

Since huge volumes of raw healthcare data are generated from patients everyday [58], the healthcare systems have heavy pressure on three stages of processing these medical data, which are data collection, data learning, and data prediction. These data generated by the sensors have three main characteristics of big data: volume, velocity, and variety [5]. The case becomes more critical with the more elderly population for continuous monitoring. The efficient learning process with big data is improved by adopting Cloud platforms used in many studies [39][22], while other challenges (e.g., privacy) are analyzed in [74]. Even though researchers have proposed various intelligent systems for healthcare to solve various challenges, there are two significant limitations in the existing solutions as follows, impacting on the system performance.

- **Isolated system:** The system does not share their \textit{knowledge generated from patients’ data} with healthcare systems used by different medical institutions. Sharing knowledge can improve diagnostics accuracy effectively and efficiently, but the system has to consider an efficient and robust method to learn from new unknown knowledge which is sent by other systems.

- **Data processing without priority:** The system processes data equally without considering patients’ urgency, which leads to potential delays in treatment of severe conditions of patients. The data processing includes data transmission from IoT devices to the medical institutions (namely data collection), and data prediction in the medical institutions.
In order to address the isolation, every healthcare system in a medical institution with different locations should be linked together by various networking techniques. As a result, a collaborative healthcare community is achieved to share medical knowledge among different medical institutions. Such community shown in Figure 1.2 can be implemented based on a specific network topology which is typically classified into the centralized network or the distributed/Peer-to-Peer (P2P) network.

Considering the typical architecture of Distributed Data Mining approaches [20] based on the centralized network, we could easily design a data-sharing model shown in Figure 1.3(a), which is beneficial in terms of sharing and learning the data in a centralized manner. Firstly, every client gathers data and sends them to the central server. Secondly, the central server updates its own classifier from those data. Finally, the latest updated classifier is synchronized to all clients. However, the centralized model certainly faces some noticeable limitations as below.

- **Intensive central dependence:** All clients depend on the central server and are isolated from other clients. During the maintenance or failure period of the central server,
clients are not able to receive any updates from the central server.

- **Imbalanced sharing and learning processes:** The central server requires intensive computation and communication resources if the size of data samples and the number of clients increase significantly.

![Figure 1.3](image)

Figure 1.3: The comparison between the centralized-based learning model and P2P learning model. In centralized-based learning model, every medical institution sends bio-signal data from patients’ IoT devices to the central server for processing. However, in our proposed P2P learning model, medical institutions are connected without the central server. The medical knowledge extracted from bio-signal data is shared among medical institutions.

Therefore, the P2P network which is shown in Figure 1.3(b) is used to enhance the healthcare systems, which can eliminate the dependency of the central server and balance the load in every medical institution. In the P2P healthcare system, the sharing and learning process is defined as P2P learning in our study, which is also considered as an incremental learning process [26]. A scenario to illustrate a P2P learning system is shown in Figure 1.4. Suppose that there are three hospitals (H1, H2, H3) which collect and predict data from patients. At first, H1 only has a dataset of Disease A and can only detect Disease A. H2 only has a dataset of Disease B and can only detect Disease B. H3 only has a dataset of Disease C and can only detect Disease C. After sharing and learning knowledge among H1, H2 and H3, all of them have the ability to detect Disease A, B and C.
Our main objective of this study is to explore an enhanced intelligent healthcare system to overcome these limitations by supporting P2P learning and processing data with the consideration of patients’ urgency. The system is required to provide good diagnostic accuracy and to process data efficiently in terms of collecting data, learning from data and predicting data.

1.2 Research Challenges

We aim to design a fast P2P learning system with which every medical institution (peer) can improve its diagnostics accuracy by learning knowledge shared from other institutions and can optimize the processing flow of patients’ data based on different urgency. In addition, the system can diagnose new diseases without touching raw training datasets with the new diseases. In order to achieve our learning system, there are five research challenges (RCs) required to be solved.

RC-1. Existing ML classifiers consume large amounts of time for training and do not support P2P learning.

Various ML classifiers (e.g. Support Vector Machine (SVM) [14], Hidden Markov Model (HMM) [23][56][70] and Neural Network (NN) [37][29]) have been proposed by many studies for healthcare diagnosis. The majority of these existing classifiers have slow training times,
which restricts the healthcare application of them since many bio-features of existing diseases are changed and new diseases are discovered with the change of lifestyle and environment. For example, HMM takes 30 min in an x3.large Amazon virtual machine to be trained with 600 samples [23]. Even though ELM [36] provides fast and good prediction accuracy for analyzing medical data, ELM does not have the P2P learning capacity.

**RC-2.** The raw training medical data are large.

A variety of health information collected from patients are used to guide decisions from clinicians [7] and train those smart healthcare systems. For example, the advanced technology in wearable sensors has made it possible to monitor multiple vital signs of a patient anytime, anywhere. As multiple vital signs from a large number of patients are accumulated, the issue of big data is evolved. For instance, vital signs such as Heart Rate (HR), Blood Pressure (BP), Respiratory Rate (RR) and Oxygen Saturation (SPO\textsubscript{2}) are a crucial part of big medical data [86]. If the numerical value of each vital sign contains 4 bytes and the frequency of data collection is 1 minute, then for 6 vital signs total 24 bytes data gathered per minute, which is equivalent to 33.75 Kilobyte (KB) per day, or 12 Megabyte (MB) per year. If such data are gathered from 5 million patients, then the data amount will be 57.3 Petabyte (PB) per year. Sharing these large amounts of data among different medical institutions is unfeasible in terms of time.

**RC-3.** Every peer has different orders of receiving and learning knowledge which is shared by other peers.

In our design of the P2P learning system, every peer shares the knowledge and learns the knowledge from other peers, which highly relies on the capacity of the network and the performance of a computer. For example, network latency considers the time it takes for a packet of data to get from one designated point to another and bandwidth is the capacity of a wired or wireless network communications link to transmit the maximum amount of data from one
Research Challenges

point to another over a computer network or internet connection in a given amount of time. The network latency and bandwidth have a significant impact on the packet receiving time in a peer, which means lower latency and higher bandwidth can reduce the transmission time. In addition, CPU speed is the rate where a processor can complete a processing cycle, which is typically measured in Megahertz (MHz) or Gigahertz (GHz), which contributes to the overall performance of a computer. This means a 1.8 GHz processor has approximately twice the speed of a 900 MHz processor. So the P2P learning system faces difficulties related to uncertain receiving and learning orders among peers caused by different processing performances of networks and computers.

RC-4. Every peer has dynamic data labels in both learning and prediction processes.

The traditional learning and prediction requires data labels to be predefined and fixed, which is non-robust in real cases. With the development of technologies and the changing lifestyle, various new diseases will be discovered in different medical institutions. Different locations also impact on the evaluation of diseases. For example, developed counties (e.g. America and Australia) have more cases of diabetes while more patients in developing countries (e.g. India and Africa) are suffering from malaria. This means that the knowledge shared by different medical institutions contains various diseases. For example, in Figure 1.4, Peer H1 has to learn knowledge containing unknown data label $B$ and $C$, while data label $A$ and $B$ are unseen to Peer H3. In addition, a medical institution is likely to predict new samples which do not belong to any learned diseases. For instance, Peer H1 performs prediction of samples with Disease $C$ when only having the knowledge of Disease $A$ and $B$. These issues related to dynamic data labels need to be solved in both learning and prediction processes of the P2P learning system.

RC-5. The patient’s urgency is dynamic with the consideration of time-sequential data.

Biosignal data are commonly collected from patients and are used for medical prediction
Research Questions

in healthcare systems. As shown in Figure 1.1, these biosignal data are time-sequential data, which usually are collected in a specific frequency (e.g. every minute or 10 minutes). Based on clinical priority settings, these data can be provided different priorities which are used to sort the flow of patients so patients with more urgent conditions should be diagnosed or treated before those with less urgent conditions [54]. But these time-sequential data have different numbers of increasing and decreasing trends, illustrated in Figure 2.2, which makes patient urgency change over time. The accuracy and efficiency to assess patients’ urgency with time-sequential biosignal data are considered in this challenge.

1.3 Research Questions

In order to overcome the research challenges, we define the following three research questions to enhance the healthcare system.

RQ-1. How to develop a fast and P2P learning system for healthcare to predict clinical events using vital biosignal?

This research question addresses the challenges related to the learning and sharing efficiency in the P2P learning system described in RC-1 and RC-2. The significance of dealing with these issues is primarily to provide an efficient P2P architecture for further study. Since different medical data can be used for healthcare diagnoses, vital biosignals (e.g. heart rate and blood pressure) are targeted to implement and analyze our P2P learning system.

RQ-2. How to enhance our proposed healthcare system in RQ-1 with an ability of distributed online-sequential learning (OS-learning) [47] and prediction?

This research question is designed to address the challenges related to uncertain learning orders among peers and dynamic data labels explained in RC-3 and RC-4. Specifically, the aim of this research question is to improve the robustness of our P2P system by embedding OS-learning
Research Contributions

[47], which allows ML classifiers to learn from training data available in a sequential order.

**RQ-3.** How to optimize the processing flow of our healthcare system in *RQ-2* in terms of data collection and prediction?

From the solutions of *RQ-1* and *RQ-2*, a robust, learning-efficient and sharing-efficient P2P system is achieved for healthcare diagnoses. However, the current system does not consider the urgency of patients’ data when collecting and predicting data, which impacts on the medical services provided to patients in terms of waiting time and the collected data volume. This research question addresses this issue by solving *RC-5* related to the assessment of patient’s urgency. This research question aims to optimize our P2P system and make it more practical and efficient by introducing patients’ urgency.

1.4 Research Contributions

To address the aforementioned research questions, extensive studies have been performed. We explored novel systems and algorithms which are inspired by different techniques related to computer network, machine learning, big data and distributed computing. In general, the contributions of this research are summarized as follows.

1. **Fast and P2P Learning Framework for Sharing Knowledge**

Wearable devices in the IoT make home-based personal healthcare systems popular and affordable. With an increasing number of patients, such healthcare systems are challenged to store and process enormous volumes of data. Some medical institutions employ Cloud services to meet requirements of analyzing big data without considering sharing their own knowledge which could increase diagnostic accuracy efficiently. In order to obtain such collaborative healthcare community in the Cloud environment, we propose a P2P learning system which is fast, robust and learning-efficient. Our system continuously collects vital biosignals from wearable devices of users (e.g., chronic patients living alone at home) and analyzes the biosignals in real-time with ELM [36]. The traditional centralized learning
Research Contributions

models suffer in having huge communication costs to share massive amounts of personal vital biosignal data among the institutions for the training purpose, explained in Section 1.1. Our P2P learning model can overcome this limitation by allowing every institution to maintain its own raw data while also being updated by other institutions shared knowledge a.k.a semi-model which is lightweight output during the training process, as well as being smaller than raw data. In relation to RQ-1, Chapter 3 contains the details description of this system and its main contributions correspond to [80].

2. Enhanced Classifier for P2P Learning to Handle Unlearned Data Labels

ML-based healthcare systems are employed by many medical institutions, which collect biosignal data continuously from patients and detect their clinical status (e.g. diseases) by various machine-learning based classifiers. The classifiers are required to be trained frequently in order to improve the diagnostic accuracy. To facilitate the improvement in the training efficiency and diagnostic accuracy with a large amount of patients’ data, the P2P system is explored in Chapter 3. But the system lacks the ability to detect data with unlearned knowledge which affects the disease prediction accuracy significantly. So we propose CELM with a Confidence Interval in Chapter 4 related to RQ-2, which is based on traditional ELM, available in our P2P system for improving the efficiency and accuracy in disease prediction. More specifically, our model is able to learn continuously from samples (known as OS-learning [47]) with uncertain data labels during the P2P learning process while filtering prediction results with a CI, which shows the confidence interval where true predictions are likely to lie. One-class\(^4\) to multiclass prediction during the P2P process is also supported by our model, which enhances the robustness of our P2P system.

3. Optimal Algorithms for Data Collection and Prediction with Patient Urgency

A robust, learning-efficient and sharing-efficient P2P healthcare system has been proposed in Chapter 4, which allows every medical institution sharing its knowledge with

\(^4\)One-class classification tries to identify data of a specific class by learning from a training set containing only the samples of that class.
others to improve the diagnostic accuracy. Except for the learning and sharing process, the processing flows of data collection and prediction have a significant impact on our P2P healthcare system. In our healthcare system, multi-biosignal data are collected continuously from patients by various body sensors and are sent to a medical institution by portable devices for further analysis (e.g. knowledge discovery or the clinical event prediction). Instead of sending the time-series patients’ data continuously, the data are organized and transmitted in a specific time window based on patients’ urgency, which is called a Time Window Data Chunk (TWDC). Concretely, each TWDC is given a priority which is evaluated by our proposed assessment algorithm and is able to represent the patients’ urgency. Data with a higher priority are processed in front of those with a lower priority, which can optimize the waiting of data before prediction. Our proposed assessment algorithm in relation to RQ-3 is inspired by National Early Warning Score (NEWS) [2] used in Emergency Department (ED). In our system, only the valuable TWDCs for analysis which are decided by the data priority are sent to the medical institution, which can reduce the volume of transmitted data. By analyzing patients’ urgency in the data, our P2P healthcare system becomes more optimal and practical. The detailed description of this assessment algorithm is explained in Chapter 5.

1.5 Thesis Organization

The organization of this thesis includes the following chapters as below:

- **Chapter 2: Preliminaries**

  The theoretical foundations of techniques that we used to explore our P2P learning system are presented in this chapter, covering ELM, the incremental learning process of ELM and One-Class ELM. These techniques are related to ELM classifier with no need to tune the hidden layer and are adapted to solve different challenges in our P2P learning system in the following chapters. In addition, the dataset used to evaluate the models in the following chapters is described in this chapter, which is preprocessed with the MIMIC-II [64] dataset from MIT Physiobank archive.
• Chapter 3: Fast and P2P Learning Healthcare System

In order to solve \textit{RQ-1}, a P2P learning system which is fast, robust and learning-efficient is proposed in this chapter. Our system allows every medical institution to share knowledge with other institutions for improving the diagnostic accuracy individually. The shared knowledge named \textit{semi-model} has small data size, which is generated from the raw data. The extensive experimental analysis demonstrates that our P2P learning model is efficient in learning and sharing for patient diagnosis. The potential impact under different network topologies, network sizes and the number of learning peers are also highlighted.

\textit{Copyright/ credit/ reuse notice:} The contents of this chapter have been taken and revised as needed from our paper published as:


DOI: https://doi.org/10.1016/j.future.2018.05.042

©2018 Elsevier B.V. All rights reserved.

• Chapter 4: CELM with A Confidence Interval (CI)

In relation to \textit{RQ-2}, this chapter presents an enhanced ML classifier to make our healthcare system proposed in \textit{RQ-1} more efficient, robust and accurate by supporting the incremental learning (known as online sequential learning) and filtering prediction results without confidence. Our classifier is named \textit{CELM with A Confidence Interval} and the CI shows the interval where true predictions are likely to lie. The extensive experimental results are analyzed to show that the model is fast in the incremental learning process and achieves high accuracy in diagnosing clinical events by analyzing patients’ vital biosignal data.

\textit{Copyright/ credit/ reuse notice:} The contents of this chapter have been taken and revised as needed from our paper published as:
DOI: https://doi.org/10.1016/j.comnet.2018.11.002
©2018 Elsevier B.V. All rights reserved.

- **Chapter 5: Data Priority for P2P Healthcare System**

In this chapter, novel algorithms are presented in relation to the challenges stated in RQ-3, which can be embedded with our P2P learning system explored above. These algorithms make our healthcare system more efficient and practical by introducing data priority. The data priority is explained in detail, which is evaluated by patients’ time-series biosignal data and can represent patients’ urgency. Our extensive experiments are provided to show that our proposed method can reduce the volume of data transmission and the waiting time of data for further analysis is optimized.

- **Chapter 6: Conclusion**

The thesis is concluded in this chapter by summarizing the main contributions, key findings and limitations of the proposed intelligent healthcare system. In addition, the significance of this research and potential future directions are also discussed.

In summary, the succeeding core chapters (Chapter 3–5) of this thesis contribute to serveral key research questions on intelligent healthcare systems in terms of P2P learning and data assessment, which directs future research. The MIMIC-II [64] dataset from MIT Physiobank archive is used to evaluate the performance of each model in every chapter. More specifically, six biosignals and five clinical events are targeted in this thesis, which is described in Section 2.3. Note that the core chapters appear in a self-contained and self-explanatory manner. Therefore, the relevant contexts and content including discussions on related works, description of the architecture, experimental setups and evaluation metrics are presented in each of these chapters separately. All experiments in this thesis are implemented in MATLAB version 9.1.0.441655
(R2016b) installed in Microsoft Windows 7 Professional X64 operating system with Intel Core i5 3.10GHZ processor and 6 Gigabyte physical memory.
Chapter 2

Preliminaries

Basic ELM, the incremental learning process of basic ELM and one-class ELM are introduced in this chapter, providing theatrical foundation to develop our P2P healthcare system. In addition, the preprocessed dataset which is used to evaluate the performance of our proposed methods in all chapters is described.

2.1 Extreme Learning Machine (ELM)

The ELM [36] is a Single Hidden Layer Feed Forward Neural Network (SLFN) without tuning the hidden layer. The main advantage of ELM is that it overcomes limitations of backpropagation algorithms which are commonly used in artificial neural networks by randomly generating input weights and analytically calculating output weights. The main limitations of backpropagation include over fitting, high computation cost of the learning process and local minima. Moreover, the ELM’s learning speed and performance are also significantly better than other conventional learning algorithms.
Figure 2.1: The structure of ELM. Samples with labels and features are fed into the input layer in ELM and then are operated with random input weights and an activation function in the hidden layer. The final model is calculated based on output matrices from the hidden layer.

