Sequential learning for extreme learning machine

Liang, Nanying

2006

Liang, N. (2006). Sequential learning for extreme learning machine. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/4601>

<https://doi.org/10.32657/10356/4601>

Nanyang Technological University

Downloaded on 26 Apr 2023 18:09:30 SGT

Sequential Learning for Extreme Learning Machine

Liang Nanying

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in fulfillment of the requirements for the degree of

Doctor of Philosophy

2005

Acknowledgements

I wish to express my sincere gratitude to all the people who have assisted me during the years of my postgraduate study at Nanyang Technological University. I am especially grateful to my supervisor, Associate Professor Paramasivan Saratchandran, for his professional guidance, constant support and trust. His insightful comment has been an inspiration for my work. Also, I wish to thank Assistant Professor Guang-Bin Huang and Professor Narasimhan Sundararajan for serving as my advisors and providing valuable insight and advice.

I owe a debt of gratitude to many members in Computer Control Lab. Technicians, Mohamed Ziauddin and Wong Chong Howe, provided me facilities for carrying out the project. I am especially grateful to my colleagues, Zhang Runxuan and Rong Haijun, for their help and partnership. I wish to thank my friends Wang Ting, Poh Siew Hua, Xu Jun, Zhang Sen, Zhou Jing, Liu Yang, Chen Cheng, and many others, for nurturing my happiness and peace of mind.

I wish to dedicate this thesis to my family for their endless love, encouragement, and support.

Table of Contents

Acknowledgements	i
Table of Contents	ii
Summary	v
List of Figures	vii
List of Tables	xii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Major Contribution of the Thesis	3
1.4 Organization of the Thesis	4
2 SLFNs: Function Approximation Theory and Constructive Methods	6
2.1 SLFNs and Function Approximation Theory	8
2.1.1 Universal Approximation Theorems for SLFNs	10
2.2 Constructive Methods	15

TABLE OF CONTENTS iii

2.3	Summary	17
3	Extreme Learning Machine	19
3.1	Unified SLFN with Multiple Linear Output Nodes	19
3.2	ELM theory	22
3.3	Comparison of ELM with Regularization Theory and SVMs	34
3.4	Algorithm Implementation	36
3.5	Summary	38
4	Online Sequential ELM	39
4.1	Motivation	39
4.2	Derivation of Online Sequential ELM	41
4.2.1	Online Sequential ELM Theory	42
4.2.2	Algorithm Implementation	58
4.3	Quantitative Comparison With Other Sequential Learning Algorithms	61
4.3.1	Mode of Learning	61
4.3.2	Type of Activation Functions	62
4.3.3	Model Selection	62
4.4	Performance Evaluation of OS-ELM	63
4.4.1	Model Selection in Simulations	64
4.4.2	OS-ELM Performance Evaluation on Regression Problems .	68
4.4.3	OS-ELM Performance Evaluation on Classification Problems	79
4.4.4	OS-ELM Performance Evaluation on Mackey-Glass Time Series Prediction Problem	89
4.5	Summary	94

TABLE OF CONTENTS

iv

5 OS-ELM Application in BCI	98
5.1 BCI System Outline	100
5.1.1 Preprocessing	100
5.1.2 Feature Extraction	101
5.1.3 Classification/Detection	103
5.2 Performance Evaluation	104
5.2.1 EEG Database	104
5.2.2 Preprocessing & Feature Extraction	104
5.2.3 Classification	107
5.3 Summary	119
6 Conclusion and Recommendations	120
6.1 Conclusion	120
6.2 Recommendations for Further Research	122
Author's Publications	124
Bibliography	125
Appendix	133

Summary

The approximation capability of feedforward neural networks has been deeply studied, and it has been shown that the neural network with as few as a single hidden layer is a universal approximator. We refer such neural networks as Single hidden Layer Feedforward Neural networks (SLFNs). Depending on the kind of activation function used in the hidden layer, two popular structures of SLFNs can be recognized in the literature: SLFNs with ridge activation function, and SLFNs with radial basis activation function: radial basis function neural networks (RBF-NNs).

By recognizing the need for sequential learning in practice, we lay the foundation of the thesis on sequential learning algorithm design for SLFNs. Traditional sequential algorithms for training SLFNs with ridge activation function are referred to as stochastic BP-based algorithms. For training RBF-NNs, they are referred to as RAN-based algorithms. However, the intrinsic drawbacks of these sequential learning algorithms prohibit their applications into complicate and large-scale problems. This motivates us to propose a novel method, which is referred to as Online Sequential Extreme Learning Machine (**OS-ELM**) in the thesis. **OS-ELM** is derived from batch Extreme Learning Machine (**ELM**) on the basis of recursive least-squares (RLS) algorithm. The resulted sequential algorithm trains both SLFNs with ridge function and RBF-NNs in a unified framework. Additionally it can operate in the learning mode of one-by-one as well as chunk-by-chunk.

The implementation of **OS-ELM** is composed of two phases, viz., initialization and update phase. In the initialization phase of the algorithm, the values for the hidden unit parameters of a SLFN are randomly assigned. In the update phase,

only the output unit parameters are required to adjust iteratively on the basis of current sample(s).

Simulations on benchmark problems, including regression, classification, and time series prediction, are carried out. The performance of **OS-ELM** is evaluated and compared with other sequential learning algorithms, including SGBP, RAN, RANEKF, MRAN, GAP-RBF, and GGAP-RBF. The results show that **OS-ELM** achieves improved generalization performance with reduced computational complexity compared with other sequential learning algorithms. Moreover, the learning evolution patterns of **OS-ELM** indicate that the algorithm achieves faster convergence than SGBP.

The advantages of **OS-ELM** lead to our proposal for the application of the algorithm into the online classifier design in brain computer interfaces (BCIs). Simulation on a well-known EEG database consisting of the cognitive mental task classification is performed. The performance of **OS-ELM** is evaluated and compared with the recently developed FGAP-RBF, an improved RAN-based algorithm. The results show that **OS-ELM** produces better testing accuracy with less training time as compared to FGAP-RBF.

List of Figures

2.1	A Single hidden Layer Feedforward Network (SLFN) with a single linear output unit	7
3.1	A Single hidden Layer Feedforward Network (SLFN) with multiple linear output nodes	20
4.1	An example of model selection for OS-ELM (Sigmoid) and SGBP in the learning mode of one-by-one in the Auto-MPG case: Average RMSEs are showed as a function of the number of hidden units. . .	66
4.2	The learning evolution patterns for OS-ELM (with 25 hidden units using sigmoidal activation function) and for SGBP (with 13 hidden units) in the learning mode of one-by-one on a single trial in model selection for the Auto-MPG case.	66
4.3	The learning evolution patterns for OS-ELM (with 25 hidden units using sigmoidal activation function) and for SGBP (with 13 hidden units) in the learning mode of one-by-one on a single trial in the Auto-MPG case.	71
4.4	The learning evolution patterns for OS-ELM (with 25 hidden units using sigmoidal activation function) and for SGBP (with 11 hidden units) in the learning mode of one-by-one on a single trial in the Abalone case.	72

4.5	The learning evolution patterns for OS-ELM (with 50 hidden units using sigmoidal activation function) and SGBP (with 9 hidden units) in the learning mode of one-by-one on a single trial in the California Housing case.	72
4.6	The learning evolution patterns of OS-ELM (with 25 hidden units using sigmoidal activation function) in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Auto-MPG case.	77
4.7	The learning evolution patterns of OS-ELM (with 25 hidden units using sigmoidal activation function) in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Abalone case.	78
4.8	The learning evolution patterns of OS-ELM (with 50 hidden units using sigmoidal activation function) in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the California Housing case.	78
4.9	The learning evolution patterns for OS-ELM (with 180 hidden units using sigmoidal activation function) and for SGBP (with 80 hidden units) in the learning mode of one-by-one on a single trial in the Image Segment case.	82
4.10	The learning evolution patterns for OS-ELM (with 400 hidden units using sigmoidal activation function) and for SGBP (with 25 hidden units) in the learning mode of one-by-one on a single trial in the Satellite Image case.	83
4.11	The learning evolution patterns for OS-ELM (with 200 hidden units using sigmoidal activation function) and for SGBP (with 12 hidden units) in the learning mode of one-by-one on a single trial in the DNA case.	83

4.12	The learning evolution patterns for OS-ELM with 180 hidden units using sigmoidal activation function in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Image Segment case.	87
4.13	The learning evolution patterns for OS-ELM with 400 hidden units using sigmoidal activation function in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Satellite Image case.	88
4.14	The learning evolution patterns for OS-ELM with 200 hidden units using sigmoidal activation function in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the DNA case.	88
4.15	The first 1000 training targets on the generated Mackey-Glass time series.	90
4.16	The learning evolution patterns for OS-ELM with 120 hidden units using sigmoidal and gaussian activation functions in the learning mode of one-by-one on a single trial in the Mackey-Glass time series prediction problem.	92
4.17	The testing targets (circles) on the generated Mackey-Glass time series and predicted values (dots) by OS-ELM with 120 hidden units using the sigmoidal function in the learning mode of one-by-one.	92
4.18	The learning evolution for OS-ELM with 120 hidden units using sigmoidal activation function in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Mackey-Glass time series prediction problem.	95
5.1	The diagram shows the general scheme of a BCI System.	99

5.2	A diagram shows four categories of EEG waves in the typical frequency bands: Beta, Alpha, Theta, and Delta.	102
5.3	A flowchart shows the procedure for ocular artifact removal and feature extraction from each trial of a task.	105
5.4	An example of model selection for OS-ELM (Sigmoid) and OS-ELM (Gauss) in session 1: Average accuracies are showed as a function of the number of hidden units.	109
5.5	A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session1.	113
5.6	A diagram shows the learning evolution for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session2.	114
5.7	A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session3.	114
5.8	A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session4.	115
5.9	A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session5.	115

5.10 A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session8.	116
5.11 A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session9.	116
5.12 A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session10.	117
5.13 A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session11.	117
5.14 A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session12.	118
5.15 A diagram shows the learning evolution patterns for OS-ELM (Sigmoid), OS-ELM (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session13.	118

List of Tables

4.1	Specifications of the benchmark problems.	63
4.2	Performance comparison between OS-ELM and other algorithms in the learning mode of one-by-one on the regression problems. The number of hidden units for RAN, RANEKF, MRAM, GAP-RBF, and GGAP-RBF is the average over 50 trials.	69
4.3	Performance comparison between batch ELM and OS-ELM in the learning mode of chunk-by-chunk on the regression problems. [10,30] means that the number of samples is randomly picked in the range of 10 to 30 for each iteration.	74
4.4	Performance comparison between OS-ELM and other algorithms in the learning mode of one-by-one on the classification problems. The number of hidden units MRAN and GAP-RBF is the average over 50 trials.	81
4.5	Performance comparison between batch ELM and OS-ELM in the learning mode of chunk-by-chunk on the classification problems. [10,30] means that the number of samples is randomly picked in the range of 10 to 30 for each iteration.	84
4.6	Performance comparison between OS-ELM and other algorithms in the learning mode of one-by-one on the Mackey-Glass time series prediction problem.	90

LIST OF TABLES

xiii

4.7	Performance comparison between batch ELM and OS-ELM in the learning mode of chunk-by-chunk on the Mackey-Glass time series prediction problem. [10,30] means that the number of samples is randomly picked in the range of 10 to 30 for each iteration.	93
5.1	The patterns of brain waves are related to the mental states and tasks	102
5.2	Performance comparison between OS-ELM and FGP-RBF on the cognitive mental task classification.	110

Chapter 1

Introduction

1.1 Motivation

The universal approximation capability of Single hidden Layer Feedforward Neural networks (SLFNs) has been recognized and widely exploited in various off-line and on-line applications. Hence, there arises the need for effective training methods for SLFNs. The aim of this thesis is to develop a novel sequential learning algorithm for SLFNs that achieves good generalization performance with fast convergence speed and inexpensive computation for on-line applications, such as time series prediction and online classifier design in brain computer interfaces (BCIs).

Traditional sequential learning algorithms for SLFNs can be classified into two categories depending on the type of activation functions used in the hidden layer of SLFNs. One kind of these algorithms are based on stochastic backpropagation (BP) techniques for training feedforward neural networks with differential ridge activation function, e.g., sigmoidal function. These algorithms consist of minimizing an instantaneous cost function. In the early work, one of the representative methods was stochastic gradient descent backpropagation (SGBP) [55], in which first order information of the instantaneous cost function is used for adjusting the parameters of the neural network. Although it is efficient for implementation, such

method is characterized by slow convergence speed in practical applications [18]. Later on, the techniques exploring second order information have been devised, such as recursive Marquardt-Levenberg (ML) algorithm [44, 3]. Although the convergence speed is accelerated, the implementation efficiency is sacrificed for the estimation/approximation of the inverse Hessian matrix.

Another kind of methods is those based on resource allocation network (RAN) techniques for training RBF-NNs (SLFNs with radial basis activation function). These RAN-based methods include RAN [50], RANEKF [34], MRAN [66, 67], GAP-RBF [27], and GGAP-RBF [28]. As shown by numerical experiments with large-scale applications [28], these methods are generally characterized by expensive computational complexity. Moreover, there are many control parameters involving with the use of RAN-based algorithms, which makes the training process delicate to accomplish in success.

In the effort to overcome the drawbacks mentioned above, a novel sequential learning algorithm, referred to as Online Sequential Extreme Learning Machine (**OS-ELM**), is developed for training SLFNs in this thesis. The algorithm considers both SLFNs with ridge function and RBF-NNs in a unified framework. Moreover, this algorithm achieves good generalization performance with fast convergence speed and computational savings. Additionally, this algorithm is characterized by easy usage and flexible learning mode. For the use of the algorithm, only one model parameter is required to be tuned by the designer. And the samples learned in each iteration is not fixed to one but can be a chunk of varying size. This results in the one-by-one and chunk-by-chunk learning mode.

1.2 Objectives

The primary goal of this thesis is to devise a sequential learning algorithm for SLFNs and to evaluate its performance in various applications. More specifically, the objectives of this thesis can be summarized as follows:

- 1) To derive **OS-ELM** from batch **ELM**. Batch **ELM** has been proposed for training both SLFNs with ridge function and radial basis function in batch mode. Previous experimental results have shown its efficiency in a wide range of applications. Therefore, it is a choice from which a novel sequential learning algorithm may be developed.
- 2) To carry out the performance evaluation and comparison of **OS-ELM** with other contemporary sequential learning algorithms in the applications of regression, classification, and time series prediction.
- 3) To apply **OS-ELM** into the online classifier design in brain computer interface (BCI) domain. BCI is a potential communication channel for those with severe neuromuscular disorder. Its kernel component is an online classifier. Hence, we propose the application of **OS-ELM** in the implementation of BCI systems.

1.3 Major Contribution of the Thesis

Major contribution achieved in this thesis are as follows:

- 1) **OS-ELM** is derived from batch **ELM** on the basis of Recursive Least-Squares (RLS) algorithm. The resulted algorithm can sequentially train both SLFNs with ridge function and SLFNs with radial basis function in a unified framework.
- 2) **OS-ELM** is theoretically shown to be able to produce the same mapping function as batch **ELM** given the same training samples and initial model of a SLFN.
- 3) **OS-ELM** can learn in the sequential mode of one-by-one as well as chunk-by-chunk. Additionally, the chunk size is variable in each update iteration.
- 4) A comprehensive evaluation of **OS-ELM** is performed on seven benchmark problems, which include applications from the areas of regression, classification, and time series prediction. For the purpose of comparison, the performances of

other sequential learning algorithms, including SGBP, RAN, RANЕКF, MRAN, GAP-RBF, and GGAP-RBF, are evaluated on the same problems. The simulation results show that **OS-ELM** achieves improved generalization performance with reduced computational complexity, fast convergence speed, as well as flexible learning mode.

- 5) An online classifier in an EEG-driven BCI system has been successfully implemented by **OS-ELM**. The performance of **OS-ELM** is evaluated and compared with FGAP-RBF session by session using a well-known EEG database of cognitive mental task classification. The numerical results show that **OS-ELM** outperforms FGAP-RBF in terms of testing accuracy and training time per sample. Hence it is indicated that **OS-ELM** is a potential algorithm for the online classifier design of BCIs.

1.4 Organization of the Thesis

The main body of the thesis consists of six chapters. Chapter 2 provides an introduction to SLFNs and the theoretical foundation of their approximation capability. The representative constructive algorithms both in batch and sequential mode are reviewed as well. Chapter 3 first presents the theory of batch **ELM** in the principles of exact and approximate mapping. Then its implementation is provided and related issues are discussed.

The first part of Chapter 4 supplies the deduction of **OS-ELM** both in the learning mode of one-by-one and chunk-by-chunk. The inherent characteristics of **OS-ELM** are exploited thoroughly. It is theoretically proven that, given the same training samples and initial model of a SLFN, **OS-ELM** realizes the identical input-output mapping function with batch **ELM**. Next, the implementation details is provided and related issues are discussed. The last part of Chapter 4 is dedicated to a comprehensive performance evaluation of **OS-ELM** both in the learning mode of one-by-one and chunk-by-chunk on seven benchmark problems, which include

applications from the areas of regression, classification, and time series prediction. For comparison purpose, we carry out the simulation of other sequential learning algorithms, including SGBP, RAN, RANEKF, MRAN, GAP-RBF, and GGAP-RABF. Further, the online learning characteristics of **OS-ELM** and SGBP are illustrated by the learning evolution patterns on a single trial in these applications.