As a structure of ELM shown in Figure 2.1, there are three different layers in ELM: the input layer, the hidden layer and the output layer. Suppose that ELM has $M$ neurons in the input layer, $K$ neurons in the hidden layer and $C$ neurons in the output layer, for $N$ arbitrary
Extreme Learning Machine (ELM)

21 distinct samples \((\vec{x}_i, \vec{t}_i)\), where \(\vec{x}_i = [x_{i1}, x_{i2}, x_{i3}, \cdots, x_{im}]^T \in \mathbb{R}^M\) and \(\vec{t}_i = [t_{i1}, t_{i2}, t_{i3}, \cdots, t_{ic}] \in \mathbb{R}^C\). The \(m, i\) and \(c\) represent the index of features, samples and neurons respectively. ELM with \(K\) hidden neurons is mathematically modeled as

\[
\vec{o}_i = \sum_{k=1}^{K} [\vec{\beta}_k \cdot G(\vec{w}_k, b_k, \vec{x}_i)] = \sum_{k=1}^{K} [\vec{\beta}_k \cdot G(\vec{w}_k \cdot \vec{x}_i + b_k)]
\]

(2.1)

where \(\vec{w}_k = [w_{k1}, w_{k2}, w_{k3}, \cdots, w_{km}]^T\) and \(b_k\) are random input weights in the \(k\)-th hidden node, \(\vec{\beta}_k = [\beta_{k1}, \beta_{k2}, \beta_{k3}, \cdots, \beta_{kc}]^T\) is the weight vector connecting the \(k\)-th hidden node and the output nodes, \(\vec{o}_i = [o_{i1}, o_{i2}, o_{i3}, \cdots, o_{ic}]\) is the \(i\)-th output vector of ELM, and finally \(G(\ast)\) corresponds to an output of an activation function used in neurons of the hidden layer. Particularly, the value of elements in \(\vec{t}_i\) is 1 when the output of neuron belongs to the sample class and the rest are \(-1\). ELM can evaluate these \(N\) samples with zero error, which is the basic principle of least squares algorithm. The evaluation is shown in

\[
\sum_{n=1}^{N} \|\vec{o}_n - \vec{t}_n\| = 0
\]

(2.2)

and can be expressed as

\[
H \cdot \beta = T
\]

(2.3)

where

\[
H = \begin{bmatrix}
G(\vec{w}_1, b_1, \vec{x}_1) & \cdots & G(\vec{w}_k, b_k, \vec{x}_1) \\
\vdots & \ddots & \vdots \\
G(\vec{w}_1, b_1, \vec{x}_n) & \cdots & G(\vec{w}_k, b_k, \vec{x}_n)
\end{bmatrix}_{N \times K} = \begin{bmatrix}
g_{11} & \cdots & g_{1k} \\
\vdots & \ddots & \vdots \\
g_{n1} & \cdots & g_{nk}
\end{bmatrix}_{N \times K}
\]

(2.4)

\[
\beta = \begin{bmatrix}
\vec{\beta}_1 \\
\vdots \\
\vec{\beta}_k
\end{bmatrix}_{K \times C} \quad \text{and} \quad T = \begin{bmatrix}
\vec{t}_1 \\
\vdots \\
\vec{t}_n
\end{bmatrix}_{N \times C}
\]

(2.5)

\(H\) is named as the hidden layer output matrix of ELM with a specific input dataset \(X = [\vec{x}_1, \vec{x}_2, \vec{x}_3, \cdots, \vec{x}_n]\). The smallest norm least-squares solution of above linear system can be expressed as:

\[
\hat{\beta} = H^\dagger \cdot T = (H^T H)^{-1} H^T T
\]

(2.6)
\[ H^T H = \begin{bmatrix} u_{11} & \cdots & u_{1k} \\ \vdots & \ddots & \vdots \\ u_{k1} & \cdots & u_{kk} \end{bmatrix}_{K \times K} \] (2.7)

and

\[ H^T T = \begin{bmatrix} v_{11} & \cdots & v_{1c} \\ \vdots & \ddots & \vdots \\ v_{k1} & \cdots & v_{kc} \end{bmatrix}_{K \times C} \] (2.8)

where \( H^\dagger \) is the pseudo inverse which extrapolates the inverse of matrix \( H \) in Equation (2.3).

### 2.2 Incremental learning with ELM

In order to speed up the learning process with big data, an incremental learning approach based on ELM is proposed in [81].

Let the original training dataset be \( D = (X, T) \) where \( D = \{(\vec{x}_j, \vec{t}_j) | \vec{x}_j \in \mathbb{R}^M, j = 1, 2, \cdots, N \} \) and \( N \) is the number of original samples. The newly arrived training dataset is represented by \( \Delta D = (\Delta X, \Delta T) \), where \( \Delta D = \{(\vec{x}_j, \vec{t}_j) | \vec{x}_j \in \mathbb{R}^M, j = N + 1, N + 2, \cdots, N + \Delta N \} \) and \( \Delta N \) is the number of new samples. When the newly arrived training dataset is merged with the original one, we have \( D' = (X, \Delta X) \), \( T' = \{(\vec{x}_j, \vec{t}_j) | \vec{x}_j \in \mathbb{R}^M, j = 1, 2, \cdots, N, N + 1, N + 2, \cdots, N + \Delta N \} \).

The new hidden layer output matrix \( H' \) can be derived from \( H \) and \( \Delta H \). Now we have \( H' \) and \( T' \), where

\[
H' = \begin{bmatrix} H \\ \Delta H \end{bmatrix} \quad \text{and} \quad T' = \begin{bmatrix} T \\ \Delta T \end{bmatrix}
\] (2.9)

According to the matrix multiplication operator, Equations (2.7) and (2.8) become

\[ H'^T H' = \begin{bmatrix} H^T \\ \Delta H \end{bmatrix} \begin{bmatrix} H \\ \Delta H \end{bmatrix} = \begin{bmatrix} H^T & \Delta H^T \end{bmatrix} \begin{bmatrix} H \\ \Delta H \end{bmatrix} = H^T H + \Delta H^T \Delta H \] (2.10)

\[ H'^T T' = \begin{bmatrix} H^T \\ \Delta H \end{bmatrix} \begin{bmatrix} T \\ \Delta T \end{bmatrix} = \begin{bmatrix} H^T & \Delta H^T \end{bmatrix} \begin{bmatrix} T \\ \Delta T \end{bmatrix} = H^T T + \Delta H^T \Delta T \] (2.11)
As their approach defines \( U = H^T H \) and \( V = H^T T \), Equation (2.10) and (2.11) are presented as
\[
U' = U + \Delta U \quad \text{and} \quad V' = V + \Delta V
\] (2.12)
And Equation (2.6) becomes
\[
\hat{\beta} = U^{-1} V
\] (2.13)

2.2.1 One-Class ELM

ELM usually deals with the data with two or more classes, while [84] derived that ELM has the ability for one-class classification.

By examining Equations (2.1) and (2.3), when a target class \( T \) is only one class, namely all 1s in \( T \), \( \beta \) becomes a linear approximation mapping from \( H \) to \( T \), which geometrically is a hyper plane approximation [38]. Then a distance of any point (a sample) to the hyper plane constructed by the non-kernel ELM is defined as
\[
d = |H \cdot \beta - T|
\] (2.14)[46]. By measuring the distance, it shows that ELM is able to detect anomaly samples based on the one-class training.

2.3 Dataset preprocessing

To evaluate the performance of our proposed methods in this thesis, the MIMIC-II [64] dataset from MIT Physiobank archive is used, which contains various types of vital sign data from different patients. The dataset is preprocessed before the validation of our proposed method for generating the labels and features from different ranges of vital signals as well as for filtering the noise from the original dataset. For more details of the preprocessing techniques, readers are referred to [23]. MATLAB version 9.1.0.441655 (R2016b) [51] are used to process the medical data.

In our processed dataset, there are 180 features in a sample which is classified into five labels. Before training the proposed ELM-SM we map all feature values into the range \([-1, 1]\) by the min-max scaling. The features of a sample and labels are described as follows.
2.3.1 Sample

Each sample of our processed dataset contains 6 biosignals: Heart Rate (HR), Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), Mean Blood Pressure (MBP), Respiratory Rate (RR) and Blood Oxygen Saturation (SPO$_2$), as shown in Figure 2.2. The length of each biosignal is 60 minutes which is again sampled by 10-minute observation windows. Therefore we have six subsamples from each biosignal. Five features are extracted from each subsample. The features are mean, standard deviation, median, the number of increasing and decreasing trends. So one sample contains 180 features (6 biosignals * 6 subsamples * 5 features).

![Figure 2.2: An example of a sample in 60 minutes. There are six biosignals in a sample, which is sampled with 10-minute observation windows. In every observation window for a biosignal, there are five features extracted.](image)

2.3.2 Labels

In this thesis, we focus on predicting five different events using our proposed system, which are listed in Table 2.1. If the vital signals are not in the normal (expected) ranges, a clinical event is identified. For instance, NNNN represents that all vital signals are within the normal ranges whereas the other four clinical events (THTH, BHTH, TTTH, THBH) represent ranges
of vital signals that are out of the expected ranges [22] (as shown in Table 2.1).

Table 2.1: The targeted events for evaluation, their labels and generalized normal value of vital biosignals

<table>
<thead>
<tr>
<th>Event</th>
<th>Label</th>
<th>Vital sign</th>
</tr>
</thead>
<tbody>
<tr>
<td>All values in normal range</td>
<td>NNNN</td>
<td></td>
</tr>
<tr>
<td>Simultaneous Tachycardia, Hypotension, Tachypena and Hypoxia for more 30 minutes</td>
<td>THTH</td>
<td>HR HIGH ( \geq 100 ) BP LOW ( SBP \leq 90 ) ( DBP \leq 60 ) or ( MBP \leq 70 ) RR HIGH ( \geq 17 ) SPO( \leq 93% )</td>
</tr>
<tr>
<td>Simultaneous Bradycardia, Hypotension, Tachypena and Hypoxia for more 30 minutes</td>
<td>BHTH</td>
<td>HR LOW ( \leq 60 ) BP LOW ( SBP \leq 90 ) ( DBP \leq 60 ) or ( MBP \leq 70 ) RR HIGH ( \geq 17 ) SPO( \leq 93% )</td>
</tr>
<tr>
<td>Simultaneous Tachycardia, Hypertension, Tachypena and Hypoxia for more 30 minutes</td>
<td>TTTH</td>
<td>HR HIGH ( \geq 100 ) BP HIGH ( SBP \geq 120 ) ( DBP \geq 80 ) or ( MBP \geq 105 ) RR HIGH ( \geq 17 ) SPO( \leq 93% )</td>
</tr>
<tr>
<td>Simultaneous Tachycardia, Hypotension, Bradypena and Hypoxia for more 30 minutes</td>
<td>THBH</td>
<td>HR HIGH ( \geq 100 ) BP LOW ( SBP \leq 90 ) ( DBP \leq 60 ) or ( MBP \leq 70 ) RR LOW ( \leq 12 ) SPO( \leq 93% )</td>
</tr>
</tbody>
</table>
Chapter 3

Fast and P2P Learning Healthcare System

As discussed in Chapter 1, state-of-the-art healthcare systems can benefit from ML technologies by detecting symptoms of different diseases according to time-period vital biosignals which are collected from various wearable devices. By receiving those discrete data samples, a healthcare system is trained over time (e.g., an hour or half an hour) to fit new data and thus is able to diagnose future medical records with higher accuracy. However, even leveraging the processing power of the Cloud, most machine-learning algorithms still consume large amounts of time in training, which introduces new challenges in the efficient and real-time training process.

In addition, medical institutions usually employ private Cloud services to store data and accelerate training processes [1], but sharing the dataset with other institutions has been rarely considered. The healthcare systems in other institutions can improve their diagnostic accuracy by learning the shared data efficiently (the more data are available, the more accurate diagnosis a healthcare system can provide).

In addressing these issues and in relation to RQ-1 as presented in Section 1.3, we enhance the healthcare system by sharing knowledge in order to reduce training time and improve the diagnostic accuracy. ELM [37] and the MD5 hash algorithm [61] are adopted in our system. Concretely, ELM is used for generating knowledge named semi-model from raw training dataset and the MD5 hash algorithm calculates the version of a semi-model which can avoid repeated
Motivation and Contribution

The rest of this chapter is organized as follows. Section 3.1 highlights the motivations and contributions of this work. Section 3.2 and 3.3 describe the related works and the detail methodology of our proposed work respectively. Section 3.4 discusses the experimental analysis and finally, Section 3.5 summarizes the chapter.

3.1 Motivation and Contribution

Two limitations of the centralized model are targeted in this work, explained in Section 1.1, which are intensive central dependence and imbalanced sharing and learning process. In order to overcome the issues of centralized model in terms of sharing the raw data, we introduce a P2P learning model for Cloud-based healthcare system where one medical institution extracts a semi-model (lightweight output during the training process) from patients’ biosignal training data up to a certain stage and then transmits the semi-model to other neighbor peers. A comparison between the traditional centralized model and our proposed decentralized P2P system is shown in Figure 1.3. The main advantage of our proposed model lies in eliminating dependence on the central server.

Our proposed healthcare system aims to perform fast and accurate diagnoses with patients’ biosignal data. We use ELM with *semi-model (ELM-SM)* as the main classifier for prediction. ELM [37] is a SLFN which does not need to tune the hidden layer and shows excellent efficiency in terms of computational time and great success in medical diagnosis [36][42][67]. In addition, many existing studies provide a fantastic running application for the remote healthcare system [59][43]. Our proposed system focuses on the P2P learning process and can be integrated within existing applications to monitor patients’ health conditions.

The main contributions of this chapter are summarized as follows.

- We present a P2P learning model which allows medical institutions to share and learn the semi-model rather than the raw data, which reduces the computational overhead of different institutions in learning the raw biosignal. More specifically, the lightweight
semi-model is extracted from enormous training samples based on the learning process of ELM, providing faster learning and less sharing time than that using raw data.

- We introduce a new method to filter the semi-models based on their versions so that the healthcare institutions avoid learning the same semi-models. This means we eliminate the redundant sharing of semi-models among the healthcare institutions. This version is generated by the MD5 algorithm [61] and represents the unique dataset.

- Our extensive experimental analysis from publicly available biosignal data presents that the proposed P2P learning model is efficient and effective in terms of learning time and size of data exchanged among peers.

### 3.2 Related Work

Nowadays various research works have been providing promising solutions on smart healthcare systems; however they suffer from some limitations. The [74] analyzes different challenges in healthcare systems where the big data issue is addressed by adopting Cloud platforms used in many studies. For example, [22] introduces a home-based monitoring system with Hidden Markov Model (HMM). The [32] develops a healthcare framework to diagnose patients’ states using video and audio signals, while [33] enhances the monitoring system by introducing the patients’ position. The [59] and [43] propose a future application of a ubiquitous healthcare system with different bio-signals. The [59] employs Cloud, biosensor and smartphone to monitor patients with chronic lung disease. The authors develop an application on the iPhone Operating System (iOS) to show the analysis result from the Cloud. Similar to [59], the [43] shows a healthcare system with a smartphone application and health information management server. The smartphone application collects the data and displays the state of a patient’s health determined by the server. But these systems are isolated and do not consider sharing their knowledge to benefit other medical institutions.

In terms of privacy issues, the [74] and [50] adopt encryption-based techniques to develop a secure transmission protocol, and [52] uses the blockchain to share patients’ information in a distributed way. These methods focus on the privacy preserving domain and do not consider
Related Work

P2P learning, which is mainly proposed in our thesis. In addition, encryption-based techniques consume a large volume of time to encrypt and decrypt the patients’ records. Moreover, sharing the encrypted raw data is inefficient and whether different medical institutions have the same right to use the raw data from patients’ records also need to be considered.

Many different machine-learning techniques such as HMM [24][56][70], Support Vector Machine (SVM) [14] and Neural Network (NN) [29] are widely used in clinical diagnoses or support systems to detect abnormal events and symptoms of disease. However, majority of them are computationally expensive. Table 3.1 shows an overall comparison in terms of computational cost among three related works. Even though these studies used different datasets and experimental settings, we can notice that the classifiers used in these systems (HMM, convolution neural network (CNN) and Radial Basis Function Network (RBFN)) take over 15 minutes for training, which will become a serious and critical issue with increasing samples and complicated classifiers for learning in big data.

Table 3.1: Time consumption of related works

<table>
<thead>
<tr>
<th>Related Work</th>
<th>Classifier</th>
<th>Number of samples</th>
<th>Machine</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[23]</td>
<td>HMM</td>
<td>600</td>
<td>m3.xlarge in Amazon (4 CPUs and 8GB RAM)</td>
<td>30 min</td>
</tr>
<tr>
<td>[13]</td>
<td>CNN</td>
<td>606</td>
<td>2-core CPU and 8GB RAM</td>
<td>over 25 min</td>
</tr>
<tr>
<td>[42]</td>
<td>RBFN</td>
<td>70,000</td>
<td>Intel Quad-core 2.4GHz and 2GB RAM</td>
<td>over 15 min</td>
</tr>
</tbody>
</table>

The [36] addressed the efficiency issues by proposing an ELM based classifier to provide a good generalization performance in medical diagnostic systems [42][67]. Compared to other machine-learning algorithms, ELM is faster in terms of learning since it does not need to calculate the parameters of hidden layers iteratively. Table 3.2 shows another comparison among different research works in terms of their accuracy in detecting different clinical events with various classifiers in healthcare diagnostic systems. We can observe that ELM is the most effective in terms of classifying different clinical events in biosignal datasets. Moreover, in order to meet the requirements of processing big data, different Cloud providers such as Microsoft and Amazon offer data services, which store and search patients’ records [1]. However, these
### Table 3.2: Recent research works in healthcare based on machine-learning algorithms

<table>
<thead>
<tr>
<th>Related Work</th>
<th>Classifier</th>
<th>Dataset</th>
<th>Features</th>
<th>Clinical events</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>[23]</td>
<td>HMM</td>
<td>MIMIC-II</td>
<td>Heart rate, respiratory rate, blood oxygen saturation, systolic blood pressure</td>
<td>5</td>
<td>97.7%</td>
</tr>
<tr>
<td>[14]</td>
<td>SVM</td>
<td>Customized dataset from Oxford Cancer Hospital</td>
<td>Heart rate, respiratory rate, blood oxygen saturation, systolic blood pressure</td>
<td>2 (Normal or not)</td>
<td>96%</td>
</tr>
<tr>
<td>[29]</td>
<td>Neural Network</td>
<td>MIMIC-II</td>
<td>arterial blood pressure</td>
<td>2 (Acute hypotensive episode or not)</td>
<td>94%</td>
</tr>
<tr>
<td>[42]</td>
<td>ELM</td>
<td>MIT-BIH Arrhythmia</td>
<td>ECG</td>
<td>6</td>
<td>98.72%</td>
</tr>
<tr>
<td>[68]</td>
<td>kNN</td>
<td>Cleveland Heart Disease Dataset</td>
<td>13 attributes</td>
<td>7</td>
<td>97.4%</td>
</tr>
<tr>
<td>[13]</td>
<td>CNN</td>
<td>Customized dataset from central China in 2013-2015</td>
<td>79 features</td>
<td>2 (Cerebral infarction or not)</td>
<td>94.8%</td>
</tr>
<tr>
<td>[66]</td>
<td>Decision Tree</td>
<td>Pima Indian Diabetic Dataset</td>
<td>8 features</td>
<td>2 (Diabetes Type 2 or not)</td>
<td>80%</td>
</tr>
<tr>
<td>[66]</td>
<td>Bayes Net</td>
<td>Pima Indian Diabetic Dataset</td>
<td>8 features</td>
<td>2 (Diabetes Type 2 or not)</td>
<td>74%</td>
</tr>
</tbody>
</table>

Related Work

Table 3.2: Recent research works in healthcare based on machine-learning algorithms

<table>
<thead>
<tr>
<th>Related Work</th>
<th>Classifier</th>
<th>Dataset</th>
<th>Features</th>
<th>Clinical events</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>[23]</td>
<td>HMM</td>
<td>MIMIC-II</td>
<td>Heart rate, respiratory rate, blood oxygen saturation, systolic blood pressure</td>
<td>5</td>
<td>97.7%</td>
</tr>
<tr>
<td>[14]</td>
<td>SVM</td>
<td>Customized dataset from Oxford Cancer Hospital</td>
<td>Heart rate, respiratory rate, blood oxygen saturation, systolic blood pressure</td>
<td>2 (Normal or not)</td>
<td>96%</td>
</tr>
<tr>
<td>[29]</td>
<td>Neural Network</td>
<td>MIMIC-II</td>
<td>arterial blood pressure</td>
<td>2 (Acute hypotensive episode or not)</td>
<td>94%</td>
</tr>
<tr>
<td>[42]</td>
<td>ELM</td>
<td>MIT-BIH Arrhythmia</td>
<td>ECG</td>
<td>6</td>
<td>98.72%</td>
</tr>
<tr>
<td>[68]</td>
<td>kNN</td>
<td>Cleveland Heart Disease Dataset</td>
<td>13 attributes</td>
<td>7</td>
<td>97.4%</td>
</tr>
<tr>
<td>[13]</td>
<td>CNN</td>
<td>Customized dataset from central China in 2013-2015</td>
<td>79 features</td>
<td>2 (Cerebral infarction or not)</td>
<td>94.8%</td>
</tr>
<tr>
<td>[66]</td>
<td>Decision Tree</td>
<td>Pima Indian Diabetic Dataset</td>
<td>8 features</td>
<td>2 (Diabetes Type 2 or not)</td>
<td>80%</td>
</tr>
<tr>
<td>[66]</td>
<td>Bayes Net</td>
<td>Pima Indian Diabetic Dataset</td>
<td>8 features</td>
<td>2 (Diabetes Type 2 or not)</td>
<td>74%</td>
</tr>
</tbody>
</table>

solutions focus on leveraging the advantage of Cloud data storage rather than computation capacity. For Cloud computing, a centralized algorithm such as MapReduce is typically used in varieties of studies. Among them, distributed training methods with ELM [82][6] and HMM [23] using MapReduce have been proposed to accelerate the training process with the computational power of Cloud. In addition, some ensemble based models [18][76][11] based on MapReduce have been also adopted to avoid the over-fitting problem and to improve the accuracy of the classifiers.