Chapter 5 proposes the application of **OS-ELM** in the online classifier design in BCI systems. Section 5.1 describes BCI systems in the general scheme, with the emphasis on the need for an online classifier in BCIs. Section 5.2 lays down the experiments with **OS-ELM** and FGAP-RBF on a well-known EEG database of cognitive mental task classification.

Chapter 6 contains the conclusion of this thesis and recommendations for further research. Finally, Appendix A provides the details of the derivation of RLS algorithm in a chunk-by-chunk context.

Chapter 2

SLFNs: Function Approximation Theory and Constructive Methods

In the past decades, Single hidden Layer Feedforward Networks (SLFNS) with ridge activation function or radial basis activation function, have been intensively investigated. It has been shown that such layered feedforward networks can uniformly approximate continuous functions [20, 21, 22, 46, 47]. Thus their application attracts much attention in various areas, such as regression, prediction, classification, etc. Therefore the efficient implementation of SLFNs is an important issue. Several constructive methods have been proposed for SLFNs in the literature [41, 57, 39]. Here the constructive method is an algorithm that derives a SLFN from the constructive proof of function approximation.

In this chapter we first present the theory of function approximation using SLFNs followed by their constructive methods.

Figure 2.1: A Single hidden Layer Feedforward Network (SLFN) with a single linear output unit

2.1 SLFNs and Function Approximation Theory

A SLFN is composed of three layers as depicted in Figure 2.1. The first layer consists of d input nodes, which just take the inputs x_i ($i = 1, 2, \dots, d$) as their outputs. Additionally, a virtual node with a unit input is introduced. The second layer is a hidden layer and composed of n units. Each hidden unit is responded locally to their own activation functions and connected directly to all the nodes in the input layer. And the connections of the hidden units to the virtual input node are presented by dotted lines. The activation function of an individual hidden unit is nonlinear and defined by $G : \mathbb{R}^d \rightarrow \mathbb{R}$. Accordingly, the output of the i^{th} hidden unit is given by $G(\mathbf{x}, \mathbf{a}_i, b_i)$, where $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ are called the hidden unit parameters of a SLFN. Without the loss of generality, we assume the output layer consists of a single linear unit, which is fully connected to the hidden layer as well. Therefore, the output of a SLFN is a linearly weighted sum of the outputs of the hidden units and expressed as:

$$\mathcal{N}_{G,n}(\mathbf{x}) = \sum_{i=1}^n \beta_i G(\mathbf{x}, \mathbf{a}_i, b_i) \quad (2.1)$$

where the subscripts G and n represents the nonlinear activation function and the number of hidden units in the hidden layer, respectively. $\{\beta_i\}_{i=1}^n$ are called the output unit parameters of a SLFN. Depending upon the type of activation functions used in the hidden layer of a SLFN, the hidden unit parameters of the network, $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$, can be explained in their own individual ways.

Generally, the nonlinear activation functions used in SLFNs can be classified into ridge functions and Radial Basis Functions (RBFs). A ridge function has the following formula:

$$G(\mathbf{x}, \mathbf{a}, b) = \Psi(\mathbf{a}^T \mathbf{x} + b) \quad (2.2)$$

where T denotes the transpose operator, $\mathbf{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ is referred to as the direction and the threshold of the ridge function, respectively. $\Psi(\cdot)$ is a smooth nonlinear function. One of the commonly used examples is the logistic function defined by: $\varphi(n) = \frac{1}{1+e^{-n}}$.

When ridge activation functions are used in the hidden layer, a set of input-output mapping functions is realized by the model of the SLFN and can be expressed as:

$$\mathcal{N}_{\Psi,n}(\mathbf{x}) = \sum_{i=1}^n \beta_i \Psi(\mathbf{a}_i^T \mathbf{x} + b_i) \quad (2.3)$$

where $\mathbf{a}_i \in \mathbb{R}^d$ is the input weight vector connecting the input layer to the i^{th} hidden unit. $b_i \in \mathbb{R}$ is the bias applied to the i^{th} hidden unit. $\beta_i \in \mathbb{R}$ is the output weight connecting the i^{th} hidden unit to the output layer.

For a RBF hidden unit, the activation function is:

$$G(\mathbf{x}, \mathbf{a}, b) = \Phi\left(\frac{\|\mathbf{x} - \mathbf{a}\|}{b}\right) \quad (2.4)$$

where $\|\cdot\|$ denotes a norm that is usually Euclidean. $\mathbf{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the centroid and smoothing factor of a RBF, respectively. An example of RBFs is the multivariate Gaussian function defined by $Gauss(\mathbf{x}, \mathbf{a}, b) = e^{-\frac{\|\mathbf{x} - \mathbf{a}\|^2}{b}}$.

A SLFN with RBF activation functions, which is known as a RBF neural network (RBF-NN) in the literature, realizes a set of input-output mappings defined by:

$$\mathcal{N}_{\Phi,n}(\mathbf{x}) = \sum_{i=1}^n \beta_i \Phi\left(\frac{\|\mathbf{x} - \mathbf{a}_i\|}{b_i}\right) \quad (2.5)$$

where $\mathbf{a}_i \in \mathbb{R}^d$ and $b_i \in \mathbb{R}$ denote the i^{th} hidden unit's centroid and smoothing factor, respectively. $\beta_i \in \mathbb{R}$ is the output weight connecting the i^{th} hidden unit to the output layer.

2.1.1 Universal Approximation Theorems for SLFNs

Approximation capabilities of standard SLFNs have been well studied. Many researchers provided important results dealing with different types of approximation in certain general spaces, such as $L^p(R^d)$ approximation with μ compact support for $p \in [1, \infty)$, $C(X)$ approximation in the compact subset X of R^d , smooth approximation, etc. Before we state these results, it will be helpful to introduce some important terminology and concepts in functional analysis.

Let R denote the set of real numbers. For any positive integer d , R^d denotes the linear space of real d -vectors with norm $\|\cdot\|$. For any $p \in [1, \infty)$, $L^p(R^d)$ represents the space of all $f : R^d \rightarrow R$ such that f is p^{th} power integrable. The L^p norm of f is defined by $\|f\|_p = (\int_{\mathbf{X}} |f(\mathbf{x})|^p dx)^{1/p}$. $L^\infty(R^d)$ denotes the space of all $f : R^d \rightarrow R$ such that f is essentially bounded and with the so-called uniform norm defined as $\|f\|_\infty = \sup |f(x)|$. The relationship between the L^p and uniform norm is of $\lim_{p \rightarrow \infty} \|f\|_p = \|f\|_\infty$. $C(X)$ denotes the space of all continuous functions $f : X \rightarrow R$, where X is a compact subset of R^d .

The concepts of closeness and denseness are frequently used in the analysis of function approximation. They determine how to measure the accuracy of approximation and can be explained as follows. Closeness of two functions, such as s and t , is measured by a metric or distance function, $\rho(s, t)$. The format of ρ varies with specific space to be dealt with. For example, given a normed linear space X with norm $\|\cdot\|$, a metric on X then can be defined by $\rho(s, t) = \|s - t\|$. From this point of view, the metric ρ is said to be induced by the norm $\|\cdot\|$. Closeness of one class of functions to another class is described by the concept of denseness. A subset S is said to be ρ -dense in a subset T if for every $\epsilon > 0$ and for every $t \in T$, there exists an $s \in S$ such that $\rho(s, t) < \epsilon$. It means that an element of S can approximate an element of T to any desired degree of accuracy (the sense of universal approximation). Applying this concept to the context of function approximation using SLFNs, we may let S and T , respectively, correspond to the set of functions implemented by SLFNs with some ridge or RBF hidden units and the

set of functions to be approximated. On the basis of the above terminology and concepts, we introduce the universal approximation theorems for SLFNs.

Depending on the type of approximation on some general space, mild conditions are imposed on the activation function and then universal approximation capabilities of SLFNs are proven. For SLFNs with ridge nodes, Hornik *et al.* [21] have given their universal approximation capabilities on the spaces of $C(X)$ and $L^p(R^d)$ using the following theorems:

Theorem 2.1. *If Ψ is continuous, bounded and nonconstant. Then with $\{(\mathbf{a}_i, b_i) | \mathbf{a}_i \in R^d, b_i \in R\}_{i=1}^n$, $\mathcal{N}_{\Psi,n}(\mathbf{x}) = \sum_{i=1}^n \beta_i \Psi(\mathbf{a}_i^T \mathbf{x} + b_i)$ is dense in $C(X)$ with respect to uniform topology for any compact subset X of R^d .*

Theorem 2.2. *If Ψ is bounded and nonconstant. Then with $\{(\mathbf{a}_i, b_i) | \mathbf{a}_i \in R^d, b_i \in R\}_{i=1}^n$, $\mathcal{N}_{\Psi,n}(\mathbf{x}) = \sum_{i=1}^n \beta_i \Psi(\mathbf{a}_i^T \mathbf{x} + b_i)$ is dense in $L^p(R^d)$ for any finite measures μ on R^d with $p \in [1, \infty)$.*

The proofs of Theorem 2.1 and 2.2 are similar and both make use of the Hahn-Banach and Riesz representation theorems and the definition of discriminatory function. Therefore we first introduce these two theorems and the definition. And the proofs of Theorem 2.1 and 2.2 are followed.

The Hahn-Banach theorem, named for Hans Hahn and Stefan Banach, is a central tool in functional analysis and may be stated as [54]:

Theorem 2.3. *If M is a subspace of a normed linear space L and if f is a bounded linear functional on M , then f can be extended to a bounded linear functional F on L such that $\|F\| = \|f\|$.*

This theorem allows one to extend bounded linear operators on a subspace of some vector space to the whole space. An important consequence of the theorem is the following corollary [54]:

Corollary 2.1. *Let M be a subspace of a normed linear space L and let $x_0 \in L$. Then x_0 is in the closure $\bar{M}$ of M if and only if there is no bounded linear functional Λ on L such that $\Lambda(x) = 0$ for all $x \in M$ but $\Lambda(x_0) \neq 0$.*

The format of the functional Λ in the above corollary is given in the Riesz representation theorem. Depending on the space of interest, the Riesz representation theorem have different expressions. For the case of Λ on the space $C(X)$, it states:

Theorem 2.4. *Let X be a compact space of R^d . For any nonzero continuous linear functional Λ on $C(X)$, there exists a nonzero finite signed measure μ on X such that $\Lambda(f) = \int_X f(x)d\mu(x)$ for every f in $C(X)$.*

When consider the case of Λ on the space $L^p(R^d)$, it has:

Theorem 2.5. *Given any nonzero continuous linear functional Λ on $L^p(R^d)$, there exists a nonzero finite signed measure μ on R^d such that $\Lambda(f) = \int_{R^d} f(x)d\mu(x)$ for every f in $L^p(R^d)$.*

Further we introduce the definition of discriminatory function proposed by Cybenko in [11]:

Definition 2.1. *Ψ is said discriminatory if for a Borel measure μ on $M(R^d)$, $\int_{R^d} \Psi(\mathbf{a}^T \mathbf{x} + b)d\mu(\mathbf{x}) = 0$ for all $\mathbf{a} \in R^d$ and $b \in R$ implies that $\mu = 0$.*

In the definition, $M(R^d)$ denotes the space of all Borel measurable functions $f : R^d \rightarrow R$. $L^p(R^d)$ and $C(X)$ are obviously subsets of $M(R^d)$. Hence, the definition is valid for those of p^{th} power integrable and continuous functions.

Cybenko [11] has proven a wide range of types of functions are discriminatory, such as bounded, measurable (in particular, continuous) sigmoidal functions, the sine and cosine functions, exponential functions, etc. Hornik [21] further extended Cybenko's result to which any bounded and nonconstant function is discriminatory.

On the basis of the above theorems and definition, we can come to the proofs of Theorem 2.1 and Theorem 2.2:

Proof of Theorem 2.1. Given that Ψ is continuous, bounded and nonconstant. For some compact subset X of R^d , we assume that $\mathcal{N}_{\Psi,n}$ is not dense in $C(X)$. Then by the Corollary 2.1 and the Riesz representation theorem, there exists a nonzero

finite signed measure μ on X such that $\int_X \Psi(\mathbf{a}^T \mathbf{x} + b) d\mu(\mathbf{x}) = 0$ for all $\mathbf{a} \in R^d$, $b \in R$. However, Ψ is discriminatory. This implies that $\mu = 0$ which contradicts the assumption. The proof of Theorem 2.1 is complete. $\square$

Proof of Theorem 2.2. Similarly, for the activation function Ψ is bounded and non-constant and for some μ on R^d , we assume that $\mathcal{N}_{\Psi,n}$ is not dense in $L^p(R^d)$. Then proceeding as in the proof of Theorem 2.1, we have a nonzero finite signed measure μ on R^d such that $\int_{R^d} \Psi(\mathbf{a}^T \mathbf{x} + b) d\mu(\mathbf{x}) = 0$ for all $\mathbf{a} \in R^d$, $b \in R$. On the other hand, since Ψ is bounded and nonconstant, it is discriminatory which implies that $\mu = 0$. The proof of Theorem 2.2 is complete. $\square$

Some primitive results on the approximation using SLFNs with a single universal bias (i.e., all biases are identical) have been provided in [22]. For the type of smooth approximation (i.e., continuous function and its derivatives approximation), see [21].

Paralleling to SLFNs with ridge activation function, RBF neural networks exhibit universal capabilities on the $C(X)$ and $L^p(R^d)$ approximation in their own ways. These results are provided by Park & Sandberg in the following theorems [47]:

Theorem 2.6. *Let $\Phi : X \rightarrow R$ be a continuous and integrable function such that $\int_X \Phi(x) dx \neq 0$. Then with $\{(\mathbf{a}_i, b_i) | \mathbf{a}_i \in R^d, b_i \in R^+\}_{i=1}^n$, $\mathcal{N}_{\Phi,n}(\mathbf{x}) = \sum_{i=1}^n \beta_i \Phi(\frac{\|\mathbf{x} - \mathbf{a}_i\|}{b_i})$ is dense in $C(X)$ with respect to uniform topology for any compact subset X of R^d .*

Theorem 2.7. *Let Φ be an integrable function such that $\int_{R^d} \Phi(x) dx \neq 0$ and $\int_{R^d} |\Phi(x)|^p dx < \infty$. Then with $\{(\mathbf{a}_i, b_i) | \mathbf{a}_i \in R^d, b_i \in R^+\}_{i=1}^n$, $\mathcal{N}_{\Phi,n}(\mathbf{x}) = \sum_{i=1}^n \beta_i \Phi(\frac{\|\mathbf{x} - \mathbf{a}_i\|}{b_i})$ is dense in $L^p(R^d)$ for any finite measures μ on R^d with $p \in [1, \infty)$.*

The universal approximation theorems for SLFNs with RBF hidden nodes are similar with those for SLFNs with ridge hidden nodes. We infer that their proofs can be analogous to the proofs for Theorem 2.1 and 2.2 by making use of an argument of *reductio ad absurdum* together with the application of the Hahn-Banach and Riesz representation theorems. The definition of discriminatory function is

slightly revised for the case when RBF function with some positive smoothing factor ($b \in R^+$) is used [47].

Proof of Theorem 2.6. Given Φ is a continuous and integrable function such that $\int_X \Phi(x)dx \neq 0$. For the compact subset X of $C(X)$, we assume that $\mathcal{N}_{\Phi,n}$ is not dense in $C(X)$. Then by the corollary of Hahn-Banach theorem and the Riesz representation theorem, there exists a nonzero finite signed measure μ on X such that $\int_X \Phi(\frac{\|\mathbf{x}-\mathbf{a}\|}{b})d\mu(x) = 0$ for any $\mathbf{a} \in R^d$ and $b \in R^+$. While Φ is discriminatory which implies $\mu = 0$. The proof of Theorem 2.6 is complete. $\square$

Proof of Theorem 2.7. Similarly, for the activation function Φ is integrable such that $\int_X \Phi(x)dx \neq 0$ and $\int_X |\Phi(x)|^P dx < \infty$ and for some measure μ on R^d , we assume that $\mathcal{N}_{\Phi,n}$ is not dense in $L^p(R^d)$. Then again we have a nonzero finite signed measure μ on R^d such that $\int_{R^d} \Phi(\frac{\|\mathbf{x}-\mathbf{a}\|}{b})d\mu(x) = 0$ for any $\mathbf{a} \in R^d$ and $b \in R^+$. While Φ is discriminatory which implies $\mu = 0$. The proof of Theorem 2.7 is complete. $\square$

Universal approximation capabilities of RBF neural networks using one uniform smoothing factor (i.e., all smoothing factors in the hidden RBF activation functions are same) also have been investigated by Park & Sandberg in [46] [47].

The significant contribution of the uniform approximation theorems lies in that they provide the fundamental existing evidence of the universal approximation capabilities of SLFNs. However, how to implement such SLFNs is not answered by these theorems, such as the determination of the sufficient number of hidden nodes required in the theorems, the training of SLFNs given a set of training data to a prescribed accuracy of approximation, etc.