Note that the above-described methods depend strongly on pivotal central servers and have little consideration for the cost of data transmission. The MapReduce framework [16] focuses on enhancing the processing speed for a single user (master) by taking advantage of distributed computing or Cloud computing (slaves) paradigm. This means every medical institution can hire any Cloud service with MapReduce to deal with a lot of patients’ records effectively in our scenario. But under this framework, only the master has the complete knowledge for the classification task. Moreover, it does not consider how to share knowledge among different masters for improving the prediction accuracy of all masters, which means that institutions
are unlikely to share their own learned knowledge to help others. Sharing raw data among masters is also not an efficient method due to the large volumes of new data collected in a medical institution every day.

In order to overcome the drawbacks described in the existing works, we were inspired to develop a novel learning structure by employing the P2P structure with ELM-SM. Compared to existing works, our proposed system has fully distributed learning ability which can benefit other systems connected in a P2P network. The proposed semi-model based method can reduce the size of raw data and speed up the sharing and learning process.

### 3.3 Methodology

In this section, we propose our P2P learning model for a collaborative healthcare community based on ELM-SM. The aim is to build a smart healthcare system in a P2P architecture, which is fast and efficient in terms of learning and exchanging information. The goal is achieved by generating semi-models from raw data, followed by sharing them with Cloud-based medical institutions. We regard every medical institution as a peer in our model.

#### 3.3.1 System scenario and overview

The overview of the proposed system is shown in Figure 3.1. We consider a scenario that an elderly patient lives alone at home and his/her physiological conditions are monitored by a remote healthcare system in a medical institution continuously. Various wireless sensors are attached to a patient’s body and send the biosignal data to a wearable smart device (e.g. smart watch). The device transmits the data to the private Cloud hired by the medical institution for processing. The Cloud runs different algorithms for data cleaning, ELM-SM training and knowledge discovery. Then the trained ELM-SM is used to measure future clinical states based on newly received data from patients. The clinicians (e.g. doctors and nurses) in the medical institution can make diagnostic decisions according to a patient’s clinical states and notify the patient properly.

We describe the overview of our system into two parts: (1) data collection and prediction in
3.3.1.1 Data collection and prediction in a single peer

In our proposed system shown in Figure 3.1, time-sequential data from wearable body sensors of a patient are continuously sent to his/her portable IoT devices. The devices then transmit those physiological data to a specific peer for further processing. The data are stored in the Cloud and are used to train the healthcare system. In our system, training dataset contains vital biosignal records of patients and their labels (clinical events). In addition, ELM-SM installed in every peer has the ability to detect anomalous clinical events of newly received
data from IoT devices and then the peer can notify the patients for any further actions. For instance, the medical institution can automatically call the patients to make an appointment with doctors if it finds any abnormalities in the records.

### 3.3.1.2 Data sharing among the peers

In our proposed model, every peer has a certain number of neighbor peers connecting with it directly, which forms a P2P network with a specific protocol. For example, Figure 3.1 shows that there are four neighbor peers connecting to the peer in the middle of the image. A peer extracts a semi-model from the training dataset and shares the semi-model with its neighbors. An example of sharing process between a peer and its neighbor peer is shown in Figure 3.2. The semi-model is defined as lightweight output during the training process of ELM, which is used to calculate the final model (the generation process of a semi-model is described in the next section). Every peer filters the learned semi-models from received semi-models and stops sharing these learned semi-models to its neighbor peers. Generally, a semi-model is smaller than raw training dataset, which makes our proposed system more efficient in the data-sharing process.

### 3.3.2 ELM-SM

In this section we describe the process of generating a semi-model, updating ELM-SM with the newly generated semi-model from a neighbor peer and filtering different versions of semi-models to avoid repeated learning in a peer.
Figure 3.2: Example of data sharing. The sharing process happens between a peer and its neighbor peer. Peer A shares a new semi-model with Neighbor Peer B. Peer B shares it with Neighbor Peer C and Peer C shares it with Neighbor Peer D.

Figure 3.3: The process of proposed semi-model generation based on ELM. A training raw dataset is passed through the input layer and a certain calculation is performed with random input weights in the hidden layer. The output from the hidden layer contains two components (C1 and C2), which forms a semi-model. A semi-model is used to calculate the final model in a learner. More specifically, C1 is generated from the symmetric matrix ($H^TH$).
3.3.2.1 The generation of semi-model

Figure 3.3 shows the process of generating a semi-model. There are two main components of a semi-model. We name them as C1 and C2.

- C1: This component consists of the elements from resultant matrix \((H^T H)\) generated by Equation (2.7). According to the definition of the symmetric matrix, we have \(u_{ij} = u_{ji}\) in

\[
H^T H = \begin{bmatrix}
u_{11} & \cdots & u_{1k} \\
\vdots & \ddots & \vdots \\
u_{k1} & \cdots & u_{kk}
\end{bmatrix}_{K \times K}
\]  

(3.1)

So, C1 is defined as \([u_{ij}]\) shown in below equation where \(i \leq j\).

\[
C1 = \begin{bmatrix}
u_{11} & u_{12} & \cdots & u_{1k} \\
u_{21} & u_{22} & \cdots & u_{2k} \\
\vdots & \vdots & \ddots & \vdots \\
u_{k1} & u_{k2} & \cdots & u_{kk}
\end{bmatrix}
\]  

(3.2)

And we can easily transform C1 to \(H^T H\) with the number of hidden neurons \(K\). In addition, we can observe that the size of \(H^T H\), as well as C1, depends on the number of neurons in the hidden layer.

- C2: This component is regarded as \(H^T T\) generated from Equation (2.8) with the hidden layer output matrix \(H\) and data label vectors \(T\). According to Equation (2.8), the size of C2 depends not only on the number of hidden neurons but also on the number of output neurons.

Finally, the learner calculates the model based on the semi-model with Equation (2.6). Figure 3.4 shows an example of two components C1 and C2 in a semi-model. The semi-model is generated from ELM with 3 hidden neurons and 5 targeted outputs.

The proposed semi-model generation process is shown in Algorithm 1. Every peer is required to have same random input weights (\(w\) and \(b\) in Equation (2.1)) in Step 6. At the beginning, the training sample matrix \(X\) and their label matrix \(T\) are passed into GENERATE function in Step 5. The label matrix contains label vectors of related samples. The value of the elements in a label vector is 1 when the output neuron belongs to the sample class and the rest are \(-1\). For example, a label vector \((-1, -1, 1, -1, -1)\) means that there are 5 targeted
classes and this vector belongs to the third class. The hidden output matrix \((H)\) is calculated based on Equation (2.1) in Step 6. In Step 7 and 8, we calculate \(M = H^T H\) and \(C_2 = H^T T\) which are shown in Equation (2.7) and (2.8) respectively. Finally, we generate the semi-model \(C_1\) and \(C_2\) which can be shared with other peers.

**Algorithm 1** Proposed semi-model generation algorithm

1: **Input:** \(X\) and \(T\)

2: **Output:** \(C_1\) and \(C_2\)

3: **Initialize:** \(w, b, H, M, C_1, C_2\)

4: 

5: **procedure** `generate(\(X, T\))`

6: \[H \leftarrow X \ast w + b\] \(\triangleright\) Equation (2.1)

7: \[M \leftarrow H^T \ast H\] \(\triangleright\) Equation (2.7)

8: \[C_2 \leftarrow H^T \ast T\] \(\triangleright\) Equation (2.8)

9: \[C_1 \leftarrow \text{Elements of } M \text{ where } i \leq j\] \(\triangleright\) Equation (3.2)

10: **return** \(C_1, C_2\)

11: **end procedure**

### 3.3.2.2 Semi-model version based on MD5

The MD5 value calculated from a semi-model is used to speed up the process of model comparison, which is applied in data filter mentioned in the next section. Even though each component of a semi-model is smaller than a large-scale raw training dataset, comparing two different components requires high computation cost. To make the process faster we apply the MD5 hash algorithm which is commonly used as a checksum to verify the data integrity. The change of even a bit in the data can result in the difference of the 128-bit hash code. Since the number of output neurons is usually much less than the number of hidden neurons, it implies that \(C_2\) also becomes much smaller than \(C_1\). Thus, \(C_2\) is used to calculate the MD5 hash code. An example of a semi-model version generated by the MD5 hash algorithm is shown in Figure 3.4. Moreover, only first six digits after the decimal point in every element of a matrix are used to calculate the hash value because of the floating point precision in computer numerical calculations.
3.3.2.3 Update process of ELM-SM in a peer

Figure 3.5 shows the process of updating the semi-model in a peer. In our P2P system, every peer receives semi-models from its neighbor peers in an uncertain time and order. But the peer only consumes a new semi-model in each update process. Before processing a semi-model, a peer filters the learned semi-model and also stops sharing this semi-model with its neighbors. Learned semi-models are detected by the data filter with a version table. The table contains versions of learned semi-models, which is generated based on the MD5 hash algorithm. Then each component of this new semi-model is merged into the original semi-model in ELM-SM. Finally, the updated semi-model is used to calculate the new classifier based on Equation (2.6), which is shown in Figure 3.3.
Figure 3.5: The process of updating the semi-model in a peer. When receiving a semi-model from a neighbor peer, a peer filters and discards the learned semi-model based on its own version table. The version table contains the MD5 hash code of learned semi-models. Then components of own semi-model added with that of the new unlearned semi-model are used to calculate the updated classifier.

The update process is introduced in Algorithm 2. The update function takes new $C_1$ and $C_2$ generated by Algorithm 1 from other peers as input. The local components of the semi-model are represented by $M_1$ and $M_2$ respectively. At the beginning, the MD5 version is calculated in Step 12. Then the data filter is applied to the version, justifying whether ELM-SM has already learned the knowledge based on its version table. If the version does not exist, ELM-SM in the peer is updated with the new semi-model. Firstly, $M_1$ and $M_2$ are added with new $C_1$ and $C_2$ respectively in Step 15 and 16. Then $M_1$ is transformed into the matrix $M$, which is used to calculate a new classifier based on Equation (2.6) in Step 19. The new version is also recorded in the version table in Step 18.
Algorithm 2 Proposed update algorithm based on semi-models in a peer

1: **Input:** $C_1$ and $C_2$
2: **Output:** $\beta$ of ELM
3: **Initialize:** $M_1, M_2, VersionTable$
4: 5: **procedure** DATA\_FILTER($newVersion$) 6: if $VersionTable$ has $newVersion$ then 7: return true 8: else 9: return false 10: end if
11: end procedure
12:
13: **procedure** UPDATE($C_1, C_2$) 14: $newVersion \leftarrow$ Calculate MD5 value of $C_2$
15: if DATA\_FILTER($newVersion$) then 16: return
17: end if
18: $M_1 += C_1$
19: $M_2 += C_2$ \hspace{1cm} $\triangleright$ Equation (2.11)
20: Transform $M_1$ into matrix $M$
21: Add $newVersion$ into $VersionTable$
22: return $M^{-1} \ast M_2$ \hspace{1cm} $\triangleright$ Equation (2.6)
23: end procedure

3.4 Experimental evaluations and results

To evaluate the performance of our proposed method, we preprocess the data from the MIMIC-II [64] dataset and obtain our training and testing datasets. The process of data preprocessing is described in Section 2.3. Recall that our datasets contains 180 features which are extracted from six biosignals and are labeled into 5 events.

The rest of our experimental section is organized as follows. Firstly, we analyze the performance of ELM-SM in terms of accuracy and training time. Then we analyze the performance of the proposed P2P learning system in terms of the size of initial learning peers and the size of the overall network. To test the performance we use three different metrics:

- **Step:** The required number of hops until all peers are updated with the same model.
• Data exchange: The total number of semi-models that are sent to and received by a peer.

• Data size: Size of a semi-model and its raw data being shared.

In order to evaluate the proposed ELM-SM, we calculate sensitivity, specificity and accuracy as,

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \times 100 \tag{3.3}
\]

\[
\text{Specificity} = \frac{TN}{TN + FP} \times 100 \tag{3.4}
\]

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{3.5}
\]

where TP, TN, FP, and FN are True Positive, True Negative, False Positive, and False Negative respectively.

### 3.4.1 Accuracy of ELM-SM

Our overall dataset contains 6681 samples in total. We split the samples into training and testing sets which contain 70% (4675 samples) and 30% (2006 samples), respectively. The total number of samples for training and testing phases are shown in Tables 3.3 and 3.4 respectively in terms of each event.

<table>
<thead>
<tr>
<th>Event</th>
<th>NNNN</th>
<th>THTH</th>
<th>BHTH</th>
<th>TTTH</th>
<th>THBH</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of samples</td>
<td>1819</td>
<td>98</td>
<td>400</td>
<td>198</td>
<td>2160</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event</th>
<th>NNNN</th>
<th>THTH</th>
<th>BHTH</th>
<th>TTTH</th>
<th>THBH</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of samples</td>
<td>781</td>
<td>42</td>
<td>172</td>
<td>85</td>
<td>926</td>
</tr>
</tbody>
</table>

Our proposed ELM-SM shows the best accuracy of 95.47% among 10 different results based on 500 hidden neurons and the sigmoid activation function. Table 3.5 shows overall sensitivity, specificity and accuracy for each event. From this table we can observe that the event TTTH shows poor performance in terms of sensitivity (to detect the true event correctly). We believe this is because there are insufficient samples for this particular event (TTTH).
Although another event THTH does not contain the enough number of samples compared to other events, it shows a reasonable performance of 92.86% to detect the true event correctly. Excepting these two events, other events show excellent performance with over 98% in terms of the sensitivity. In terms of specificity, all events show more than 99% accuracy to exclude the false events correctly, except NNNN which still shows very good accuracy of 93.88%.

Table 3.5: The performance measure of each event using ELM-SM with 500 hidden neurons and the sigmoid activation function. Here 95.47% is obtained.

<table>
<thead>
<tr>
<th></th>
<th>NNNN</th>
<th>THTH</th>
<th>BHTH</th>
<th>TTTH</th>
<th>THBH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity</td>
<td>98.72%</td>
<td>92.86%</td>
<td>98.84%</td>
<td>1.53%</td>
<td>99.57%</td>
</tr>
<tr>
<td>Specificity</td>
<td>93.88%</td>
<td>99.90%</td>
<td>99.73%</td>
<td>99.84%</td>
<td>99.45%</td>
</tr>
<tr>
<td>Accuracy</td>
<td>91.15%</td>
<td>95.12%</td>
<td>97.14%</td>
<td>81.25%</td>
<td>99.35%</td>
</tr>
</tbody>
</table>

3.4.2 Comparison between traditional ELM and ELM-SM in terms of training time

In this part, we compare time consumption between traditional ELM and proposed ELM-SM in the incremental training process. We split the whole dataset shown in Table 3.3 into 5 different datasets: NNNN set, THTH set, BHTH set, TTTH set and THBH set. These datasets are used to train ELM and ELM-SM in a certain order. For example, as shown in Figure 3.6, in Learning Round 1, ELM and ELM-SM learn from NNNN set individually. In Round 2, ELM needs to calculate a new model with NNNN and THTH samples, while ELM-SM learns from a semi-model generated from THTH samples. When new training datasets arrive gradually, ELM consumes more and more time to learn because there are more training samples. However, ELM-SM has a stable and better performance because redundant calculations are eliminated by the semi-model.
Experimental evaluations and results

3.4.3 Performance of P2P learning

In this section, we analyze the performance of our proposed P2P learning model in terms of the network size, initial learning peers and the maximum number of semi-models that are exchanged among the peers. In the following, we explain the experimental setup used to test the performance of the proposed learning system.

3.4.3.1 Experimental Setup for the proposed P2P learning system

The proposed P2P learning system is implemented by the AKKA toolkit [63] based on SCALA [55], a distributed programming language. Under this toolkit we briefly explain the simulator implementation, the adopted network topologies and the convergence criteria of the P2P learning process.
Simulator implementation

In our simulator, every peer has three Actors\(^1\):

- The network overlay actor selects the neighbor peers and shares semi-models from the classifier actor with them. In addition, it sends semi-models to the classifier actor for further processing.

- The classifier actor performs the classification and filters semi-models from the network overlay actor. Then it transmits unknown semi-models to the training actor. In addition, it shares these semi-models with other neighbor peers through the network overlay actor.

- The training actor updates the model with semi-models and transmits the new model back to the classifier actor.

In particular, all peers are deployed with ELM-SM which is initialized with 500 neurons in the hidden layer. All ELM-SMs have the same parameters – random input weights \((w \text{ and } b\) in Equation (2.1)) and the sigmoid activation function – as others in the whole network.

Network Topologies

Linear and 3D network topologies\(^2\) are adopted in our proposed P2P learning model. The Linear and 3D networks are similar to the Ring and 3D Torus topologies respectively. In our experiments, every peer has a unique index starting from zero. The Linear network organizes all peers into a ring where each peer has two neighbor peers, while there are six neighbors for every peer in the 3D network topology.

---

\(^1\)Actors are the universal primitives of concurrent computation and maintain their own state. They can easily be deployed in Cloud.

\(^2\)Note that a real protocol for building and maintaining the network dynamically is beyond the scope of this thesis. Also, we assume that the network is initiated based on the definition of the network structure and is not changed during the whole experiments.
Figure 3.7: Example of P2P learning process of the proposed system in 3D network topology. There are 20 peers in total with three initial learning peers in the network. Different colors represent different versions of ELM-SM in a peer. In step 0, three initial learning peers shown in different colors have already learned from raw datasets and while the black peers have not learned anything. In each step, peers receive and learn the semi-models from their neighbor peers, which changes the color. It takes 15 steps for all peers to update their own model to the same ELM-SM, when the total number of peer is 20 and the number of initial learning peers is 3 in the 3D network topology.
Convergence Criteria

The peers keep sharing their newly learned semi-models with their neighbors. Ideally, if a constant number of semi-models are shared in the system, all peers will finally have an identical model version within a certain time of exchanging the semi-models, which is the main criterion of convergence. Figure 3.7 shows an example of the converging process in the 3D network with 20 peers. From this figure, we can observe that every peer has six neighbors and different colors represent different model versions in a peer. For example, black represents that the corresponding peer has not learned anything yet. Initially, three peers start training with a specific unique dataset separately. After 15 steps (Figure 3.7(p)), all peers in the network finish the learning process and have the same ELM-SM.