It needs to note that the extension to the approximation of vector functions is “straightforward” by putting q SLFNs (with a single linear output node) together, and let each SLFN approximate a function which maps the input domain into one of components in the output domain. However one SLFN with q linear output

nodes can approximate vector functions in a more compact way. We will come back to this discussion in Chapter 3.

2.2 Constructive Methods

The universal approximation theorems introduced above are existence theorems in the sense that their proofs were carried out by an argument of *reductio ad absurdum*. Hence, constructive methods can not be derived from above theorems.

Recently, the proofs devised in the constructive way can be found in [41, 57, 39]. In constructive proofs, the function approximation is treated as a computable problem, which involves the estimation of the network architecture as well as the calculation of the network parameters. Hence, several constructive methods for SLFNs have been derived and will be discussed as follows.

In Meltser *et al.* [41], a constructive method with respect to the uniform norm has been devised for $\mathcal{N}_{\Psi,n}(\mathbf{x})$ when sigmoidal activation function is used. The method successively updates the parameters of the network and continuously adds new hidden units. The basic idea of the method was based on backpropagation (BP) in the form of descent on the cost function of the uniform norm. However, in the usage of this method to real problems, the issue of when to add new hidden units is not determined.

In stead of hidden unit growing, Scarselli & Tsoi [57] provided another method for implementing $\mathcal{N}_{\Psi,n}(\mathbf{x})$ to approximate polynomial functions of order r with respect to the L^p norm for $1 \leq p < \infty$. The hidden unit number n is fixed and determined by:

$$n = \binom{r + d - 1}{d - 1} \quad (2.6)$$

where d is the input dimension and r is the order of the polynomial function.

The hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ are chosen and fixed to satisfy the unique interpolation property of the polynomial function to be approximated.

The output unit parameters $\{\beta_i\}_{i=1}^n$ are determined to minimize the cost function:

$$\min_{\beta} = \|f(\mathbf{x}) - \mathcal{N}_{\Psi,n}(\mathbf{x})\|_p \quad (2.7)$$

where the vector $\beta = [\beta_1, \beta_2, \dots, \beta_n]^T$, encompassing all the output unit parameters.

The method implied that the hidden unit parameters can be chosen and fixed regardless of the exact polynomial function to be approximated. On the other hand, the selection of n and $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ is related to this special type of functions to be approximated. Hence the method is specific for the application of polynomial function approximation.

Methods analyzed above are applied into the construction of SLFNs with ridge activation function. Liao *et al.* [39] have come up with a constructive method for SLFNs with radial basis activation function, viz., RBF neural networks.

In Liao *et al.* [39], the constructed RBF neural networks can approximate any continuous function on a compact set $\mathbf{X} \subset \mathbb{R}^d$ with respect to the L^p norm for $1 \leq p < \infty$ provided that the radial basis activation function is continuous almost everywhere, locally essentially bounded, and nonpolynomial. The condition of integrability of the activation function was removed. Further, in their study, $b_i \in \mathbb{R}$ instead of $b_i \in \mathbb{R}^+$ in Theorem 2.7.

In the constructed method, the number of hidden units is continuously increased by one and one sample is learned per iteration. This results in the same number of hidden units as the number of training samples at the end of the learning process. The hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ are chosen by random selection (RS) or orthogonal least-squares (OLS) method. The output unit parameters $\{\beta_i\}_{i=1}^n$ are then estimated using the same method as in [57].

Numerical experiments have shown that the training performances achieved by the constructive method using RS and OLS are comparable. Thus it is also implied that the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ in the implementation of RBF neural networks can be chosen randomly regardless of the exact function to be approximated. On the other hand, the problems with this method are the intensive computational complexity when a large number of training samples are used, and the poor generalization performance when the training samples are corrupted by noise.

The shortcomings of above constructive algorithms can be summarized as follows:

- 1) They are activation function specific: each constructive method assumes a specific type of activation functions for the SLFN hidden layer.
- 2) They involve intense computational complexity for vector function approximation (excluding [41]): The extension for the approximation of vector functions $f : \mathbf{R}^d \rightarrow \mathbb{R}^q$ by putting q SLFNs (with single linear output unit) together, although straightforward, intensively increases the computational complexity.

2.3 Summary

In this chapter, the theories of SLFN function approximation using ridge function and radial basis function hidden units have been presented. Constructive design methods have been given and the issues inherent with these methods in real applications have been discussed.

The work of Scarselli [57] and Liao [39] imply that the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ in both ridge function and radial basis function SLFNs, can be fixed regardless of the exact function to be approximated. Furthermore, numerical experiments in Liao [39] show that the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ can be chosen randomly without increasing the training error. In other words, the non-linear computational units in the hidden layer can be very general for function

2.3 Summary

18

approximation.

Chapter 3

Extreme Learning Machine

Recently, a constructive algorithm referred to as **Extreme Learning Machine (ELM)** has been developed by Huang *et al.* [31, 29, 30, 25]. **ELM** constructs SLFNs both using ridge and radial basis activation in a unified batch training framework. Simulation results show that **ELM** achieves improved generalization (testing) performance with reduced computational complexity in benchmark problems [24] and in the application of mental task classification [38].

3.1 Unified SLFN with Multiple Linear Output Nodes

The issues inherent with the constructive methods in [41, 57, 39] have been discussed in Section 2.2. In the effort to solve these issues, a novel constructive algorithm **ELM** has recently been proposed by Huang *et al.* [31, 29, 30, 25].

First **ELM** can construct SLFNs with hidden unit activations of ridge function and radial basis function in a unified batch training framework. Second, inspired by the fact that the nonlinear computational units in the hidden layer are very general for function approximation, **ELM** can be extended to construct SLFNs with multiple linear output nodes to solve the problem of vector function approximation. This

strategy used by **ELM** reduces the computational complexity, compared with the straightforward way by putting q SLFNs (with a single linear output nodes) together. Third, **ELM** is a constructive method that allows SLFNs to approximate any type of continuous functions, instead of a specific type of functions as in [57].

Figure 3.1: A Single hidden Layer Feedforward Network (SLFN) with multiple linear output nodes

A SLFN with multiple linear output nodes is depicted in Figure 3.1. From the viewpoint of input-output mapping, such a SLFN represents a set of vector functions $f : \mathbb{R}^d \rightarrow \mathbb{R}^q$, whose components in the output domain can be defined by:

$$\mathcal{N}^j_{\mathbf{G},n}(\mathbf{x}) = \sum_{i=1}^n \beta_{ji} \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i) \quad (3.1)$$

where $j = 1, 2, \dots, q$ is the index of the output component. β_{ji} is the output weight connecting the i^{th} hidden unit to the j^{th} output node.

Let $\beta_i = [\beta_{1i}, \beta_{2i}, \dots, \beta_{qi}]^T \in \mathbb{R}$ denote the output weight vector connecting the i^{th} hidden unit to the output layer, and $\mathcal{N}_{G,n}(\mathbf{x}) = [\mathcal{N}_{G,n}^1(\mathbf{x}), \mathcal{N}_{G,n}^2(\mathbf{x}), \dots, \mathcal{N}_{G,n}^q(\mathbf{x})]^T$ denote the output vector. Then we may rewrite Equation (3.1) in the vector formula:

$$\mathcal{N}_{G,n}(\mathbf{x}) = \sum_{i=1}^n \beta_i \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i) \quad (3.2)$$

Note that $\mathcal{N}_{G,n}$ with $\beta_i \in \mathbb{R}$ in Equation (2.1) is one component of $\mathcal{N}_{G,n}$ with $\beta_i \in \mathbb{R}^q$ in Equation (3.2).