Performance Metric

Recall that we consider three different metrics to test the performance of the P2P learning system, namely steps of convergence, data size and the maximum number of exchanging semi-models in a peer. Particularly, in order to evaluate the performance of P2P learning more practically, we count the steps of convergence as a metric instead of required time. This is an advantage as we do not rely on the capacity or power of any machine or network since different machines and networks may provide different processing and transmitting rates, which usually affect the overall performance results. Specifically, we calculate steps as follows. Every peer maintains its own step. When a peer sends a semi-model to its neighbors, the host peer transmits a new step value (its current step + 1) to neighbors. The current step in a neighbor peer is overridden with the biggest value. Otherwise, the neighbor peer discards the received step value if its value is bigger. Figure 3.8 shows an example of step calculation in a peer. We can thus analyze the state of each peer based on its step (shown in Figure 3.7) and calculate the maximum steps among all peers for the system to converge.
Experimental evaluations and results

3.4.3.2 Varying the size of the network

In this part of the analysis, we fix the number of initial learning peers as three and analyze the impact of different network sizes. We record the number of steps required in five different network sizes and two network topologies until all peers have an identical model version. Figure 3.9 shows the result of the required number of steps in terms of varying the size of P2P learning network (10, 100, 1000, 10,000 and 100,000 peers). The required number of steps is shown in log scale since the results of the Linear network is much higher than the 3D network, especially with 100,000 peers. In the beginning, the training dataset we used in testing ELM-SM above is split equally into three shards, which are fed randomly and separately to three peers.
Experimental evaluations and results

Figure 3.9: Required steps (shown in log scale) in terms of varying the size of the P2P network. Three random peers are fed with the training data at the initial stage.

From Figure 3.9, we can observe that it takes 8 steps for 3D network topology to converge when the size of the network is 10. The required steps increase to 26 with increasing the network size from 10 to 100. Finally, the 3D and Linear network require 120 and 51150 steps respectively when the network size is 100,000. These results demonstrate that the required steps are much higher in the Linear network topology than the 3D. More specifically, we can observe that, for the Linear network, the increasing rate of steps is much higher than the 3D (10 times increasing rate).

3.4.3.3 Varying the number of initial learning peers

In our experiment, we adopt a constant size of a network with 100 peers, then analyze 10 different numbers of initial learning peers from 10 to 100 stepping by increment rates of 10. The training dataset is divided into shards equally based on the number of initial learning peers. For example, the training dataset with 4675 samples is split into 10 shards, each of
which has 468 samples except for the last one with 463. All initial learning peers are randomly selected and fed with a unique shard. Then they generate their own semi-model and share it across the whole network, so that other peers can improve their classifier gradually by receiving other's semi-models. As shown in Figure 3.10, with more initial learning peers in the network, both Linear and 3D network require more steps until the system converges. When the number of initial learning peers is less than 40 in 100, the 3D network requires fewer steps than the Linear network. More specifically, increasing the number of initial learning peers also increases the number of steps for the Linear network from 136 to 307. The required steps for the 3D network increase from 53 to 570 (more than 10 times increasing rate) when the number of initial learning peers increases from 10 to 100.

![Figure 3.10: Required steps in terms of varying the number of initial learning peers. There are overall 100 peers in the P2P network.](image)

**3.4.3.4 Communication cost in terms of exchanging the semi-models**

In this analysis, we initialize the number of network peers with 100 and calculate the maximum number of semi-model packets that are exchanged (sent and received) with a peer during the learning process. A peer does not return the same semi-model which it received from its
neighbors. For example, peers A and B are neighbors of each other. If A received a new semi-model from B, A does not transmit the semi-model back to B. Figure 3.11 shows the number of exchanged semi-model packets in terms of initial learning peers. From this figure it is clear that increasing the number of initial learning peers affects the number of the semi-model packets that are exchanged or transmitted. Specifically, the number of received packets is slightly higher than the number that is sent since some semi-models are used to identify that a peer has learned all new data in the system and stops the broadcast process. Peers in the 3D structure requires five times as many packets to exchange than those in the Linear structure.

![Figure 3.11: The maximum number of semi-model packets sent and received in a peer during the learning process. The size of each P2P network is 100.](image)

### 3.4.4 Comparison of data size between the semi-model and raw data

In order to analyze the comparison of data size, we divide the training dataset randomly into three shards, which contain 1338, 1338 and 1339 samples respectively. Compared to 240,840 (1338x180) or 241,020 (1339x180) elements in the raw training dataset containing 180 features in one sample, there are only 128,000 elements (125,500 elements in $C1$ and 2500 elements in $C2$) in each semi-model with 500 neurons in the hidden layer and 5 targeted classes. So,
compared to the raw dataset, the data volume of the semi-model becomes 50% less, which makes it more efficient to transmit. The number of data elements between the semi-model and raw data are compared in Figure 3.12. In this figure, every horizontal red line represents the size of raw data with 180 features per sample grouped in a shard with 500, 1000, 1500, 2000, 2500 and 3000 samples respectively. We can observe that the size of a semi-model with fewer than 550 hidden neurons (in the horizontal axis) is always smaller than the size of 1000 samples in a dataset. In our scenario, 1000 samples can contain six biosignal records of around 42 (1000/24) patients in 24-hour monitoring.

Figure 3.12: Comparison of data size between the semi-model and raw data in the P2P learning process. The number of targeted classes is 5 in total, shown in Table 2.1 and the number of features in each sample is 180. The data size is represented by the number of data numeral elements in matrices. The bar chart is from the semi-model and lines are from the raw training dataset.
3.5 Summary

In this chapter, we propose a novel P2P learning system for healthcare-related knowledge sharing in order to achieve a collaborative healthcare community in the Cloud environment. More specifically, our contributions and achievements are twofold. First, we eliminate dependence on the central server. Second, any peer or medical institution can share the knowledge which it has already learned from the raw data and can share its knowledge with neighbor institutions. As a result, the neighbor institutions learn the datasets from shared knowledge without spending a huge time on learning from raw datasets. Therefore, instead of transmitting raw biosignal data in traditional centralized learning models, our proposed system transmits the semi-models to its neighbors which makes the system more efficient in terms of learning and sharing the knowledge. The experimental analysis also shows excellent performance in terms of computation and communication costs.

Moreover, our system has distinct privacy advantage because the system transforms the raw dataset into a new random space and the raw information cannot be extracted from a semi-model. We will analyze privacy security of the system in our future work. In addition, we wish to conduct more research on more complex network topologies to improve the performance of the proposed P2P learning system.
Chapter 4

Collaborative ELM (CELM) with A Confidence Interval (CI)

The most important role of the our P2P learning system explored previously in Chapter 3 is to transmit the learned knowledge as semi-model in the network and then update every Machine-Learning (ML) classifier of each peer with these knowledge. There are two main features in the P2P learning system: fully distributed data and classifier [57]. Both medical data and the classifier belong to a specific medical institution and are processed locally. Every medical institution shares their knowledge with other institutions to improve diagnosis accuracy, which makes the ML classifiers collaborative. In the collaborative learning system, a peer needs to have the ability to deal with unknown data labels in both learning and prediction processes, which is not supported in the system shown in Chapter 3. Therefore, Online-Sequential Learning (OS-learning) [47], showing the usual approach of building classifiers and updating them, is considered to be introduced to enhance the P2P learning system.

Concretely, the OS-learning is a capability for continuously learning without retraining the classifier when a new training dataset is fed into the classifier. With the OS-learning ability, the healthcare classifier can be efficiently adapted to new data samples of different diseases. Figure 4.1 shows a comparison of the learning process between a typical classifier (Neural Network [36][29] as an example) with and without the OS-learning ability. The retraining process is shown in Figure 4.1(a) and the continuous/incremental learning (updating) process
is introduced in Figure 4.1(b). The retraining process shows that newly arrived training dataset needs to be combined with the learned datasets as a new training dataset, and then to be used to train a new classifier. Compared to the retraining process, the OS-learning classifier only uses a newly arrived dataset to update itself, which can reduce the redundant calculation of learning.

Figure 4.1: Comparison of two classifiers with and without the OS-learning ability when the classifiers continuously learn from training dataset with an additive data label class. (a) The non-OS-learning classifier learns from a dataset with “BLUE” disease at the beginning. When a new dataset with “RED” disease came, the classifier is retrained with a training dataset containing all samples of “BLUE” and “RED”. The same retraining process is repeated with the dataset with “GREEN” disease. (b) The classifier learns continuously from samples in “BLUE”, “RED” and “GREEN” dataset in order.

In order to adapt OS-Learning to our P2P learning system and in relation to RQ-2, this chapter presents Celm with a Confidence Interval (CI). Similar to semi-model in Chapter 3, we named shared knowledge as knowledge-parameters used to update our proposed classifier. We make basic ELM collaborative based on its training calculation and use a threshold-based algorithm to calculate the confidence interval which might contain the most of true predictions.

The rest of this chapter is organized as follows. Section 4.1 highlights the motivations and contributions of this work. Section 4.2 describes the related works. Then the methodology of our proposed system is explained in Section 4.3, 4.4 and 4.5 respectively. The empirical experiments are conducted in Section 4.6. Finally, a summary is given in Section 4.7.
4.1 Motivation and Contribution

As discussed in RQ-2, in order to make our P2P system more robust, motivations, challenges and contributions of this chapter are described in this section.

One challenge of OS-learning in the P2P system is different learning orders among the peers which may be caused by network latency and computer performance (e.g. the bandwidth and CPU speed). In this case, after learning knowledge with a new Global Data Label (GDL)\(^1\), a peer may consider the labels by its own choice while other peers may have different local labels. In order to facilitate the collaborations among the ML classifiers from different peers, every peer needs to manage data label locally and to update its classifier with newly arrived knowledge from other peers which may contain unknown data labels. We term these labels managed in each peer as Local Data Label (LDL). So a method which performs transformation between GDL and LDL is required to support P2P learning, called as data label domain transformation (DLDT) in this chapter.

4.1.1 Motivations

Traditional OS-learning is only defined for the learning process using raw dataset for local classifiers. For P2P learning, the OS-learning is required to be covered among all peers instead of a single peer and to consider an efficient way to share knowledge instead of sharing raw data.

While distributed OS-learning ability makes the ML classifiers efficient and robust to learn additive observations which are likely to contain new data labels, a robust classification or prediction process is required to improve the accuracy when additive observations occur during the prediction process. Traditional ML classifiers can only predict data labels which the classifiers have learned from training datasets, no matter the true label of predicted data belongs to the learned data labels or not. Moreover, existing ML classifiers either support one-class classification\(^2\) \([46][25][84][38]\) or multiclass classification \([36][14]\). But in the OS-learning process, the classifier performs one-class or multiclass classification continuously by learning from datasets

---

\(^1\)GDL indicates unique labels for data among all peers. For example, in healthcare systems, it can be a name of a disease like “diabetes”.

\(^2\)One-class classification tries to identify data of a specific class by learning from a training set containing only the samples of that class.
with uncertain data labels. So the update capacity from one-class to multiclass classification is necessary for the classifiers to become more robust and efficient in the P2P healthcare diagnostic system.

Recently, many studies have focused to explore various ML classifiers for the distributed environment [28][83]. Instead of solving the challenges in learning and predicting samples with additive data labels, these techniques assume that the number of data labels is fixed in both learning and classification processes. Moreover, they mainly focus on speeding up the calculation with huge training samples or reducing the communication cost in the P2P learning process.

More specifically, in this chapter we propose the main issues of the P2P learning process in existing healthcare systems can be summarized as follows:

- Lack of distributed OS-learning ability: The existing models in healthcare systems are retrained with all raw datasets when encountering patients’ data with new diseases. For example, the classifier has to learn both dataset “A” and “B” when it receives “B” with new data labels, even it already learned “A” previously.

- Non-robust classification during the incremental learning environment: The models in healthcare systems cannot diagnose patients’ medical condition correctly when features of patients’ data do not belong to the knowledge that has been already learned by the classification models.

### 4.1.2 Contributions

Compared to our previous work in Chapter 3, we enhance the P2P learning system with dynamic data labels in both P2P learning and prediction schemes. More specifically, in this chapter we propose a fast, robust and accurate classifier, named *Collaborative Extreme Learning Machine (CELM) with a Confidence Interval (CI)*, to learn and predict data in the P2P environment. The CELM is an improved model of basic Extreme Learning Machine (ELM) [36], which is a form of Single Hidden Layer Feedforward Neural Networks (SLFNs) without the need of turning the hidden layer. The training calculation of SLFNs makes our model
Related Work

Efficient in terms of learning and sharing. In addition, the CI is used to improve prediction accuracy by filtering prediction results without confidence, and shows an interval which might contain the most of true predictions. These filtered results can be analyzed by other data analysis methods (e.g. Support Vector Machine (SVM) [14] and Hidden Markov Model (HMM) [23][56][70]) or experts, which is not considered in this chapter. Our proposed model not only meets the requirements of P2P learning, but also shows fast training speed and excellent prediction accuracy of diagnosing healthcare data.

The main contributions of this chapter are summarized as follows.

- We propose an enhanced P2P healthcare system which diagnoses medical events based on patients’ vital biosignal data. The medical events are predicted by an enhanced ELM classifier named CELM, which supports distributed OS-learning and provides robust data prediction.

- The proposed CELM is able to learn from the biosignal data continuously from knowledge-parameters which are also learned and shared by another peer, instead of learning from raw training datasets in the distributed environment.

- We embed CELM with a CI in the prediction process, which provides better accuracy in terms of healthcare biosignal data.

4.2 Related Work

From previous chapters, one of the issues of existing healthcare systems is that these isolated systems do not share any medical knowledge from others to improve the diagnostic accuracy. In order to break the isolation, the [57] and [79] developed collaborative ML models in a fully distributed environment. Their algorithms consume plenty of time to get acceptable prediction accuracy, because a peer needs to transmit and updates its knowledge (the calculated parameters from the learning stage) many times during the learning process. This is caused by tuning the gradient-descent based classifiers (e.g. SVM [14]).

In the past decades, the Feed-Forward Network (FFN) is very popular for processing data and the Gradient Gescent (GD) is the most common algorithm for tuning the network. The
[49] provides a clear summary of online learning for large-scale data using GD methods in terms of binary classification. The GD based methods are slow, while being easy to converge to the local optimal solution with more layers in multi-layer FFN. These limitations restrict the healthcare application of FFN since many bio-features of existing diseases are changed and new diseases are discovered as the change of lifestyle and environment. A method with fast training speed is required to keep healthcare systems up-to-date and providing accurate diagnoses.

In order to solve the tuning problem, ELM, a single hidden layer feed-forward network, is proposed in [36]. As a tuning free learning algorithm, ELM determined output weighted analytically by randomly selected input weights and biases [36]. It has been proved that ELM is much faster than traditional learning algorithms and obtains better generalization performance, especially in medical diagnostic systems [42][67].

However, basic ELM does not support OS-learning which is significant to healthcare systems in terms of training speed and diagnostic accuracy. OS-learning ability helps the systems to update quickly without the need of training a new classifier for detecting newly discovered diseases. So many enhanced prediction algorithms have been proposed to enhance ELM. The [47] proposed Online Sequential ELM (OS-ELM) which can learn sequential training data in both one-by-one and chunk-by-chunk modes. Similar to OS-ELM, other OS-ELM based algorithms, such as Enhanced Online Sequential ELM (EOS-ELM) [41] and Robust OS-ELM (ROS-ELM) [31] only support fixed data labels and require a specific number of samples to initialize. The [8] solves additional data labels by simply adding 0 in corresponding output weights of OS-ELM. But this means that existing knowledge is not available to the new output calculation. Even though [40] explored incremental ELM to address the above limitations in OS-ELM and fully supports OS-learning, instead of considering P2P or distributed learning which is significant to improve diagnostic accuracy by sharing knowledge, the above-mentioned techniques only enhance the local models and focus on multiclass classification.

In terms of distributed learning, [28] and [83] employ MapReduce (a common framework for distributed computing) to speed up the ELM training process with a large number of samples. Their systems contain a master server and several slave servers. The slave servers are used
to calculate matrix parameters in parallel, while the master server manages all slavers and
calculate a final model with the outputs from each slaver. But these systems do not consider
OS-learning and the communication cost among the servers.

In addition, the classifier should be always ready for the prediction/classification task during
OS-learning, which means the classifier is available to predict/classify new data even though it
only learned from data with one target data label. The one-class prediction ability makes the
healthcare systems more robust. The [46], [25], [84] and [38] propose different One-Class ELMs
to perform one-class classification. But their methods do not support the case of OS-learning
and distributed learning.

In [80], we explored a novel P2P learning system based on ELM for clinical diagnosis
using multi-biosignal data. The system provides the foundation for P2P learning in terms
of the network sharing, which allows every medical institution peer being able to share the
new knowledge discovered from patients’ biosignal data with other peers in order to improve
diagnostics accuracy. But the system only supports learning knowledge with certain data labels
which are predefined, and these labels cannot be increased during the P2P learning process.

Inspired by the above algorithms, we explore CELM with a CI, an enhanced ELM algorithm,
which supports OS-learning and one-class to multiclass classification to facilitate the healthcare
P2P learning system effectively.

### 4.3 System model

The system model of our proposed method, shown in Figure 4.2, consists of mainly three
components: body sensor devices for biosignal data collection, a portable device for data
transmission and user interaction, and medical institutions for data analysis.
Multiple vital signals
A patient with wearable body sensors
Wearable smart device
Large number of patients
Medical Institution
Machine-learning based healthcare system

Figure 4.2: The architecture of our proposed system. Wearable devices collect multiple vital signals from body sensors in a patient, and send these data to a medical institution. The institution employs servers to detect clinical events with a machine-learning based healthcare system and notifies the patient. Every institution also generates knowledge-parameters from their own patient’s raw data and broadcasts the knowledge-parameters with other neighbor institutions to achieve a smarter healthcare system.

- **Body sensor devices**: These kinds of devices are embedded with different sensors to collect patient data in real time. For example, Shimmer is a sensor platform with Bluetooth to acquire real-time electrocardiogram (ECG) from a user [9]. Generally, the sensors are placed in different parts of the body to monitor specific data (e.g. blood pressure or heart rate).

- **Portable device**: The portable device is the main media for patients to interact with the system. It is used to receive the patients data from various body sensor devices using wireless communication capability (e.g. Bluetooth or 4G) and then transmits the data to a medical institution for further analysis. It can also receive a response from the medical institution (e.g. a message or a phone call) once an abnormal clinical event is detected.
Medical institution: It is the data processing body for handling patients data with powerful and complex terminal servers. The servers take the responsibility for recording the data and discovering new knowledge from the data. They can also detect medical status using newly received patients’ data and then notify patients properly. Moreover, they share their knowledge with other peers (other medical institutions) to improve the diagnostic accuracy of other medical institutions in the P2P healthcare network system.

4.4 Overview of our proposed system

Figure 4.2 shows the overview of our proposed P2P learning system where vital biosignal data from patients’ wearable devices are used to train the classifier and diagnose patients’ medical conditions. In our system, medical knowledge of a peer is shared among all peers in the P2P network. We can consider a scenario where an elderly patient living alone at home with wireless body sensors is monitored by a remote healthcare system in a medical institution continuously. The system in the clinical site keeps tracking his/her physiological conditions by receiving biosignal data collected from the patient’s wearable smart device (e.g. smartwatch). More specifically, the system cleans the noisy data and performs data normalization, trains the CELM classifier and also discovers medical knowledge. Therefore, future clinical states can be measured by the trained CELM using newly arrived data from patients. Then clinicians (e.g. doctors and nurses) in the medical institution can diagnose patients’ clinical states and notify them properly.

Concretely, our proposed system is divided into two main stages.