In real-world applications, the function is unknown a priori, but only a finite number of training input-output samples are available. Unlike the constructive algorithms in [41] (where the function to be approximated is assumed known a priori) and [57] (where the degree of the polynomial to be approximated is given), **ELM** does not assume any knowledge about the function to be approximated but constructs SLFNs based on a finite number of input-output samples. We assume that a finite set of N training input-output samples $\{(\mathbf{x}_k, \mathbf{t}_k) | \mathbf{x}_k \in \mathbb{R}^d, \mathbf{t}_k \in \mathbb{R}^q\}_{k=1}^N$ is available. The actual output vector of the SLFN can be defined as:

$$\mathcal{N}_{G,n}(\mathbf{x}_k) = \sum_{i=1}^n \beta_i \mathbf{G}(\mathbf{x}_k, \mathbf{a}_i, b_i) \quad (3.3)$$

for all $k = 1, 2, \dots, N$. Further we may rewrite Equation (3.3) in a matrix form expressed as:

$$\mathcal{N}_{G,n}(\mathbf{X}) = \mathbf{H}(\mathbf{X})\beta \quad (3.4)$$

where

$$\mathbf{H}(\mathbf{X}) = \begin{bmatrix} \mathbf{G}(\mathbf{x}_1, \mathbf{a}_1, b_1) & \mathbf{G}(\mathbf{x}_1, \mathbf{a}_2, b_2) & \cdots & \mathbf{G}(\mathbf{x}_1, \mathbf{a}_n, b_n) \\ \mathbf{G}(\mathbf{x}_2, \mathbf{a}_1, b_1) & \mathbf{G}(\mathbf{x}_2, \mathbf{a}_2, b_2) & \cdots & \mathbf{G}(\mathbf{x}_2, \mathbf{a}_n, b_n) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{G}(\mathbf{x}_N, \mathbf{a}_1, b_1) & \mathbf{G}(\mathbf{x}_N, \mathbf{a}_2, b_2) & \cdots & \mathbf{G}(\mathbf{x}_N, \mathbf{a}_n, b_n) \end{bmatrix}_{N \times n} \quad (3.5)$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}_{N \times d} \quad \mathcal{N}_{\mathbf{G},n}(\mathbf{X}) = \begin{bmatrix} \mathcal{N}_{\mathbf{G},n}(\mathbf{x}_1)^T \\ \mathcal{N}_{\mathbf{G},n}(\mathbf{x}_2)^T \\ \vdots \\ \mathcal{N}_{\mathbf{G},n}(\mathbf{x}_N)^T \end{bmatrix}_{N \times q} \quad \beta = \begin{bmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_n^T \end{bmatrix}_{n \times q} \quad (3.6)$$

are referred to as the hidden layer output matrix, input matrix, actual output matrix, and output weight matrix, respectively. We also define the desired output matrix as

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times q} . \quad (3.7)$$

The hidden layer output matrix plays an important role in the theory of **ELM**. Hence a detail discussion about it will be presented in the following section.

3.2 ELM theory

The theory of **ELM** consists of two principles, viz., exact mapping and approximate mapping [31, 29, 30, 25].

- 1) Exact Mapping: Given a set of N distinct training input-output samples such that $x_i \neq x_j \Rightarrow t_i \neq t_j$ and vice versa for $1 \leq i, j \leq N$. Let the number of hidden units be equal to the number of training samples, $n = N$. Then $\mathcal{N}_{G,N}$ realized by the SLFN with N hidden units can exactly reproduce the desired input-output responses (i.e., memorize the set of N training samples with zero error):

$$\|\mathcal{N}_{G,n}(\mathbf{X}) - \mathbf{T}\|^2 = 0 \quad (3.8)$$

provided that the activation function used in the hidden layer is infinitely differentiable. Further, the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ can be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to any continuous probability distribution. Then the output unit parameters $\{\beta_i\}_{i=1}^n$ are analytically determined by solving a linear system in one step. This result will be summarized in Theorem 3.1 as the exact mapping theorem of **ELM**.

- 2) Approximate Mapping: Given a set of N distinct training input-output samples such that $x_i \neq x_j \Rightarrow t_i \neq t_j$ and vice versa for $1 \leq i, j \leq N$. The number of hidden units is selected and in general is less than the number of training samples, $n < N$. Then $\mathcal{N}_{G,n}$ realized by the SLFN with n hidden units can approximate the input-output correspondences with a small positive error $\epsilon > 0$:

$$\|\mathcal{N}_{G,n}(\mathbf{X}) - \mathbf{T}\|^2 < \epsilon \quad (3.9)$$

provided that the activation function used in the hidden layer is infinitely differentiable. Again the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ can be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to any continuous probability distribution, and the output unit parameters $\{\beta_i\}_{i=1}^n$ are analytically determined by solving a

generalized linear system in one step. This result also will be formally stated in Theorem 3.2 as the approximate mapping theorem of **ELM**.

The principles of the theory of **ELM** imply the following facts. First the hidden unit parameters of a SLFN are randomly chosen according to some continuous probability distribution (such as continuous uniform distribution). Hence they are irrelative to the problem to be solved. Then the output unit parameters are estimated by solving a (generalized) linear system in one step. To sum up, the construction of a SLFN through **ELM** avoids the lengthy and complicated training procedure, such as backpropagation (BP), where all parameters of the network are *iteratively* updated using appropriately selected learning parameters or control parameters.

To proceed further, we first consider **ELM** in the case of exact mapping where $n = N$. Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^N$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution. We obtain that the hidden layer output matrix is nonsingular. It is proven in the following lemmas regarding the hidden unit activations of ridge and radial basis function are used, respectively.

When ridge activation function is used in the hidden layer of a SLFN, Lemma 3.1 proven by Tamura states that [60]:

Lemma 3.1. *Let $\{\mathbf{x}_k\}_{k=1}^N$ be a set of distinct points in $\mathbb{R}^d$. Given $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ are randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution and $n = N$. If the ridge activation function $\Psi(\mathbf{a}^T \mathbf{x} + b)$ is infinitely differentiable. Then with probability one, the square hidden layer output matrix $\mathbf{H}$ is nonsingular.*

Proof: Suppose that $\mathbf{H}$ is singular. We see that there exists a non-zero vector $\mathbf{c} = [c_1, c_2, \dots, c_N]^T$ such that:

$$\sum_{k=1}^N c_k \begin{bmatrix} \Psi(\mathbf{a}_1^T \mathbf{x}_k + b_1) \\ \Psi(\mathbf{a}_2^T \mathbf{x}_k + b_2) \\ \vdots \\ \Psi(\mathbf{a}_N^T \mathbf{x}_k + b_N) \end{bmatrix} = 0 \quad (3.10)$$

For the i^{th} equation in (3.10), we have:

$$\sum_{k=1}^N c_k \Psi(\mathbf{a}_i^T \mathbf{x}_k + b_i) = 0 \quad (3.11)$$

Without loss of generality, we assume $c_N \neq 0$ such that:

$$\Psi_{N,i}(\mathbf{a}_i^T \mathbf{x}_N + b_i) = - \sum_{k=1}^{N-1} \frac{c_k}{c_N} \Psi_{k,i}(\mathbf{a}_i^T \mathbf{x}_k + b_i) \quad (3.12)$$

We may revise Equation (3.12) as:

$$\Psi_{N,i}(b'_i) = \sum_{k=1}^{N-1} c'_k \Psi_{k,i}(b'_i + e_k) \quad (3.13)$$

where $b'_i = b_i + \mathbf{a}_i^T \mathbf{x}_N$, $c'_k = -\frac{c_k}{c_N}$, and $e_k = \mathbf{a}_i^T \mathbf{x}_k - \mathbf{a}_i^T \mathbf{x}_N$. $\mathbf{a}_i$ is randomly chosen from $\mathbb{R}^d$. Such that we can assume that $\mathbf{a}_i^T \mathbf{x}_k \neq \mathbf{a}_i^T \mathbf{x}_{k'}$. Therefore $e_k \neq e_{k'}$ and $e_k \neq 0$ since $\mathbf{x}_k \neq \mathbf{x}_{k'}$ for all $k \neq k'$. Further b'_i owns its arbitrariness to the factor of $\mathbf{a}_i$.

The above formula states that a ridge function can be represented by a weighted sum of $N - 1$ ridge functions shifted by the item of e_k . The derivative of the left and right hand side of Equation (3.13) at the random point $b'_i \in \mathbb{R}$ generates the

following series of linear equations:

$$\Psi_{N,i}^{(0)}(b'_i) = \sum_{k=1}^{N-1} c'_k \Psi_{k,i}^{(0)}(b'_i + e_k) \quad (3.14)$$

$$\Psi_{N,i}^{(1)}(b'_i) = \sum_{k=1}^{N-1} c'_k \Psi_{k,i}^{(1)}(b'_i + e_k) \quad (3.15)$$

$$\Psi_{N,i}^{(2)}(b'_i) = \sum_{k=1}^{N-1} c'_k \Psi_{k,i}^{(2)}(b'_i + e_k) \quad (3.16)$$

$\vdots$

where $\Psi_{k,i}^{(q)}$ is the q^{th} derivative of a ridge function used in the i^{th} hidden unit as the k^{th} input-output sample is presented. It is found that the series of linear equations is infinite in that $\Psi(\mathbf{a}^T \mathbf{x} + b)$ is infinitely differentiable. This indicates a contradiction that infinite equations can be satisfied by finite $N - 1$ parameters when b'_i is random. The proof is completed. $\square$

We may extend Tamura's arguments to the case when RBF is used as the activation function in the hidden layer.