- Data collection: The collection process is used to gather patients’ data for learning the features and predicting the clinical events. In the collection process, time-sequential data from wearable body sensors of a patient are continuously sent to his/her portable devices. The devices then transmit those physiological data to a specific peer for further processing. The data are stored on the server and are used to train the healthcare system. In our system, a training dataset contains vital biosignal records of patients and their labels (clinical events).
• Data sharing: In this stage, every peer shares its knowledge with other peers which is significant in terms of improving the prediction accuracy. In the system, every peer has a certain number of neighbor peers connecting with it directly, which forms a P2P network with a specific protocol. Each CELM in a peer extracts knowledge-parameters from training datasets and broadcasts the knowledge-parameters to its neighbors. In particular, to eliminate the redundancy of knowledge-parameters (filtering the learned knowledge received from other peers) efficiently, the MD5 hash value of each knowledge-parameter is generated for comparison [61].

4.4.1 Overview of our proposed classifier

In our proposed system, we introduce a novel classifier CELM which can provide efficient and robust P2P learning and prediction in smart e-healthcare system. The proposed CELM with a CI enhances basic ELM with distributed OS-learning ability and adopts threshold comparison to improve the prediction accuracy.

Figure 4.3 shows the overview of our proposed classifier. There are three key parts of our proposed model: the DLDT, knowledge-parameter and the CI.
Figure 4.3: The P2P learning process and prediction process in a peer. (a) A peer receives knowledge-parameters from neighbour peers. Then the transformed parameters after DLDT are used to update CELM. After that, a new CI is calculated using the updated CELM and the validation set reserved in the initial stage. After training, the original knowledge-parameters are transmitted to other neighbour peers. (b) At the beginning, a prediction result of data is calculated by CELM and then filtered by the CI. Finally, DLDT is applied to the filtered output to obtain the final prediction outcome.

- DLDT: It is a transformation process between GDL and LDL. For example, doctors use “diabetes” to name a kind of disease, so the disease name can be a GDL in all peers. But different peers may use different LDLs (e.g. 0, 1 or 2) to label diabetes samples for calculation. This may happen when the peers receive the knowledge-parameters from other peers in uncertain order. As a result of using different labels in different peers, the process of updating the knowledge in each peer faces new difficulties. In order to make the meaning of data labels consistent and to perform correct update calculation, DLDT is necessary in our proposed model.

- Knowledge-parameter: It contains all information of a training dataset for updating our proposed classifier and sharing the parameters efficiently across the P2P system, which are extracted by CELM in a peer and are broadcasted across the whole P2P network. There are two main components (which will be described in more detail in Section 4.5.3) in a knowledge-parameter. They are represented as matrices stored in different data structures (Array and Map). In addition, the version of a knowledge-parameter is calculated by the MD5 algorithm, which represents a unique training dataset.
and can avoid the model learning from the same parameters repeatedly.

- **CI**: It is an interval where a certain percentage of the population is likely to lie, which is used to filter prediction results. If the value of a prediction result is within the CI, this prediction is considered as strong and confident. Applying the CI can improve prediction accuracy significantly and can make prediction process more robust in the P2P learning system. The calculation of the CI is derived from a bound-based threshold method and one-class ELM mechanism [84]. Every time when CELM is updated, the CI is required to be updated.

According to our proposed model, a single peer has three main steps to learn and predict the clinical events.

- **The initial process**: Initially the model has no prior knowledge once it is created. Based on learning calculation of ELM, a single peer generates a local knowledge-parameter from an initial training dataset and learns from these parameters. The initial dataset is reserved as the validation set.

- **The P2P learning process**:
  - A peer receives new knowledge-parameters shared from other peers and filters learned knowledge-parameters by the version of the parameters.
  - The local knowledge-parameter is updated with the newly received knowledge-parameters, and then used to calculate the updated model. Moreover, the GDLs in knowledge-parameters are transformed from global unique strings to local numerical indexes.
  - A new CI of a local model is calculated using the updated model and the validation set.
  - The peer broadcasts the newly received knowledge-parameters to other peers. Every peer shares information with its neighbor peers and the neighbor peers transmit to their neighbors [80].
• The prediction process:
  – The proposed CELM calculates prediction vectors for patients’ biosignal data.
  – The prediction vectors are filtered based on the CI to get the confident prediction vectors. Then the index of maximum values in the confidence prediction vectors are achieved.
  – The numerical output is transformed from the local numerical index to the global unique string.

4.5 Proposed Collaborative CELM with a CI

In this section, details of our proposed model are described in terms of DLDT, knowledge-parameter generation and the calculation of a CI. The learning and prediction processes are also depicted. The basic functions of matrix calculation used in our algorithms are described in 4.5.1 and Table 4.1 shows all notations.
Table 4.1: Summary of notations.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>A Global Data Label (GDL)</td>
</tr>
<tr>
<td>$i$</td>
<td>A Local Data Label (LDL)</td>
</tr>
<tr>
<td>$A$</td>
<td>The data structure: Array</td>
</tr>
<tr>
<td>$M$</td>
<td>The data structure: Map</td>
</tr>
<tr>
<td>$n$</td>
<td>The number of learned GDLs</td>
</tr>
<tr>
<td>$K$</td>
<td>The number of hidden neurons</td>
</tr>
<tr>
<td>$X$</td>
<td>A training set</td>
</tr>
<tr>
<td>$D$</td>
<td>A validation set</td>
</tr>
<tr>
<td>$w,b$</td>
<td>The random parameters in Equation (2.1)</td>
</tr>
<tr>
<td>$U$</td>
<td>Represents $H^T H$ in Equation (2.7)</td>
</tr>
<tr>
<td>$V$</td>
<td>Represents $H^T T$ in Equation (2.8)</td>
</tr>
<tr>
<td>$H$</td>
<td>The hidden neuron output in Equation (2.4)</td>
</tr>
<tr>
<td>$T$</td>
<td>The data label matrix</td>
</tr>
<tr>
<td>$\beta$</td>
<td>The output matrix in Equation (2.6)</td>
</tr>
<tr>
<td>$C_1$</td>
<td>The first component in a knowledge-parameter</td>
</tr>
<tr>
<td>$C_2$</td>
<td>The second component in a knowledge-parameter</td>
</tr>
<tr>
<td>$C_B$</td>
<td>The confidence upper bound in a CI</td>
</tr>
<tr>
<td>$f$</td>
<td>The factor controlling the confidence upper bound</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>Represents newly arrived data</td>
</tr>
</tbody>
</table>

4.5.1 Function definitions in our algorithms

The functions in our algorithms described in this section are defined as follows.

1. ones($m,n$) returns an m-by-n matrix of ones.
2. zeros($m,n$) returns an m-by-n matrix of zeros.
3. triu($X$) returns the upper triangular part of $X$.
4. triu($X,k$) returns the element on and above the k-th diagonal of $X$.
5. repmat($A,1,n$) horizontally stacks a column vector $A$ $n$ times and returns the result matrix.
6. max($A,[],2$) returns a column vector containing the maximum value of each row in the matrix $A$.
7. mean($A$) returns the mean of the elements of the vector $A$.
8. std($A$) returns the standard deviation of the elements of the vector $A$. 
(9) \text{size}(X, \text{dim}) \text{ returns the size of the dimension of } X \text{ specified by scalar } \text{dim}.

4.5.2 Data Label Domain Transformation (DLDT)

The DLDT shows the process of transformation of data label from global to local domain. Practically, the global label GDL is represented by a unique string (e.g. “diabetes”) and the local label LDL is represented by a numerical index (e.g. 0 or 1 or 2). Considering an example that a peer uses 0 to label diabetes samples while other peers label the same samples with 1, the DLDT makes the meaning of each label consistent among the peers regardless of the value of local indexes.

As shown in Algorithm 3, the transformation is executed in both learning and prediction process, which helps every peer to manage their LDL indexes locally. Global unique strings are transformed into local numerical indexes in the learning process and the reversed transformation is performed in the prediction process. Both transformation processes are shown in Algorithm 3(a) and (b). Array and Map are the data structures adopted in the algorithms. There are three key processes in DLDT as follows. More specifically, String \(s\) and Integer \(i\) represent a GDL and LDL respectively. Every GDL (String \(s\)) is inserted into Array \(A\) according to the receiving order. The relationship between GDL and LDL are stored in Map \(M\).

- The transforming process from GDL to LDL in learning
  In the learning process, Algorithm 3(a) transforms a global unique string into a local index which is stored in the key-value structure Map \(M\). Concretely, the key is from global unique strings and the value is from local indexes.

- The transforming process from LDL to GDL in prediction
  In the prediction process, Algorithm 3(b) transforms a local index into a global unique string which is stored in the data structure Array \(A\). The order of strings in the Array depends on the time of encountering new GDLs in a peer.

- The recording process with any new GDL
  When encountering unknown GDLs each time, our model records the new GDLs with Map and Array. The insert process is shown in Algorithm 3(c). More specifically, a
Algorithm 3 The functions of Data Label Domain Transformation (DLDT)

1: **Initialize:** Map $M$ contains the relationship between GDL and LDL, Array $A$ contains all leanred GDLs.

2: 

3: (a) The proposed transformation from GDL to LDL in learning.
4: **Input:** Global Data Label (GDL): $s$
5: **Output:** Local Data Label (LDL)
6: **procedure** `GET_LD_L_by_GDL(s)`
7: if $M$ has $s$ then
8: return $M[s]$ $\triangleright$ Return the corresponding LDL
9: else
10: LDL does not exist.
11: end if
12: **end procedure**

13: 

14: (b) The proposed transformation from LDL to GDL in prediction.
15: **Input:** Local Data Label (LDL): $i$
16: **Output:** Global Data Label (GDL)
17: **procedure** `GET_GDL_by_LDL(i)`
18: if $i < \text{the length of } A$ then $\triangleright$ Check whether the requested GDL is existed
19: return $A[i]$ $\triangleright$ Return the corresponding GDL
20: else
21: GDL does not exist
22: end if
23: **end procedure**

24: 

25: (c) The proposed recording process with a new GDL.
26: **Initialize:** Array $A$ and Map $M$
27: **Input:** Global Data Label (GDL): $s$
28: **procedure** `RECORD_GDL(s)`
29: Push $s$ into $A$ $\triangleright$ Save an unlearned GDL
30: $i \leftarrow \text{The length of } A$ $\triangleright$ Generate a new LDL for the GDL
31: $M[s] = i$ $\triangleright$ Save the relationship between the GDL and LDL
32: **end procedure**

newly arrived GDL is always added to the end of the Array. The local index from Array is used as an LDL in our algorithm.
4.5.3 Knowledge-parameter generation of a dataset

The knowledge-parameter of a training dataset contains the required information to update peers’ knowledge. This helps to provide an efficient sharing and learning speed compared to learning from raw training datasets.

We explore our own knowledge-parameter based on $U$ and $V$ identified in Equation (2.12), which is generated from a training dataset. Concretely, $U = H^T H$ and $V = H^T T$, where $H$ represents the hidden layer output matrix and $T$ represents data label vectors. There are two components in the knowledge-parameter, named as $C_1$ and $C_2$, which are used to update the knowledge of peer in our proposed P2P learning process.

4.5.3.1 Generating the first component ($C_1$) in a knowledge-parameter

In order to reduce the communication cost in terms of data volume, we take only part of $U$, since $U$ is a symmetric matrix. $C_1$ is extracted from the upper triangle part of $U$, which is shown in Section 3.3.2.1.

The detailed process of generating $C_1$ is shown in Algorithm 4(a). We calculate $U$ using Equation (2.1) and (2.7) (Algorithm 4 Step 8 and 9), and then transform the symmetric set $C_1$ into an array, in Step 10 and 11. The process of transforming $C_1$ back to $U$ is shown in Algorithm 4(b). In the Algorithm 4, $K$ is the number of the hidden neurons and $X$ represents training sample data. $w$ and $b$ are random input weights parameters shown in Equation (2.1).
Algorithm 4 The functions of $C_1$ in a knowledge-parameter

1: Initialize: The number of hidden neurons: $K$
2: 
3: (a) Generate $C_1$ from samples with the same GDL.
4: Initialize: The random parameters: $w$ and $b$
5: Input: A training dataset with one GDL: $X$
6: Output: $C_1$
7: procedure GENERATE_C1($X$)
8:     $H = X \ast w + b$ ▷ Calculate the hidden layer output matrix
9:     $U = H^T \ast H$ ▷ Calculate the $U$ using Equation (2.7)
10:    $m = \text{triu}(\text{ones}(K, K))$ ▷ Generate the upper triangular matrix of ones
11:    $C_1 = U(m == 1)$ ▷ Extract the upper triangular part of $U$
12:    return $C_1$
13: end procedure
14: 
15: (b) Transform $C_1$ to $U$.
16: Input: $C_1$
17: Output: $U$
18: procedure TRANSFORM_C1_TO_U($C_1$)
19:    $m_1 = \text{triu}(\text{ones}(K, K))$ ▷ Generate the upper triangular matrix of ones
20:    $m = m_1$ ▷ Copy the matrix
21:    $m(m_1 == 1) = C_1$ ▷ Change $C_1$ to the upper triangular part of $U$
22:    $z = m^T$ ▷ Calculate the lower triangular
23:    $U = z + \text{triu}(m, 1)$ ▷ Merge the upper and lower triangular matrix
24:    return $U$
25: end procedure

4.5.3.2 Generating the second component ($C_2$) in a knowledge-parameter

The other component in a knowledge-parameter $C_2$ is extracted from a training dataset and is used to construct $V$ in Equation (2.12) where $V = H^T T$. $H^T T$ is generated from Equation (2.6) with the hidden layer output matrix $H$ and data label vectors $T$. At the beginning, we order all samples based on their LDL numerical indexes. Then, according to Equation (2.4)
and (2.8), \( V \) is calculated as

\[
V_{ij} = H^T T = \sum_{j=1}^{N} g_{ij} \ast t_{jc} \quad \text{and} \quad t_{jc} \in \{-1, 1\}
\]

\[
H^T = \begin{bmatrix}
g_{11} & \cdots & g_{1n} \\
\vdots & \ddots & \vdots \\
g_{k1} & \cdots & g_{kn}
\end{bmatrix}_{K \times N} \quad \text{and} \quad T = \begin{bmatrix}
t_{11} & \cdots & t_{1c} \\
\vdots & \ddots & \vdots \\
t_{n1} & \cdots & t_{nc}
\end{bmatrix}_{N \times C}
\]

(4.1)

where \( i \) and \( j, c \) represent the index of rows and columns respectively. Since the value of \( t_{jc} \) only belongs to \(-1\) or \(1\), The calculation of \( V \) can be merged based on the same \( t_{jc} \). So considering every column of \( V \), we have

\[
V_c = 2 \ast \sum_{j=p}^{q} g_{ij} - \sum_{j=1}^{N} g_{ij}
\]

(4.2)

where \( p \) to \( q \) confines the range in which \( t_{jc} \) is \(1\), and \( N \) is the number of training samples.

Let \( Z = \sum_{j=1}^{N} g_{ij} \). It is clear that every \( Z \) is the same in the calculation of each column. Since each column of \( V \) represents an output neuron directing a unique LDL, the only difference is the range \((p, q)\). So \( Z \) can be represented as follows based on different ranges having the same \( t \).

\[
Z = \sum_{j=1}^{a} g_{ij} + \sum_{j=a+1}^{b} g_{ij} + \cdots + \sum_{j=y+1}^{z} g_{ij} + \sum_{j=z+1}^{N} g_{ij}
\]

(4.3)

where every sum within the ranges \((1, a), (a + 1, b) \cdots (y + 1, z)\) and \((z + 1, N)\) is calculated using samples with the same LDL. The range \((p, q)\) in Equation (4.2) can be one of the ranges in Equation (4.3) and indicates a GDL \((s)\). So \( C_2 \) is a set of tuples shown as

\[
C_2 = \{\cdots, < s, \sum_{j=p}^{q} g_{ij} >, \cdots \}
\]

(4.4)

We adopt the key-value data structure (Map) to store each sum. In particular, \( X \) are samples with the same GDL in Algorithm 5.
Algorithm 5 The functions of $C_2$ in a knowledge-parameter

1: (a) Generate $C_2$ from samples with the same GDL.
2: Initialize: $C_2$, The random parameters: $w$ and $b$
3: Input: A training dataset $X$ with the GDL $s$
4: procedure GENERATE_C2($X, s, C_2$)
5: $H = X \ast w + b$ \hfill $\triangleright$ Calculate the hidden layer output matrix
6: $T = \text{ones(size}(X, 1), 1)$ \hfill $\triangleright$ Generate all-ones data labels
7: $C_2[s] = H^T \ast T$ \hfill $\triangleright$ Equation (4.4)
8: end procedure
9: 
10: (b) Get $V$ and $Z$ from $C_2$.
11: Initialize: The number of hidden neuron: $K$
12: Input: $C_2$, The number of learned GDLs: $n$
13: Output: $V, Z$
14: procedure GET_V_AND_Z_FROM_C2($C_2, n$)
15: $Z = \text{zeros}(K, 1)$ \hfill $\triangleright$ Generate a $Kx1$ matrix with 0
16: for each GDL in $C_2$ do
17: $Z = Z + C_2[GDL]$ \hfill $\triangleright$ Equation (4.3)
18: end for
19: $V = \text{repmat}(\text{-}Z, 1, n)$ \hfill $\triangleright$ Generate a $Kx n$ matrix with $-Z$
20: for each GDL in $C_2$ do
21: $LDL = \text{GET_LDL_BY_GDL}(GDL)$
22: $V(:, LDL) = 2 \ast C_2[GDL] + V(:, LDL)$ \hfill $\triangleright$ Update the column of $V$ using Equation (4.2)
23: end for
24: return $V, Z$
25: end procedure

Algorithm 5(b) shows the transformation from $C_2$ to $V$ and the calculation of $Z$ using $C_2$, where $n$ is the number of total targeted data labels. The column of $Z$ is calculated from Step 14 to 16 based on Equation (4.3) and then transformed to the matrix in Step 17. Finally, $V$ is achieved from Step 18 to 20 based on Equation (4.2).

4.5.4 The calculation of a Confidence Interval (CI)

The CI represents an estimate of an interval $[0, C_B]$ where future observations will fall, with a certain probability, given what has already been observed, which can increase prediction accuracy and make prediction more robust. We denote the upper bound of a CI as $C_B$. Since the lower bound of a CI is always 0 in our method, the range of a CI only depends on $C_B$. So
the way to calculate a CI is to identify $C_B$.

Similar to [25], we adopt a bound-based threshold method to generate confidence bounds, which is used to filter prediction results lacking confidence. The value of the upper bound is controlled by a custom factor $f$ (a specified percentage to confine the range). According to One-Class ELM [84], the confidence upper bound represents an expected distance between an output and the hyper plane, which is used to classify whether the data belong to the specific data class or not. If the distance is closed to 0, the prediction result is more confident. According to the mechanism, the CI is used to filter prediction results which have no confidence in our proposed algorithm.

The method is to calculate a CI using the absolute deviation (AD) between an actual and predicted value as follows:

$$C_B = \text{Mean}(d) + f \ast \text{Std}(d) \quad (4.5)$$

Where $\text{Mean}(d)$ and $\text{Std}(d)$ are the mean and the standard deviation of the AD throughout all training data respectively, and $f$ is the custom factor. $d = |\max(H \cdot \beta) - T|$ is inferred from Equation (2.14). $H$ and $T$ represent the hidden layer output matrix and data label vectors respectively.

Algorithm 6 shows the CI calculation with a validation set $D$. The normal prediction similar to ELM is from Step 5 to 7. Then the AD values are calculated through all samples in Step 8 based on Equation (2.14) and finally, the upper bound is obtained in Step 9 according to Equation (4.5).
Algorithm 6 Calculate the upper bound of a Confidence Interval (CI)

1: **Initialize:** The random parameters: $w$ and $b$, The custom factor: $f$, The output matrix in Equation (2.6): $\beta$
2: **Input:** A dataset: $D$
3: **Output:** The upper confident bound: $C_B$
4: **procedure** GET_CONFIDENT_BOUND($D$)
5: $H = D \ast w + b$
6: $T = H \ast \beta$
7: $M = \max(T, [], 2)$ \hspace{1cm} $\triangleright$ Get prediction values of all samples in $D$
8: $d = ||M|-1|$ \hspace{1cm} $\triangleright$ Calculate the distance using Equation (2.14)
9: $C_B = \text{mean}(d) + f \ast \text{std}(d)$ \hspace{1cm} $\triangleright$ Equation (4.5)
10: **return** $C_B$
11: **end procedure**

4.5.5 Learning process of our proposed model

As shown in Figure 4.4, our proposed model can learn continuously using a knowledge-parameter which is generated from different training datasets or shared from other peers. When $U$ and $V$ in a model are updated, a new classifier is achieved using $\beta = U^{-1}V$ in Equation (2.13). In order to update our proposed model, we need $U$ generated from $C_1$, as well as $U$ and $Z$ generated from $C_2$. More specifically, the process of updating $C_1$ is accomplished by adding corresponding elements of $C_1$ and newly arrived $\triangle C_1$, while the calculation of updating $V$ and $Z$ with newly arrived $\triangle C_2$ is derived as follows.

Suppose there are $N$ samples learned in the system and $\triangle C_2$ is generated from a training dataset with $\triangle N$ samples. If there are $n$ new GDLs in $\triangle C_2$, the number of columns $C$ in $V$ is updated with $C' = C + n$, and every element of $\triangle V$ is calculated with Equation (4.2) and $\triangle C_2$ as follows.

$$\triangle V_c = 2 \ast \sum_{j=p'}^{q'} g_{ij} - \sum_{j=N+1}^{N+\triangle N} g_{ij} \hspace{1cm} p', q' \in (N+1, N+\triangle N) \hspace{1cm} (4.6)$$

where $\sum_{j=p'}^{q'} g_{ij} = 0$ if $c$ is not new. $Z$ in Equation (4.3) is updated as

$$Z' = Z + \triangle Z \hspace{1cm} \triangle Z = \sum_{j=N+1}^{N+\triangle N} g_{ij} \hspace{1cm} (4.7)$$
In original columns \((c')\) of \(V\), when encountering new GDLs in \(\Delta C_2\), Equation (4.2) becomes

\[
V'_{c'} = V_{c'} - \sum_{j=N+1}^{N+\Delta N} g_{ij} = V_{c'} + \Delta V_{c'}
\]  

(4.8)
since \(\sum_{j=N'}^{q'} g_{ij'} = 0\). In new columns \((c'')\) of \(V\) representing new GDLs in \(\Delta C_2\), we have

\[
V''_{c''} = \Delta V_{c''} - Z = 2 * \sum_{j=p'}^{q'} g_{ij} - \sum_{j=N+1}^{N} g_{ij} - \sum_{j=1}^{N} g_{ij}
\]  

(4.9)

If we initialize new columns of \(V_{c''}\) with \(-Z\), the update process of \(V\) using Equation (4.8) and (4.9) becomes

\[
V'_{c} = V_{c} + \Delta V_{c}
\]  

(4.10)

where \(c\) is original or new columns.

---

**Figure 4.4:** Learning process based on a knowledge-parameter \(C_1\) and \(C_2\). CELM is installed in a local peer, which contains local \(C_1\) and \(C_2\). When receiving \(\Delta C_1\) and \(\Delta C_2\), the local peer adds local \(C_1\) and \(\Delta C_1\) together and transformed the additive result into the updated \(U\). Then if there is any new GDL in \(\Delta C_2\), the local \(V\) initializes new columns with the local \(Z\). After that \(\Delta V\) and \(\Delta Z\) are calculated using \(\Delta C_2\), which are used to update local \(\Delta V\) and \(\Delta Z\) respectively. Finally, a updated model is calculated with updated \(U\) and \(V\), and is used to update the CI with the validation dataset.
As shown in Algorithm 7, firstly $C_1$ is updated with $\Delta C_1$ in Step 4 and transformed into $U$ in Step 5. Then additive GDLs are detected analytically based on GDLs in $C_2$ from Step 7 to 11. When additive GDLs exist, $Z$ is used to initialize new columns of $V$, shown from Step 13 to 14. Then $\Delta V$ and $\Delta Z$ are generated from $\Delta C_2$ using Algorithm 5 (b). According to Equation (4.7), $Z$ is updated in Step 17. A new model is calculated in Step 22 after $U$ and $V$ are updated according to Equation (2.13). Finally, a new CI is calculated based on the new model using the validation set.

**Algorithm 7** Learning process with a knowledge-parameter

1. **Initialize:** $C_1 = 0$, $U = 0$, $V = \{\}$, $Z = 0$, The confidence upper bound:$C_B$, The output matrix in Equation (2.6):$\beta$, A validation set:$D$
2. **Input:** A newly arrived knowledge-parameter:($\Delta C_1$, $\Delta C_2$)
3. **procedure** LEARNING($\Delta C_1$, $\Delta C_2$)
4. \[ C_1 += \Delta C_1 \] \hspace{1cm} \triangleright Update $\Delta C_1$ (Step 1 in Figure 4.4)
5. \[ U = \text{TRANSFORM}_C_1 \text{TO}_U(C_1) \] \hspace{1cm} \triangleright Step 2 in Figure 4.4
6. \[ q = 0 \] \hspace{1cm} \triangleright The number of new GDLs
7. **for** each GDL in $\Delta C_2$ **do**
8. \[ LDL = \text{GET}_LDL\text{BY}_GDL(GDL) \]
9. **if** LDL does not exist **then**
10. \[ \text{RECORD}_GDL(GDL) \]
11. \[ q = q + 1 \]
12. **end if**
13. **end for**
14. **if** $q > 0$ **then** \hspace{1cm} \triangleright New GDLs exist in $\Delta C_2$
15. \[ m = \text{repmat}(-Z, 1, q) \]
16. \[ V = [V, m] \] \hspace{1cm} \triangleright Initialize new columns with $-Z$
17. **end if**
18. \[ n \leftarrow \text{the number of total GDLs} \]
19. \[ \Delta V, \Delta Z = \text{Get}_V\text{And}_Z\text{From}_C2(\Delta C_2, n) \] \hspace{1cm} \triangleright Step 4 in Figure 4.4
20. \[ Z += \Delta Z \] \hspace{1cm} \triangleright Update $Z$ (Step 5 in Figure 4.4)
21. \[ V += \Delta V \] \hspace{1cm} \triangleright Update $V$ (Step 6 in Figure 4.4)
22. \[ \beta = U^{-1} \ast V \] \hspace{1cm} \triangleright Update the model (Step 7 in Figure 4.4)
23. \[ C_B = \text{CET}_\text{CONFIDENT}_\text{BOUND}(D) \] \hspace{1cm} \triangleright Step 8 in Figure 4.4
24. **end procedure**
4.5.6 Prediction process for our proposed model

Figure 4.5 illustrates the prediction process with a CI. When having only one output neuron, the proposed model performs one-class classification. On the other hand, in the multiclass prediction process, the model filters prediction results using a CI and then outputs LDLs. Finally, each LDL is transformed into its GDL mentioned above.

Algorithm 8 shows our proposed predicting algorithm for a sample. The algorithm calculates a prediction result vector \((T)\) of a sample in Step 5 and 6 using Equation (2.1). If the distance between the predicted value and the hyper plane is not within the CI, the algorithm detect the non-confident prediction, which is shown in Figure 4.5(b). Otherwise, the prediction is confident, which is shown in Figure 4.5(c). Finally, DLDT is applied to LDL to obtain GDL. In particular, if there is only one targeted class in the system, one-class classification is
performed, which tells whether a sample belongs to the targeted data class.

**Algorithm 8** proposed predicting algorithm for one sample

1. **Initialize:** $H = 0$, The random parameters: $w$ and $b$, The output matrix in Equation (2.6): $\beta$, The confidence upper bound: $C_B$
2. **Input:** A sample: $X$
3. **Output:** GDL
4. **procedure** `Predict(X)`
   5. $H = X \ast w + b$
   6. $T = H \ast \beta$
   7. $t \leftarrow \text{The maximum value of } T$ $\triangleright$ The prediction value in Figure 4.5
   8. $L \leftarrow \text{The index of } t \text{ in } T$ $\triangleright$ Get the LDL
   9. **if** $||t| - 1| <= C_B$ **then** $\triangleright$ The filtering process in Figure 4.5
      10. **if** only one targeted class exists **then**
          11. The data belongs to the targeted class.
          12. **else**
          13. The data label is GET\_GDL\_BY\_LDL($L$).
          15. **end if**
      16. **else**
      17. **if** only one targeted class exists **then**
          18. The data does not belong to the targeted class.
          19. **else**
          20. The prediction is non-confident.
          21. The data label is GET\_GDL\_BY\_LDL($L$).
          22. **end if**
      23. **end if**
   24. **end procedure**

### 4.5.7 Time complexity

Matrix multiplication dominates computational time in the learning and prediction processes of our method. According to [69], the product of $X \times Y$ matrix $A$ and $Y \times Z$ matrix $B$ runs in $\mathcal{O}(X \ast Y \ast Z)$. We denote $N$ as the number of samples, $K$ as the number of hidden neurons and $C$ as the number of output neurons. The time complexity of each proposed algorithm is shown in Table 4.2. The number of features in each sample is not considered since it can be eliminated at the beginning of the calculation.
4.6 Experimental results

Our experimental section is organized in four parts.

- Data preprocessing: We describe the features of the sample and labels targeted in our preprocessed dataset for P2P learning and prediction. The data cleaning process for correcting measurement errors in the original dataset and the data grouping process are also introduced.

- Identifying optimal parameters for CELM initialization: We find out the optimal parameters to initialize our proposed model which can provide the highest prediction accuracy. Note that we ignore the filtering process with the CI at this part. The effects of the custom factor $f$ controlling the CI are analyzed in the next part.

- Evaluation of our proposed model in a peer during P2P learning: We analyze our processed CELM with a CI in a peer in terms of the prediction accuracy of one-class and multiclass prediction during P2P learning. Besides that, in the multiclass prediction, the effects of the factor $f$ on a CI and filter rate are also introduced. The factor $f$ is used to control the CI and the filter rate shows the percentage of prediction lacking confidence during P2P learning based on different $f$s.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm 3</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Algorithm 4 (a)</td>
<td>$O(K \times K)$</td>
</tr>
<tr>
<td>Algorithm 4 (b)</td>
<td>$O(K \times K)$</td>
</tr>
<tr>
<td>Algorithm 5 (a)</td>
<td>$O(N \times K)$</td>
</tr>
<tr>
<td>Algorithm 5 (b)</td>
<td>$O(K \times C)$</td>
</tr>
<tr>
<td>Algorithm 6</td>
<td>$O(N \times K \times C)$</td>
</tr>
<tr>
<td>Algorithm 7</td>
<td>$O(K \times C)$</td>
</tr>
<tr>
<td>Algorithm 8</td>
<td>$O(N \times K \times C)$</td>
</tr>
</tbody>
</table>
Experimental results

- Analytic comparison with existing works: A comparison among our work with existing works is described in terms of the dataset, prediction accuracy and supported features.

4.6.1 Data preprocessing

In our preprocessed dataset shown in Section 2.3, a sample contains 180 features, which is classified into five labels. We need to correct values of one biosignal (Mean Blood Pressure) due to the obvious measurement error and training samplers are also grouped by their labels in our experiments. The data cleaning and grouping process are described as follows.

4.6.1.1 Data cleaning

Some MBP values have an obvious measurement error according to the relation among SBP, DBP, and MBP [3]

\[
MBP_1 = DBP + \frac{SBP - DBP}{3}
\]  \hspace{1cm} (4.11)

The error is detected by

\[
|MBP_1 - MBP_2| > 20
\]  \hspace{1cm} (4.12)

where \(MBP_2\) is the measured value of \(MBP\). If the measurement error occurs in the MBP biosignal, the measured value is replaced by the calculation value \(MBP_1\).

4.6.1.2 Data grouping

After preprocessing the dataset, we achieve 6681 samples for our experiments. 70\% of the samples form a training dataset and the rest as a testing dataset are shown in Table 4.4. In order to test the performance of our proposed model in OS-learning, we organize the whole training dataset into 5 groups (from D0 to D4), shown in Table 4.3. A group label is used to identify each sample with its representing clinical event.

<table>
<thead>
<tr>
<th>Dataset acronym</th>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>The number of training samples</td>
<td>1159</td>
<td>98</td>
<td>400</td>
<td>198</td>
<td>2160</td>
</tr>
</tbody>
</table>
### Experimental results

#### Table 4.4: The testing set

<table>
<thead>
<tr>
<th>Label</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of samples</td>
<td>781</td>
<td>42</td>
<td>172</td>
<td>85</td>
<td>926</td>
</tr>
</tbody>
</table>

#### 4.6.2 Identifying optimal parameters for CELM initialization

In order to find out optimal parameters for initializing our proposed classifier, we train CELM with all training samples (D0-D4) and analyze the accuracy without filtering. Every accuracy is calculated with the average of 10-time repeated experiments. In all experiments, the same $w$ and $b$ are used to initialize CELM. There are four activation functions compared in Figure 4.6: Sigmoid function ($\text{sig}$), Sine function ($\text{sin}$), Hard-limit function ($\text{hardlim}$) and RBF kernel function ($\text{rbf}$). The number of hidden neurons is increasing from 100 to 1000 at an interval of 100. $\text{Sig}$ shows the best accuracy in all cases, slightly better than $\text{hardlim}$. In addition, under the $\text{sig}$ function, there is a little improvement in accuracy (around 0.2%) when the number of hidden neurons is over 300. Overall, the best accuracy reveals in 500 hidden neurons with the $\text{sig}$ function, which is 95.76%. So all the experiments below follow these settings – 500 hidden neurons and the $\text{sig}$ function – performing the best accuracy.
4.6.3 Evaluation of our proposed model in a peer during P2P learning

In this section, we analyze the performance of our proposed CELM with a CI in the P2P learning process. Three metrics are considered in our analysis: accuracy, the effects of the factor \( f \) on a CI as well as filter rate. In the following, we explain the experimental workflow used to test the performance of the proposed model.

4.6.3.1 Experimental workflow

In order to stimulate the P2P learning process in a peer, the following steps are designed with ignoring the sharing process of knowledge-parameters across a real network.

- Two CELM classifiers (E0 and E1) are created and initialized with same parameters. E1 is the host peer while E0 acts like a neighbor peer for sharing knowledge-parameters.

- An initial dataset is selected for learning. The first 25 samples in the dataset form a validation set and the rest are regarded as training sets.
• E1 learns from the validation set at the beginning. In every Learning Round (LR), E0 generates a knowledge-parameter using every 25 samples in the training dataset. Then E1 is updated with the knowledge-parameter and calculates the CI using the validation set. After updating, E1 does the classification with the testing dataset.

More specifically, 10 different factors \( f \) from 0.1 to 1.0 for controlling the CI are considered in the following experiments.

### 4.6.3.2 Accuracy of one-class classification during P2P learning

In this part, we analyze the accuracy of our proposed model for one-class prediction during the P2P learning process using two datasets D0 and D4 respectively. With the experimental workflow, D0 and D4 with a large number of samples are selected respectively as the initial dataset to evaluate the incremental one-class prediction performance of our proposed method. In the testing dataset, only samples belonging to the targeted class are labeled with 1, the rest are \(-1\).
Figure 4.7: Accuracy of one-class prediction in a peer during the P2P learning process

Figure 4.7 shows the accuracy of one-class prediction using D0 and D4 receptively during the incremental learning process. The red baseline accuracy in the figure represents the proportion
of samples with the targeted data label in the dataset. Obviously, compared to the baseline, our proposed model performs a significant improvement with different factors which also have a slight impact on the final accuracy after finishing P2P learning.

4.6.3.3 General performance during P2P learning (from one-class to multiclass)

After analyzing the accuracy of one-class prediction, we consider multiclass prediction for whole datasets using our proposed model in the P2P learning process in terms of accuracy, the changing of the CI and filter rate. In this experiment, all datasets from D0 to D4 are merged together in ascending order as the initial dataset. The variation in accuracy, the CI and the filter rate with different factors \( (f) \) are analyzed during the P2P learning process. In particular, our model starts to encounter samples with a new data label at LR 72, 75, 92 and 100 receptively.

Accuracy during P2P learning with different factors \( (f) \)

Figure 4.8 shows accuracy in a peer during the P2P learning process. The red line is achieved by ELM with same initialized parameters. Due to the lack of one-class prediction ability, the accuracy of native ELM is empty from LR 1 to LR 71. In terms of multiclass prediction, compared to native ELM, our proposed model shows a noticeable improvement of the accuracy especially when there are some unknown data labels existing in the testing dataset from LR 71 to LR 100. As decreasing the factor \( (f) \), the final accuracy falls slightly from the highest 98.4% \( (f = 0.1) \) to the lowest 97.8% \( (f = 0.9) \).
Experimental results

The effects of different factors \( (f) \) on a CI

Since the lower bound of any CI is always 0, Figure 4.9 shows the changing of the confidence upper bound in a peer during the P2P learning process. In one-class learning stage (before LR 72), the confidence upper bound is close to 0 and does not change significantly with the increasing factor. When more data labels are targeted, a larger factor increases the value of confidence upper bound. When the learning process finished, the confidence upper bound value presents from 0.305 \( (f = 0.1) \) to 0.497 \( (f = 1) \).
The effects of different factors \((f)\) on the filter rate

Our model adopts a CI to filter prediction results lacking confidence in multiclass classification. The filter rate during the P2P learning process is shown in Figure 4.10. As more data are fed to our model, the filter rate is decreasing. In addition, the final filter rate falls from 0.32 \((f = 0.1)\) to 0.14 \((f = 1)\) with the increment in \(f\).
4.6.4 Analytic comparison with existing works

Our model can be applied to different fields to address prediction or classification issues. In terms of healthcare diagnosis systems, even though existing studies adopt different models, datasets and platforms to predict clinical events, it is possible to do a general comparison in terms of performance and features. Our model is compared with 4 existing works shown in Table 4.5. In particular, the work in [23] requires 30 minutes for an Amazon m3.xlarge virtual machine with 4 CPUs and 8GB RAM to train their classifier using Hidden Markov Model (HMM). But less than 10 seconds are needed to update our model in every LR of above experiments. What is more, our model shows the best diagnostic accuracy (Max 98.4%) among all methods.
Table 4.5: Comparison among our model and similar works in terms of results and features.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of biosignals</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Clinical event</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Only acute hypotension</td>
<td>Hemodynamic instability</td>
</tr>
<tr>
<td>Number of normal samples</td>
<td>1657</td>
<td>700</td>
<td>1370</td>
<td>30</td>
<td>571</td>
</tr>
<tr>
<td>Number of abnormal samples</td>
<td>4081</td>
<td>1720</td>
<td>130</td>
<td>30</td>
<td>116</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Max 98.4%</td>
<td>97.7%</td>
<td>Max 95% (GMM) &amp; 96% (SVM)</td>
<td>94%</td>
<td>ROC max 0.86</td>
</tr>
<tr>
<td>Support OS-learning</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Support one-class classification</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Support collaborative learning</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

4.7 Summary

In this chapter, we present an enhanced fast and robust P2P learning system for medical knowledge sharing to achieve better diagnostic accuracy. The enhanced system improves the basic P2P learning system shown in [80] by removing the limit of data labels during the incremental learning and prediction process, which makes the P2P system more robust and flexible. In order to overcome the limitation, we proposed CELM with a DI which is the main contribution and achievement in this chapter. The proposed model facilitates the P2P learning system with the distributed OS-learning ability and performs high prediction accuracy by filtering prediction results using a confidence interval (CI). In the P2P learning process, both the CELM and CI are required to be updated. In particular, OS-learning ability allows our model updating continuously from training datasets in the P2P learning process no matter the datasets contain unknown data labels or not, and the CI indicates an interval where true prediction results are likely to lie. Our proposed model provides the robust prediction process since it can perform prediction from one-class to multiclass during OS-learning. Our experiments adopt a real healthcare dataset and show better performance compared to existing works in terms of training time and accuracy. In the future, we are going to address issues related to kernel calculation and unbalanced dataset in the P2P learning environment.
Chapter 5

Data Priority for P2P Healthcare System

In existing chapters, a robust and effective P2P learning system is explored to address the isolation in current existing healthcare systems. Our system helps medical institutions (peers) to provide better diagnostics services by sharing their knowledge and learning the knowledge in an efficient and robust way.