Lemma 3.2. *Let $\{\mathbf{x}_k\}_{k=1}^N$ be a set of distinct points in $\mathbb{R}^d$. Given $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution and $n = N$. If the radial basis activation function $\Phi(\frac{\|\mathbf{x} - \mathbf{a}\|}{b})$ is infinitely differentiable. Then with probability one, the square hidden layer output matrix $\mathbf{H}$ is nonsingular.*

Proof: Similar with the case of ridge activation function, suppose that $\mathbf{H}$ is singular. We see that there exists a non-zero vector $\mathbf{c} = [c_1, c_2, \dots, c_N]^T$ such that:

$$\sum_{k=1}^N c_k \begin{bmatrix} \Phi(\frac{\|\mathbf{x}_k - \mathbf{a}_1\|}{b_1}) \\ \Phi(\frac{\|\mathbf{x}_k - \mathbf{a}_2\|}{b_2}) \\ \vdots \\ \Phi(\frac{\|\mathbf{x}_k - \mathbf{a}_N\|}{b_N}) \end{bmatrix} = 0 \quad (3.17)$$

where the i^{th} equation is given by:

$$\sum_{k=1}^N c_k \Phi\left(\frac{\|\mathbf{x}_k - \mathbf{a}_i\|}{b_i}\right) = 0 \quad (3.18)$$

Without loss of generality, we assume $c_N \neq 0$ such that:

$$\Phi_{N,i}\left(\frac{\|\mathbf{x}_N - \mathbf{a}_i\|}{b_i}\right) = - \sum_{k=1}^{N-1} \frac{c_k}{c_N} \Phi_{k,i}\left(\frac{\|\mathbf{x}_k - \mathbf{a}_i\|}{b_i}\right) \quad (3.19)$$

We may revise Equation (3.19) as:

$$\Phi_{N,i}(n_i) = \sum_{k=1}^{N-1} c'_k \Phi_{k,i}(n_i - e_k) \quad (3.20)$$

where $n_i = \frac{\|\mathbf{x}_N - \mathbf{a}_i\|}{b_i}$, $c'_k = -\frac{c_k}{c_N}$, and $e_k = \frac{\|\mathbf{x}_N - \mathbf{a}_i\|}{b_i} - \frac{\|\mathbf{x}_k - \mathbf{a}_i\|}{b_i}$. $\mathbf{a}_i$ and $\mathbf{b}_i \neq 0$ can be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$. Such that we can assume that $\frac{\|\mathbf{x}_N - \mathbf{a}_i\|}{b_i} \neq \frac{\|\mathbf{x}_k - \mathbf{a}_i\|}{b_i}$. Therefore $e_k \neq e_{k'}$ and $e_k \neq 0$ since $\mathbf{x}_k \neq \mathbf{x}_{k'}$ for all $k \neq k'$.

The above formula states that a RBF can be represented by a weighted sum of $N - 1$ RBFs shifted by the item of e_k . The derivative of the left and right hand side of Equation (3.20) at an random point $n_i \in \mathbb{R}$ generates the following series of linear equations:

$$\Phi_{N,i}^{(0)}(n_i) = \sum_{k=1}^{N-1} c'_k \Phi_{k,i}^{(0)}(n_i - e_k) \quad (3.21)$$

$$\Phi_{N,i}^{(1)}(n_i) = \sum_{k=1}^{N-1} c'_k \Phi_{k,i}^{(1)}(n_i - e_k) \quad (3.22)$$

$$\Phi_{N,i}^{(2)}(n_i) = \sum_{k=1}^{N-1} c'_k \Phi_{k,i}^{(2)}(n_i - e_k) \quad (3.23)$$

$\vdots$

where $\Phi_{k,i}^{(q)}$ is the q^{th} derivative of a RBF used in the i^{th} hidden unit as the k^{th} input-output sample is presented. We find that the series of linear equations is infinite in that $\Phi(\frac{\|\mathbf{x}-\mathbf{a}\|}{b})$ is infinitely differentiable. This indicates a contradiction that infinite equations can be satisfied by finite $N - 1$ parameters when n_i being randomly generated. The proof is completed. $\square$

Applying Lemma 3.1 and 3.2 (regarding H is nonsingular) to the exact mapping condition in Equation 3.8, we obtain a set of linear equations as $\mathbf{H}(\mathbf{X})\beta = \mathbf{T}$. And then the output unit parameters can be uniquely determined. This result can be formally stated in the theorem of **ELM** for exact mapping (i.e., memorization) [31, 29, 30, 25]:

Theorem 3.1. *Given N distinct input-output samples $\{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^q\}_{i=1}^N$. For $n = N$, $\mathcal{N}_{G,N}$ realized by the SLFN with N hidden units using an infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^N$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution. Then with probability one, $\mathcal{N}_{G,N}$ exactly reproduces N distinct input-output responses by linearly determining the output unit parameters $\{\beta_i\}_{i=1}^N$.*

Proof: Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution. Inserting the exact mapping condition of Equation (3.8), we obtain the following linear system of equations for the output unit parameters $\{\beta_i\}_{i=1}^n$ (see also [56, 5, 52, 53, 12]):

$$\mathbf{H}(\mathbf{X})\beta = \mathbf{T} \tag{3.24}$$

where $\beta = [\beta_1^T, \beta_2^T, \dots, \beta_n^T]^T$ encompasses all the output unit parameters. Since the input-output samples are distinct and activation function $\mathbf{G}(\mathbf{x}, \mathbf{a}, b)$ is infinitely differentiable, then with probability one, the square hidden layer output matrix $\mathbf{H}(\mathbf{X})$ is nonsingular and thus invertible according to Lemma 3.1 and Lemma 3.2.

Therefore, the linear system in (3.24) can be solved for the output unit parameters β as:

$$\beta = \mathbf{H}^{-1}(\mathbf{X})\mathbf{T} \quad (3.25)$$

The proof is complete. $\square$

Lemma 3.1 and Lemma 3.2 have proven that the square hidden layer output matrix is nonsingular and thus invertible. However in the cases where the input samples are close to each other, $\mathbf{H}$ approaches singularity, and the linear system $\mathbf{H}\beta = \mathbf{T}$ may be ill-conditioned. In order to solve this problem, a scheme will be introduced in the implementation of the algorithm in Section 3.4.

ELM for exact construction of the input-output mapping (i.e., memorization with zero error) may lead to prohibitively intensive computational complexity in that the number of hidden nonlinear computational units increases with the number of the training samples. Additionally in real applications, the input-output samples are contaminated by noise due to measurement artifacts, interferences, etc. Although the constructed SLFN can realize the input-output correspondences exactly, it may not have good generalization. This is known as the problem of over fitting and should be avoided. On the other hand, it is found that if the network is parsimonious, less likely it will cause over fitting occur.

For the sake of reducing computational complexity and improving generalization performance, **ELM** prefers the number of hidden units less than the number of training samples, and then constructs the mapping.

Hereinafter we consider the case of approximate mapping where the number of hidden units is selected by some model selection method and it is less than the number of samples, $n < N$. In such case, the hidden layer output matrix $\mathbf{H}$ is non-square and non-invertible. It is necessary to introduce the concept of the generalized inverse (or pseudoinverse) of an arbitrary matrix $\mathbf{H}$. It plays the role

of $\mathbf{H}^{-1}$ when $\mathbf{H}$ is singular.

The generalized inverse of an arbitrary matrix $\mathbf{H}$, defined by Moore(1920) and Penrose(1955) independently, is known as Moore-Penrose inverse. We introduce the definition of generalized inverse proposed by Penrose only [51]:

Definition 3.1. $\mathbf{G}$ is the generalized inverse of $\mathbf{H}$ if

$$\mathbf{HGH} = \mathbf{H} \quad (3.26)$$

$$(\mathbf{HG})^T = \mathbf{HG} \quad (3.27)$$

$$\mathbf{GHG} = \mathbf{G} \quad (3.28)$$

$$(\mathbf{GH})^T = \mathbf{GH} \quad (3.29)$$

For the sake of convenience, we denote the generalized inverse of $\mathbf{H}$ by $\mathbf{H}^\dagger$. Then applying the definition of generalized inverse into the context of a linear system, we may proceed with the following lemmas [9, 58]:

Lemma 3.3. A linear system of equations $\mathbf{H}\beta = \mathbf{T}$, where $\mathbf{H} \in \mathbb{R}^{N \times n}$, $\text{rank}(\mathbf{H}) = r \leq \min(N, n)$, is solvable if and only if $\mathbf{T} = \mathbf{HH}^\dagger \mathbf{T}$.

We prove the necessity and sufficiency of the condition for the solvability of $\mathbf{H}\beta = \mathbf{T}$ as follows:

Proof. From the property $\mathbf{HH}^\dagger \mathbf{H} = \mathbf{H}$ in Equation (3.26) in the definition of generalized inverse, we have $\mathbf{HH}^\dagger \mathbf{H} = \mathbf{H} \implies \mathbf{HH}^\dagger \mathbf{H}\beta = \mathbf{H}\beta \implies \mathbf{HH}^\dagger \mathbf{T} = \mathbf{T}$. It shows that the condition $\mathbf{T} = \mathbf{HH}^\dagger \mathbf{T}$ is necessary for solving $\mathbf{H}\beta = \mathbf{T}$. On the other hand, it is also sufficient since $\mathbf{HH}^\dagger \mathbf{T} = \mathbf{T} \implies \beta^* = \mathbf{H}^\dagger \mathbf{T}$, which is a solution to $\mathbf{H}\beta = \mathbf{T}$. $\square$

In the following lemma, we state the format of the solutions to the linear system in Lemma 3.3.