As mentioned in Chapter 1, except for the isolation, current healthcare systems do not consider patients’ urgency when processing their medical data. This leads to a delay in the assistance for urgent patients and the data volume collected from patients are huge since abnormal and normal data are collected together. In order to solve this issue related to \( RQ.3 \), this chapter shows optimal schemes of data collection and data prediction by considering data priority which can represent patients’ urgency. Data assessment algorithms are explored to evaluate the priority using multi-biosignal data which are collected from patients’ wearable sensors. The assessment algorithms are inspired by National Early Warning Score (NEWS) used in Emergency Department. With patients’ urgency, our P2P system becomes more practical and optimal in the processing flow of data collection and prediction.

The rest of this chapter is organized as follows. Section 5.1 highlights the motivations and contributions of this work and Section 5.2 describes the related works. Then our model and algorithms are proposed in Section 5.3 and 5.4. The empirical experiments are conducted in
5.1 Motivation and Contribution

Huge amounts of raw healthcare data are generated everyday [58], which is challenging the efficiency in the data collection. These data generated by the sensors have the 3Vs characteristics of big data: volume, velocity, and variety [5]. The case becomes more critical with the more elderly population for continuous monitoring. The big data issue in terms of data size has been introduced in $RC\cdot2$. So an efficient method of data collection is required in healthcare systems.

On the other hand, the existing healthcare systems [23][39][65] predict data equally without considering urgency, which leads to delay in treatment of severe conditions of patients. In [73], the authors define clinical decision support as “providing clinicians with computer-generated clinical knowledge and patient-related information which is intelligently filtered and presented at appropriate times to enhance patient care”. So embedding the system with a standardized method to optimize the waiting time of the data is essential and necessary. In a real medical institution, in order to make sure fair access to services and avoid confusion [27], clinical priority settings are used to sort the flow of patients so patients with more urgent conditions should be diagnosed or treated before those with less urgent conditions [54].

In this chapter, we enhance the healthcare system with data assessment in order to improve the efficiency of data collection and optimize the processing flow of data. Our goals are achieved by introducing data priority based on patients’ urgency. Through our algorithms, the data volume is reduced in data collection, which can make data transmission faster. In addition, the system can adjust the waiting time of data before predicting based on the patients’ urgency, which makes the system more practical and optimal. One of the challenges of our work is to assess the priority based on time sequential data, which is explained in $RC\cdot5$. In addition, when trying to collect less biosignal data from the patients, the system needs to provide complete data information to clinicians for accurate diagnoses. For example, doctors need to observe data in a specific time window (e.g. 1 hour or 1 day) in order to diagnose some chronic diseases correctly.
The main contributions of this chapter are summarized as follows.

- A time-window-based method for data collection is introduced in our system. The data collected from patients are organized based on a time window, which is grouped into a Time-Window Data Chunk (TWDC) in our method. Then our method decides which TWDC needs to be transmitted based on the priority from data assessment.

- We proposed a data assessment algorithm based on the clinical priority setting to identify patients’ urgency. Our algorithms can not only assess each TWDC, but also can evaluate the priority based on time sequential priorities of TWDCs, which provides complete medical information for diagnoses.

- An enhanced healthcare system with data assessment is explored, which is efficient in data collection and optimizes the processing order in prediction. With patients’ urgency, the system can reduce the medical data collected from patients by filtering normal data and can adjust the waiting time of data before predicting based on different priorities.

5.2 Related Work

Different studies [23] [10] [65] consider multiple biosignals (e.g. ECG, blood pressure, heart rate, respiration and O₂ saturation) for future abnormality prediction. But the majority are at theoretical level and still far behind to be widely used in public.

Even though [23] and [39] introduce their frameworks at the application level, some improvement can still be considered to make the healthcare system better. The [39] mainly focuses on personal state estimation based on Hidden Markov Model (HMM). Specific rules are used to decide which data need to be transmitted. The [23] explores the Cloud-based framework to deal with the pressure of data storage and processing due to a huge amount of data. Both [23] and [39] use the mobile device to collect and transmit the raw bio-signals into the server continuously. Similar to [39], HMM is adopted to do the clinical event prediction. Another practical example is the BioSign device [72] that can minimize the time of occurrence of critical clinical situation. But there is no predictive capability in the system.
The above systems show good practical solutions for healthcare, but all of them process the medical data from patients’ with the first-in-first-out principle. However, in real clinical cases, a patient’s urgency is commonly considered at the beginning in order to decide the order of medical services. The [4] introduces the effective triage system used in the Emergency Department (ED) when predicting Intensive Care Unit (ICU) admission or in-hospital mortality. The National Early Warning Score (NEWS) is explored in [2], which is a good predictor of patient outcomes and can provide additional value to monitor patients in the ED and in the hospital. According to the existing clinical settings, a patient’s urgency plays a significant role in monitoring patients in the hospital. Therefore, when processing patients’ medical data, the smart healthcare system is required to consider patients’ urgency.

Inspired by all the techniques described above, we explore an enhanced healthcare system to improve the efficiency of data transmission and optimize the prediction flow based on the data priority in this chapter.

5.3 **Overview of our proposed system**

Our enhanced healthcare system using vital biosignals is summarized as follows, shown in Figure 5.1. There are two parts in our system: the client for data collection and the server for data analysis.
Overview of our proposed system

Figure 5.1: The proposed remote healthcare system. TWBP in a portable device is used to assess the priority of every TWDC and filter TWDCs with a specific time window (e.g., 10 minutes). The selected TWDCs are transmitted to the server. The general priority of sequential TWDCs in a specific observation window (e.g., 60 minutes) is calculated by the priority processor. Then the CDSS and clinicians can diagnose the data which are ordered based on their priority. Then a proper notification is sent once an abnormal event is detected. In addition, the data store in the database for further study and the system can synchronize the updated priority classifier to the TWBP.

- The client: It uses portable devices (e.g., smartphone or smartwatch) to collect patients’ data continuously from body sensors and to evaluate data priority for these medical data based on a specific time window (e.g., 10 minutes). The data priority represents patients’ urgency, which means that a more severe patient has a higher data priority. In addition, these time-window medical data and their priorities are organized in a Time-Window Data Chunk (TWDC). The client selects which data chunks need to be transmitted to the server based on their priorities.

- The server: It is employed by a medical institution, which is used to provide the accurate prediction of patients’ medical conditions using their medical data collected from the client. The prediction order of patients’ data is optimized by the data priority, which makes sure that diagnosis decisions are provided at appropriate times based on patients’ urgency. In addition, with more and more patients’ data, the server can provide a more
accurate solution to assess patients’ data and to predict different medical conditions by introducing P2P learning explained in previous chapters.

Concretely, the Time-Window Based Processor (TWBP) is the main component in the client, which contains two key parts: a priority classifier for data assessment and a filter for data selection.

- Priority classifier: It uses a Machine-Learning (ML) method to classify TWDCs into different priorities automatically. The data priority plays an important role in our system optimization. More specifically, the ML classifier is trained with medical samples containing 4 vital biosignals and the labels of training data are identified by our data assessment algorithm based on a real-life clinical setting.

- Filter: It helps the system to collect patients’ medical data more efficiently by considering different data priorities. In our healthcare system, the abnormal data are more valuable and considerable than the normal data since our system is required to provide corresponding diagnosis decisions when detecting the abnormal data. So the filter targets the abnormal data which have higher data priority. In order to make sure that complete medical data information is transmitted to the server for clinicians’ diagnoses, a proposed algorithm for data transmission is introduced into the filter in next section.

In addition, three components are required in the server: database for data storage, priority processor for prediction queue management and CDSS for data prediction.

- Database: It is used for data storage to record TWDCs from different patients. The records of different TWDCs can be used to provide reliable long-term diagnoses for patients and to discovery useful medical knowledge which can improve the smart healthcare system.

- Priority processor: It helps to arrange the prediction order of patients’ medical data based on data priority, which can reduce the waiting time of urgent patients’ data and can help our system to provide assistance to patients’ at the appropriate time based on their urgency. We introduce an algorithm to assess long-term data priority based on the
priorities of time-series TWDCs. For example, if we want to calculate a data priority in an hour, the priorities of 6 time-series TWDCs using a 10-minute window are required.

- **CDSS**: It is a Machine-Learning based classifier which can predict patients’ medical conditions accurately using medical data collected from patients in real time. In order to meet the changing in the clinical environment (e.g. new diseases), the CDSS also has the ability to do P2P learning shown in previous chapters, which can improve the diagnosis accuracy efficiently and effectively.

After introducing the main components of our proposed healthcare system, we describe different steps in the client and the server as follows.

- In the client, data collection is implemented in three steps.
  - The portable smart device collects the patient’s bio-signal data continually from his or her body sensors. As shown in Section 2.3, 6 biosignals (HR, SBP, DBP, MBP, RR and SPO2) are considered in our system, which are used to detect 5 clinical events.
  - The processor in the device deals with the data based on a specific window (e.g. 10 minutes). The data in a window are defined as Time-Window Data Chunk (TWDC) and the processor a Time-Window Based Processor (TWBP). There are two main functions achieved in the processor: classifying the priority of TWDC and filtering TWDCs.
  - After filtering, the processor transmits time-series TWDCs and their priority to the server employed by a medical institution.

- In the server, data processing is achieved in three steps.
  - After receiving enough TWDCs within a specific observation window (e.g. 60 minutes), the server groups these TWDCs together and calculates the general priority by considering all TWDCs in the observation window. Then the server orders these grouped data into the waiting list based on the priority. Higher priority has a higher index, which means less time to wait for processing.
– CDSS predicts the medical conditions using these grouped TWDCs and sends the data to clinicians once an abnormal clinical event is detected. Then proper diagnosis decisions are provided to the patient.

– The TWDCs are stored in the database for backup and further knowledge discovery. The data are also used to update the priority classifier and generate medical knowledge which can be shared and improve the diagnosis prediction. The P2P learning process is explained in previous chapters.

5.4 Methodology

In this section, we introduce the medical information about vital biosignal and clinical events which are targeted in our system. Concretely, similar to Section 2.3, the system predict 5 events by 6 biosignals. Then criteria for assessing patient’s urgency are explained. Finally, we describe two key components of our system in details: the TWBP in the client and priority processor in the server.

5.4.1 Vital biosignals and clinical events

Our system considers numerical trend data of six vital biosignals shown in Table 5.1 to identify the early sign of clinical deterioration and assess treatment effects. The values of vital biosignals are various since different conditions (e.g. age and sex) of patients have an impact on these values. In order to provide a basic diagnosis, medical science defines a common normality range of each biosignals shown in Table 5.1.

Table 5.1: Vital biosignals and their normal range.

<table>
<thead>
<tr>
<th>Biosignal</th>
<th>Acronym</th>
<th>Normal range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Respiratory rate</td>
<td>RR</td>
<td>12-18 breaths per min</td>
</tr>
<tr>
<td>Blood oxygen saturation</td>
<td>SPO₂</td>
<td>95-100%</td>
</tr>
<tr>
<td>Heart rate</td>
<td>HR</td>
<td>60-100 beats per min</td>
</tr>
<tr>
<td>Systolic blood pressure</td>
<td>SBP</td>
<td>90-120 mmHg</td>
</tr>
<tr>
<td>Diastolic blood pressure</td>
<td>DBP</td>
<td>60-90 mmHg</td>
</tr>
<tr>
<td>Mean blood pressure</td>
<td>MBP</td>
<td>60-110 mmHg</td>
</tr>
</tbody>
</table>
Tachycardia and bradycardia are defined as rising and fall in HR respectively. The rise in blood pressure is known as hypertension and fall is called hypotension. Rise and fall in RR are called as tachypena and bradypena respectively. In addition, deficiency in SPO$_2$ is named hypoxia. Our system detects these clinical conditions happen at the same time and last for a specific time period, shown in Table 2.1. According to [23], it is reliable to use 1-hour data to predict the coming clinical event.

5.4.2 Criteria for data assessment

Our system employs a supervised learning algorithm to assess patients’ urgency and achieve a proper priority of the data. In order to label all samples, our system develops a method similar to National Early Warning Score (NEWS) [2] which is used in Emergency Department. As it is difficult to obtain the level of consciousness (LOC) of the patients automatically by smart devices, our system removes LOC to simplify our assessment method. In addition, since [48] shows the weak contribution of the systolic blood pressure (SBP) and temperature parameters to NEWS performance and suggests to remove the temperature, our method considers 4 out of 6 vital biosignals to assess the data urgency.

As shown in Table 5.2, these biosignals are RR, SPO$_2$, HR and SBP. The related scores are given based on their value thresholds. For example, if the value of RR is from 9 to 11, score 1 is provided by our method. It is important to note that all values are integer. In addition, the thresholds of SBP are different from NEWS. NEWS results 0 when the SBP value is from 110 to 219, but a normal patient SBP value should be always within 80 and 120. The score 0 of other 3 vital biosignals means the value is within the normal ranges. So in order to maintain the consistency, our system modifies the score thresholds of SBP when its value is larger than 110.
Table 5.2: Vital biosignals defining the triage.

<table>
<thead>
<tr>
<th>Sign</th>
<th>Score</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>≤ 8</td>
<td>9-11</td>
<td>12-20</td>
<td>21-24</td>
<td>≥ 25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPO2</td>
<td>≤ 91</td>
<td>91-92</td>
<td>94-95</td>
<td>≥ 96</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HR</td>
<td>≤ 40</td>
<td>41-50</td>
<td>51-90</td>
<td>91-110</td>
<td>111-130</td>
<td>≥ 131</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SBP</td>
<td>≤ 90</td>
<td>91-100</td>
<td>101-110</td>
<td>110-120</td>
<td>121-170</td>
<td>171-219</td>
<td>≥ 220</td>
<td></td>
</tr>
</tbody>
</table>

Similar to NEWS [15], our assessment method calculates the total score of all vital biosignals and classifies it into 4 priorities. The classification criteria are based on the waiting time which is defined in NEWS. The detail definition of all priorities is shown in in Table 5.3. In particular, if the total score is 0, the system considers discarding the data since all vital biosignals are normal. The range of each priority is used in evaluating a time sequential priority group, explained in the later section.

Table 5.3: Priority definition by score of vital biosignals.

<table>
<thead>
<tr>
<th>Priority</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Normal</td>
<td>Elective</td>
<td>Urgent</td>
<td>Emergency</td>
</tr>
<tr>
<td>Waiting time</td>
<td>-</td>
<td>≤ 1 h</td>
<td>≤ 0.5 h</td>
<td>0</td>
</tr>
<tr>
<td>Range</td>
<td>0</td>
<td>(0,1]</td>
<td>(1,2]</td>
<td>(2, 3]</td>
</tr>
<tr>
<td>Total Score</td>
<td>0</td>
<td>1-3</td>
<td>4-6</td>
<td>≥ 7 or Score of 3 in any sign</td>
</tr>
</tbody>
</table>

5.4.3 Time-Window Based Processor (TWBP)

Instead of sending data continually from the client to the server, our system deals with the TWDC in the TWBP and considers which TWDC is required for diagnoses. Figure 5.2 shows the workflow of a TWBP with four time-series biosignal data (HR, SBP, RR and SPO$_2$). The TWBP goes through 3 steps as follows.

- The features of a TWDC of all vital biosignals are extracted.
- The priority of the TWDC is detected by the priority classifier.
• According to its priority, the filter decides which TWDC is considerable for diagnoses.

Figure 5.2: The workflow of the time-window based processor. The priority classifier assesses the priority of a TWDC based on the mean feature vector from multi biosignals. Then the filter can decide whether the TWDC should be sent or discarded according to the sequential priorities.

5.4.3.1 Time-Window Data Chunk (TWDC) and its priority

The main role of the priority classifier in TWBP is to label TWDC with a corresponding data priority. The TWDC and its priority are introduced as follows, which are the foundation for our proposed system processing patients’ data.

Suppose the collection of discrete time-series data \( X \) of time length \( T \) is split into \( K \) windows \( (W_s) \) with equal size. There are \( N \) samples in each window \( (W) \) where \( N \) is equal to \( T/K \). So the time-series data can be considered as a sequence of a TWDC—\( X_1(t), X_1(t), X_1(t), \cdots, X_K(t) \). Then the samples in each window are used to construct features. For example, if a 10-minute window is employed in the system, 60-minute data can be divided into a sequence of 6 TWDCs (from T1 to T6 shown in Figure 5.2). In order to detect the TWDC priority, the mean value of each biosignal is obtained from samples in a window
based on Equation 5.1.

\[ f = \frac{\sum_{K=0}^{N} (X_K(t))}{N} \]  \hspace{1cm} (5.1)

Then all mean values from all vital biosignals are grouped as a feature vector—\((f_1, f_2, f_3, \cdots, f_s)\) where \(s\) is the total number of biosignals. As our assessment algorithm described above only works with integers, all components of the feature vector have to be changed to the value of a number rounded to the nearest integer \((f')\) which shown in

\[ f' = \lfloor f + 0.5 \rfloor \] \hspace{1cm} (5.2)

Similar to [23], a 10-minute window is adopted in our system. After extracting features of a TWDC, the trained priority classifier can detect the priority of the TWDC. More specifically, any kinds of machine-learning algorithms can be used as the priority classifier.

### 5.4.3.2 Data collection with the filter

In our proposed system, the filter in TWBP decides the considerable medical data based on its priority, which can optimize the data collection by reducing the data volume. Our proposed data collection processor is explained as follows.

Considering priority 0 means the values of all biosignals in a TWDC are in the normal range, if an abnormal TWDC existing in a buffer, the whole sequential TWDCs need to be transmitted for further prediction. Figure 5.3 shows an example of data collection. With a new TWDC coming into the TWBP every time period \((t)\), the blue buffer window slips to the left as the increase of \(t\). There are three statuses of a TWDC: send, pending and discard. Once there is the priority of a TWDC larger than 0 in the buffer, these 6 TWDCs are sent to the server. But if the priorities of the new TWDC and the rest are all 0s, the new TWDC is marked as pending. When the buffer moves out of the pending TWDC, the pending TWDC is discarded. Clearly, instead of simply sending TWDCs one by one, our algorithm adjusts the data collection process based on abnormal TWDCs.
Figure 5.3: An example of data transmission. One new TWDC comes to the list as the increase of \( t \). The blue buffer window includes all TWDCs processed by the system. In every \( t \), the status of TWDCs are shown. In particular, pending of a TWDC means the system need to obtain more TWDCs to decide whether the TWDC should be sent or discarded.

5.4.4 Priority processor

Similar to [23], in order to diagnose different clinical events, our system needs to consider sequential TWDCs instead of just a single TWDC, which is achieved in the priority processor shown in Figure 5.4.
As mentioned in the previous section, totally K time-series priorities within the window \( W \) are used to decide the general priority of these TWDCs. Instead of averaging all priorities, time weight factors defined in Equation 5.3 are introduced in our methods to maintain the importance of time.

\[
    w_i = \frac{T_i^2}{\sum_{i=1}^{K} T_i^2} \quad \text{and} \quad T_i = i \times W
\]

Denote \( P \) as the set of all priorities, then we have \( P = (p_1, p_2, \cdots, p_K) \). The general priority \( p' \) of continuous TWDCs is calculated as follows:

\[
    p' = P \cdot W = \sum_{i=1}^{i=K} p_i \times w_i
\]

The meaning of \( p' \) within a specific range is shown in Table 5.3. Then our system puts all extracted features of sequential TWDCs into the waiting list based on their priority, shown in Figure 5.4. The data with the highest priority are selected to do the prediction. The feature extraction process is described in Section 2.3. In each biosignal of a TWDC, 5 features are
extracted, which are mean, standard deviation, median, the number of increasing trends and decreasing trends. A feature matrix which includes all extracted features of sequential TWDCs is used to predict the coming clinical event.

5.5 Experimental evaluations and results

To evaluate the performance of our proposed algorithms, we preprocess the real medical data from the MIMIC-II [64] dataset and obtain our training and testing datasets. Recall that four vital biosignals (HR, SBP, RR and SPO$_2$) are used to evaluate four data priorities representing patients’ urgency. The performance of our proposed system is analyzed with three measurements as follows.

- The accuracy of priority classification: It shows the performance of three different classifiers categorizing data into four priorities.

- The efficiency of data collection: It is used to evaluate the performance of data collection in our proposed system in terms of the data volume and the data sending frequency. The experiments are conducted with 10 different ratios (from 0.1 to 1) of abnormal data in the system.

- The average waiting time: It is used to measure the performance in the waiting queue. The time-series data priorities are generated using two different discrete distributions: uniform distribution and normal distribution.

5.5.1 Data preprocessing

In order to evaluate the accuracy of data priority assessment, we consider 6 vital biosignals from MIMIC-II numeric dataset of MIT physiobank achieve, a large public dataset containing a large number of patients. Only the records containing at least 24 hours numerical trend data of these 6 biosignals are adopted. Most of the biosignals are sampled in one minute. Data sampled per second are converted to per minute sampling by averaging all values in a minute.
The data missing values over a long period and the noisy data are also filtered. Finally, 1023 records are obtained for the experiments.

As mentioned in the previous section, 4 out of 6 vital biosignals are used to identify the priority. These signals are HR, SBP, RR and SPO$_2$. In every biosignal we average all values in a 10-minute window, so each sample has 4 mean values. Then we apply our algorithm described above to label all samples. In order to balance the dataset, we randomly select 1500 samples of each priority. Considering totally 4 priorities are targeted, there are 6000 samples in the dataset. The dataset is normalized by the z-score linearly transformation. 70% of the samples form the training dataset and the rest the testing dataset, which are shown in Table 5.4.

Table 5.4: The preprocessed data with 4 data priorities.

<table>
<thead>
<tr>
<th>Data priority</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Normal</td>
<td>Abnormal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The number of training samples</td>
<td>1050</td>
<td>1050</td>
<td>1050</td>
<td>1050</td>
</tr>
<tr>
<td>The number of testing samples</td>
<td>450</td>
<td>450</td>
<td>450</td>
<td>450</td>
</tr>
<tr>
<td>Total</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
</tr>
</tbody>
</table>

5.5.2 Priority classification

The accuracies of different neural network classifier are shown in Table 5.5. In particular, the classifiers based on the decision tree are not considered in our experiments because they are good at classifying the data labels generated from rules. In our test, they can provide over 99% accuracy. Extreme learning machine (ELM) [36] is run in Matlab, which has 500 hidden neurons with the sigmoid activation function. The result classifiers are run in Weka 3.8 [78] with default settings. We use RBF kernel in SMO.

Table 5.5: The accuracy comparison among different learning algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Multilayer Perceptron</th>
<th>SMO</th>
<th>ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>71.89</td>
<td>66.83</td>
<td>80.6</td>
</tr>
<tr>
<td>Training Time (s)</td>
<td>2.82</td>
<td>13.53</td>
<td>1.09</td>
</tr>
</tbody>
</table>

ELM shows the best accuracy 80.6% among all candidate neural network classifier. The
confusion matrix from the classification result is shown in Table 5.6. Compared to the baseline (25%), ELM shows a significant improvement in classification accuracy.

Table 5.6: The confusion matrix after performing the classification. Here 80% accuracy is obtained.

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Elective</th>
<th>Urgent</th>
<th>Emergency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>445</td>
<td>136</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Elective</td>
<td>4</td>
<td>248</td>
<td>18</td>
<td>8</td>
</tr>
<tr>
<td>Urgent</td>
<td>1</td>
<td>66</td>
<td>380</td>
<td>63</td>
</tr>
<tr>
<td>Emergency</td>
<td>0</td>
<td>0</td>
<td>42</td>
<td>378</td>
</tr>
</tbody>
</table>

Except the overall accuracy and confusion matrix, different accuracy measures (precision, sensitivity, and specificity) for each priority are applied shown in Table 5.7. From this observation, we can see that the classification is not sensible to the data with the elective priority.

Table 5.7: The performance measure of each priority using ELM.

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Elective</th>
<th>Urgent</th>
<th>Emergency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision (%)</td>
<td>75.169</td>
<td>88.889</td>
<td>74.510</td>
<td>90.000</td>
</tr>
<tr>
<td>Sensitivity (%)</td>
<td>98.670</td>
<td>55.111</td>
<td>84.444</td>
<td>84.000</td>
</tr>
<tr>
<td>Specificity (%)</td>
<td>89.111</td>
<td>97.705</td>
<td>90.377</td>
<td>96.891</td>
</tr>
</tbody>
</table>

5.5.3 Data collection efficiency

In this experiment, we use Python to stimulate the data transmission process. In order to simplify the stimulation, we regard the priority from 1 to 3 as the abnormal priority 1. Based on 10 different abnormal data ratios from 0.1 to 1, we generate a list including 6000 binary values. The abnormal data ratio is calculated as

\[
R_{\text{abnormal}} = \frac{\text{The number of 1s in the list}}{\text{The length of the list}} \quad (5.5)
\]

. And the ratio of transmission data is calculated as

\[
P_{\text{data}} = \frac{\text{The number of data sent}}{\text{The length of the list}} \quad (5.6)
\]

. And the ratio of data sending requests is calculated as

\[
F_{sr} = \frac{\text{The number of sending requests}}{\text{The length of the list}} \quad (5.7)
\]
As shown in Figure 5.5, over 20% of the TWDC can be discarded in our proposed system when 10% TWDCs are abnormal. 99.8% of TWDCs are transmitted when the abnormal ratio is 40%. If over 40% abnormal TWDCs exist in the list, all TWDCs need to be transmitted.

![Graph showing the ratio of data sent to the server with different abnormal data ratios.](image)

**Figure 5.5:** The ratio of data sent to the server with different abnormal data ratios.

Similar to the data transmission percentage, when the list has 10% abnormal TWDCs, over half of the data sending requests can be saved, because the system groups the sequential TWDCs and sends once. And 0.2% of the requests are saved with 60% data abnormal ratio. When there are over 60% abnormal TWDC in the list, each TWDC is sent to the server one by one, which means the ration of sending requests is 1.
5.5.4 The average waiting time

In this part, we implement the experiment to evaluate the waiting time of patients’ records before prediction. We assume that the waiting list can contain $Q$ records for further prediction. After the prediction of all $Q$ records is finished, new $Q$ records come to the waiting list. The CDSS consumes only 1 record every time and spends $t$ on predicting the record. The stimulation is developed with Python. 3000 records with 3 abnormal priorities are generated from the uniform distribution and the standard normal distribution respectively. Thus, we have $3000/Q$ batches for prediction and the waiting time is calculated by averaging the waiting time of records with different priorities in all batches. When considering no priority, the system deals with every data based on First In First Out (FIFO). Otherwise, the system orders the data in the waiting list according to their priorities and then processes them one by one.
Experimental evaluations and results

(a) Priorities of all data follow the uniform distribution.

(b) Priorities of all data follow the standard normal distribution.

Figure 5.7: The waiting time of records with and without priority. Every record is processed in Time $t$. 
The results with different data distributions are shown in Figure 5.7. From the figures, we can observe that as the waiting list becomes longer our system increases the waiting time of the data with the lower priority (Elective).

In Figure 5.7(a), the waiting time without considering priority is fluctuant with the increase of the waiting list length, while the results with priority are stable regardless the changing of the length of the waiting list. In Figure 5.7(b), compared to the results without considering the data priority, our proposed method can significantly reduce the waiting time of urgent data.

Table 5.8 shows the comparison of the waiting time of the system with and without priority when the length of the waiting list is 10 and all data priorities follow the standard normal distribution. We assume that the system and clinicians spend 10 minutes on diagnosing each data. We can observe that the waiting time of all data with different priority is less than the maximum waiting time from NEWS clinical definition. But the system can optimize the processing flow of data based on data priority. 75% waiting time of urgent data can be reduced by our proposed method.

Table 5.8: Waiting time comparison. Assume that every data requires 10 min to get the prediction result. The length of the waiting list is 10 and data priorities follow the standard normal distribution.

<table>
<thead>
<tr>
<th></th>
<th>Normal</th>
<th>Elective</th>
<th>Urgent</th>
<th>Emergency</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEWS suggestion</td>
<td>-</td>
<td>$\leq 1\ h$</td>
<td>$\leq 0.5\ h$</td>
<td>0</td>
</tr>
<tr>
<td>CDSS with priority</td>
<td>-</td>
<td>10 min</td>
<td>1 min</td>
<td>0</td>
</tr>
<tr>
<td>CDSS without priority</td>
<td>-</td>
<td>26 min</td>
<td>4 min</td>
<td>0.2 min</td>
</tr>
</tbody>
</table>

5.6 Summary

In this chapter, we introduce an enhanced healthcare system with assessing data priority in order to optimize the data collection and the prediction in terms of data size and waiting time. Novel algorithms inspired by real-world clinical settings are developed to evaluate data priority which can represent patients’ urgency. Through the data priority algorithms, not only fewer data are collected from patients, but also complete medical information required in long-term accurate diagnoses is provided in our proposed system. Considering different distributions
of data priority in the real case, our extensive experiments show that our proposed method can improve the efficiency for data collection and can perform optimization of waiting times according to patients’ urgency. In the future, we are going to develop the personalized data assessment with context awareness, which can provide more accurate assessment with personal information.
Chapter 6

Conclusion

In this chapter, we draw the overall conclusion and provide recommendations for further research. The aim of this research was to explore a collaborative intelligent healthcare system with the medical knowledge sharing capability for e-healthcare. From the outset, we identified that when the healthcare system learns from more patients’ data, the accuracy and efficiency of diagnoses can be improved in three key processes as follows.

- The process of learning data: Update ML classifiers of the system with new knowledge, related to RQ-1 and RQ-2. The knowledge is generated from raw medical samples, which can be shared and learned effectively.

- The process of predicting data: Predict patients’ medical conditions with their newly arrived biosignal data, related to RQ-2 and RQ-3.

- The process of collecting data: Collect multi-biosignal data continuously from wearable sensors of patients, related to RQ-3.

In addressing three research questions, this thesis proposes a set of solutions to achieve a collaborative and effective healthcare system which can simplify the task of healthcare clinicians. The major motivations and contributions were described in Section 1, which are related to two significant limitations in existing prevalent e-healthcare systems. A novel collaborative healthcare system was described in Chapter 3, which provides the foundation for healthcare
institutions sharing their medical knowledge to improve the diagnostic accuracy. Chapter 4 enhanced our system to learn from data with uncertain data labels, making the system more robust and practical. In Chapter 5, we provide solutions to optimize the processes of collecting and predicting data by considering patients’ urgency.

The first research question (RQ-1) was addressed in Chapter 3. The main objective of this chapter is to develop a collaborative system where it is easy and efficient for every medical institution to share its medical knowledge with others and to learn from the knowledge to achieve better diagnostic accuracy. Our solution is explored based on the P2P networking technology and a fast ML classifier (ELM). Compared to the traditional centralized model, the P2P technology can decentralize the network users, which eliminates the dependency of centralized servers and optimizes the sharing process among medical intuitions. Moreover, the shared and learned knowledge is generated based on the fast ML classifier, which makes the sharing and learning processes efficient compared to using raw medical data. Extensive experiments were conducted in terms of diagnosis and network performance, showing 95% accuracy of diagnosing five clinical events using six biosignal data and 50% data-saving rate in sharing. The effects of different network sizes and topologies are also analyzed in terms of the learning time when all medical institutions finish updating with new knowledge. Overall, our solutions enabled efficient sharing and learning schemes to update e-healthcare systems by leveraging decentralized networking technology and a fast ML-based healthcare model, showing excellent performance in terms of the prediction accuracy, the time and data size in sharing with different network conditions.

Regarding the second research question (RQ-2), Chapter 4 enhanced the collaborative system which is proposed in Chapter 3. This chapter aims to improve the learning efficiency and prediction accuracy when the system encounters data with unknown labels, which also makes our previous solution more robust. The enhanced system adopts our novel ML algorithm which is named Collaborative ELM (CELM) with a Confidence Interval (CI). To achieve our proposed CELM with a CI, we enhanced the incremental learning technique with the collaborative learning capacity and adopted the data filtering technique in our system. More specifically, the enhanced incremental learning allows the system learning continuously from the knowledge...
generated from raw medical data or shared with other medical institutions, where the knowledge is likely to contain unknown data labels. A method of data filtering is also applied in the prediction process based on the maximum possibility where the true predictions lie, which can make the prediction more reliable and accurate. The extensive experiments show that our system provides the highest prediction accuracy with max 98% among five existing healthcare solutions. The effects of the incremental learning in our collaborative healthcare system are also analyzed related to the prediction accuracy and the filtering rate of prediction results. In summary, our solution enhances the learning and prediction processes by the technology related to incremental learning and data filtering, which makes the system more robust, efficient, reliable and accurate for healthcare diagnosis.

In Chapter 5, we addressed the third research question (RQ-3) and optimized the data processing workflow of our healthcare system in terms of data collection and data prediction. For the system optimization, we explored a data assessment algorithm to provide different data priorities by identifying patients’ urgency using their time-series multi-biosignal data in a specific observation window (e.g. ten minutes). With the data priority, the data volume in data collection can be reduced since only the valuable data (e.g. abnormal data) are collected for further analysis, and the waiting time before the prediction is optimized since the data with higher priority are processed in front of the data with lower priority. This can help the system to provide the diagnosis services to patients in a more proper time based on their urgency. We conducted extensive experiments with different possible receiving orders of data priority in a certain period. The system can save 20% volume of data in the collection and can reduce 75% waiting time of data with the highest priority before predicting. In short, our solution optimized the schemes of data collection and data prediction in our healthcare system by considering patients’ urgency, which can reduce the volume of data transmitted in the data collection and arrange a proper waiting order in the data prediction.

Overall, the significance of this research work is to provide a collaborative healthcare system to share medical knowledge, which helps medical intuitions to detect new diseases without touching the raw training disease samples. The developed system is efficient, practical and optimal in three key processes: data collection, learning and prediction, which can reduce
Limitations and Future Directions of Research

elderly patients’ deaths caused by chronic diseases (e.g. high blood pressure). Using our proposed model, better and more accurate diagnostic decision can be made by the efficient sharing and learning processes with knowledge generated from raw medical data in different medical institutions. Our solutions are easy and low-cost to build by taking the leverage of different state-of-art technologies related to networking, machine learning and distributed computing. The contribution of this thesis on our proposed collaborative healthcare system can be summarized as follows:

- The effective sharing and learning schemes based on knowledge generated from raw medical data
- The robust learning and prediction processes to deal with unlearned data labels
- The effective data collection process by considering patients’ urgency
- The optimal data prediction process with arranging waiting order of data based on patients’ urgency

6.1 Limitations and Future Directions of Research

The system and algorithms presented in this thesis can be used for problem-specific healthcare diagnostics services for patients in the hospital or at home. This research adopted and enhanced existing machine learning techniques to solve problems in the processes of learning, prediction and data collection. While the proposed techniques outperform related schemes, there remains scope for improvement in these approaches. Here we briefly discuss the limitations of our study and recommend some directions for future research.

Our system is not tested in a real-life environment, while considering a simulation and producing meaningful results. We mostly relied on simulated prototypes and focused on learning model development, validation and their performance evaluation. We used publicly available data of patients monitored in hospital beds and presumed a similar nature for real-life data when collected in a controlled manner from wearable sensors. In future, researchers can col-
laborate with medical institutions to test the system in a testbed environment for collecting real-time patients’ data and to evaluate the performance in the real-life environment.

In our research, we ignored other autonomous functional requirements of real-time healthcare systems such as low-level infrastructure of sensors, sensor failures, the reliability of communication between sensors and mobile devices, noise in sensor data and network fault management. Such requirements need an independent research investigation. We can say that our proposed model is a foundation that expands the scope of multiple research directions.

In Chapter 3, we emphasized building an advanced healthcare system to improve diagnostic accuracy by sharing medical knowledge. But our system uses the broadcast method to share our knowledge and our experiments cover two normal network topologies: Linear and 3D, which can be considered broadly. In future, in order to make our system more efficient and robust in terms of data transmission, we will explore some advanced network protocols (e.g. Gossip [45] and Chord [71]) and network topologies (e.g. Partially Decentralized Topology [85]).

In Chapter 4, we proposed a new machine-learning model to learn efficiently and continuously from samples with unknown data labels and to improve the accuracy by identifying where the true prediction results are likely to lie. In future, different activation functions (e.g. Kernel [6]) will be explored in the P2P learning model to improve training speed and accuracy. In addition, it is rare to see some diseases in the real world, leading that the number of training samples of rare diseases is not sufficient compared to the samples of common diseases. The difference in the data number available for different diseases, known as imbalanced data [87], also needs to be addressed in the future.

In Chapter 5, new algorithms were proposed to optimize the processes of collecting data and predicting data by introducing patients’ urgency, which can save the volume of data required to be transmitted and can reduce the waiting time before diagnosing. General clinical criteria are adopted to evaluate patients’ urgency. In future, we will explore an enhanced method to evaluate patients’ urgency based on their personal situations, which is important to the healthcare system. For example, it is supposed that most of the time the BP value of a hypertensive patient is higher than the normal. In addition, we will introduce content-aware
techniques [21] to evaluate the urgency more correctly, since different contents of patients (e.g. running or sleeping) have a significant impact on the value of their biosignals.

In conclusion, the contributions of this thesis include the development of a learning system and several algorithms for making the data processing efficient and reliable. The significance of this research is to provide efficient, practical and optimal solutions for medical institutions to deliver better clinical services for patients. Our research also proposes a future study in sharing knowledge to improve the accuracy of ML-learning classifiers instead of learning from the raw data, which can benefit different kinds of healthcare systems (e.g. AAL [35][44] and context-aware monitoring systems of patients [21]). In future, we will explore different advanced network protocols and topologies to improve the sharing performance of our P2P learning system. We will also improve the system with content-aware techniques to provide more accurate personal healthcare services. In addition, the issue of learning with unbalanced healthcare data will be addressed and we will test our P2P system in the real-life application.
Bibliography