Lemma 3.4. *Given a linear system of equations $\mathbf{H}\beta = \mathbf{T}$, where $\mathbf{H} \in \mathbb{R}^{N \times n}$, $\text{rank}(\mathbf{H}) = r \leq \min(N, n)$, and $\mathbf{T} \in \mathbb{R}^{N \times q}$. If it is solvable, then its general solutions $\tilde{\beta} = \mathbf{H}^\dagger \mathbf{T} + (\mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H})\mathbf{Z}$ ($\mathbf{Z} \in \mathbb{R}^{n \times q}$) minimize $\|\mathbf{H}\beta - \mathbf{T}\|^2$. Furthermore, its special solution $\beta^* = \mathbf{H}^\dagger \mathbf{T}$ with minimum norm is unique.*

The proof in the following is composed of two parts: first we establish the general solutions that minimize $\|\mathbf{H}\beta - \mathbf{T}\|^2$. Then we show that its special solution with minimum norm is unique.

Proof: Given a linear system of equations $\mathbf{H}\beta = \mathbf{T}$, where $\mathbf{H} \in \mathbb{R}^{N \times n}$, $\text{rank}(\mathbf{H}) = r \leq \min(N, n)$. If it is solvable, first we show that its general solutions $\tilde{\beta} = \mathbf{H}^\dagger \mathbf{T} + (\mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H})\mathbf{Z}$ ($\mathbf{Z} \in \mathbb{R}^{n \times q}$) minimizes $\|\mathbf{H}\beta - \mathbf{T}\|^2$. Making use of the Pythagorean theorem in Euclidean geometry, we observe that for any $\beta \in \mathbb{R}^{n \times q}$,

$$\begin{aligned} \|\mathbf{H}\beta - \mathbf{T}\|^2 &= \|\mathbf{H}(\beta - \tilde{\beta}) + \mathbf{H}\tilde{\beta} - \mathbf{T}\|^2 \\ &= \|\mathbf{H}(\beta - \tilde{\beta})\|^2 + \|\mathbf{H}\tilde{\beta} - \mathbf{T}\|^2 + 2[\mathbf{H}(\beta - \tilde{\beta})]^T(\mathbf{H}\tilde{\beta} - \mathbf{T}) \\ &= \|\mathbf{H}(\beta - \tilde{\beta})\|^2 + \|\mathbf{H}\tilde{\beta} - \mathbf{T}\|^2 + 2(\beta - \tilde{\beta})^T(\mathbf{H}^T \mathbf{H}\tilde{\beta} - \mathbf{H}^T \mathbf{T}) \end{aligned} \quad (3.30)$$

Substituting the general solution $\tilde{\beta} = \mathbf{H}^\dagger \mathbf{T} + (\mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H})\mathbf{Z}$, ($\mathbf{Z} \in \mathbb{R}^{n \times q}$) into the last term $\mathbf{H}^T \mathbf{H}\tilde{\beta} - \mathbf{H}^T \mathbf{T}$ in Equation 3.30, we have

$$\begin{aligned} \mathbf{H}^T \mathbf{H}\tilde{\beta} - \mathbf{H}^T \mathbf{T} &= \mathbf{H}^T \mathbf{H}(\mathbf{H}^\dagger \mathbf{T} + (\mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H})\mathbf{Z}) - \mathbf{H}^T \mathbf{T} \\ &= \mathbf{H}^T \mathbf{H}\mathbf{H}^\dagger \mathbf{T} + (\mathbf{H}^T \mathbf{H} - \mathbf{H}^T \mathbf{H}\mathbf{H}^\dagger \mathbf{H})\mathbf{Z} - \mathbf{H}^T \mathbf{T} \\ &= \mathbf{H}^T \mathbf{T} + (\mathbf{H}^T \mathbf{H} - \mathbf{H}^T \mathbf{H})\mathbf{Z} - \mathbf{H}^T \mathbf{T} \\ &= 0 \end{aligned} \quad (3.31)$$

Then Equation (3.30) becomes

$$\|\mathbf{H}\beta - \mathbf{T}\|^2 = \|\mathbf{H}(\beta - \tilde{\beta})\|^2 + \|\mathbf{H}\tilde{\beta} - \mathbf{T}\|^2 \quad (3.32)$$

Since $\|\mathbf{H}(\beta - \tilde{\beta})\|^2 \geq 0$ for any $\beta \in R^{n \times q}$, we have

$$\|\mathbf{H}\beta - \mathbf{T}\|^2 \geq \|\mathbf{H}\tilde{\beta} - \mathbf{T}\|^2 \quad (3.33)$$

Thus $\tilde{\beta}$ minimizes $\|\mathbf{H}\beta - \mathbf{T}\|^2$.

Next we show that the special solution $\beta^* \in \tilde{\beta}$ is the unique solution to $\mathbf{H}\beta = \mathbf{T}$ with minimum norm. Again by the Pythagorean theorem in Euclidean geometry, we have for any $\tilde{\beta} \neq \beta^*$

$$\begin{aligned} \|\tilde{\beta}\|^2 &= \|(\tilde{\beta} - \beta^*) + \beta^*\|^2 \\ &= \|(\tilde{\beta} - \beta^*)\|^2 + \|\beta^*\|^2 + 2\beta^{*T}(\tilde{\beta} - \beta^*) \end{aligned} \quad (3.34)$$

Observe that

$$\begin{aligned}
\beta^{*T}(\tilde{\beta} - \beta^*) &= (\mathbf{H}^\dagger \mathbf{T})^T (\mathbf{H}^\dagger \mathbf{T} + (\mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H}) \mathbf{Z} - \mathbf{H}^\dagger \mathbf{T}) \\
&= (\mathbf{H}^\dagger \mathbf{T})^T (\mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H}) \mathbf{Z} \\
&= ((\mathbf{H}^\dagger \mathbf{T})^T - (\mathbf{H}^\dagger \mathbf{T})^T \mathbf{H}^\dagger \mathbf{H}) \mathbf{Z} \\
&= ((\mathbf{H}^\dagger \mathbf{T})^T - (\mathbf{H}^\dagger \mathbf{T})^T (\mathbf{H}^\dagger \mathbf{H})^T) \mathbf{Z} \\
&= ((\mathbf{H}^\dagger \mathbf{T})^T - (\mathbf{H}^\dagger \mathbf{H} \mathbf{H}^\dagger \mathbf{T})^T) \mathbf{Z} \\
&= ((\mathbf{H}^\dagger \mathbf{T})^T - (\mathbf{H}^\dagger \mathbf{T})^T) \mathbf{Z} \\
&= 0
\end{aligned} \tag{3.35}$$

Thus Equation (3.34) becomes

$$\|\tilde{\beta}\|^2 = \|(\tilde{\beta} - \beta^*)\|^2 + \|\beta^*\|^2 \tag{3.36}$$

Since $\tilde{\beta} \neq \beta^*$, we obtain $\|(\tilde{\beta} - \beta^*)\|^2 > 0$, and hence

$$\|\tilde{\beta}\|^2 > \|\beta^*\|^2 \tag{3.37}$$

which shows that the special solution β^* is the unique solution with minimum norm. The proof is complete. $\square$

Based Lemma 3.4 (regarding β^* is unique) combined with the approximate mapping condition, we may formally state the theorem of **ELM** for approximate mapping as follows [31, 29, 30, 25]:

Theorem 3.2. *Given N distinct input-output samples $\{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^q\}_{i=1}^N$ and $\epsilon > 0$. For $n < N$, $\mathcal{N}_{G,n}$ realized by the SLFN with n hidden units*

using an infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^N$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution. Then the output unit parameters $\{\beta_i\}_{i=1}^n$ can be uniquely determined by $\beta^* = \mathbf{H}^\dagger \mathbf{T}$ such that $\mathcal{N}_{G,n}(\mathbf{x})$ approximates the input-output correspondences with $\sum_{k=1}^N \|\mathcal{N}_{G,n}(\mathbf{x}_k) - \mathbf{t}_k\|^2 < \epsilon$.

Proof: Huang *et al.* [25] has proven that for any SLFN with random hidden nodes:

$$\lim_{n \rightarrow N} \sum_{k=1}^N \|\mathcal{N}_{G,n}(\mathbf{x}_k) - \mathbf{t}_k\|^2 = 0 \quad (3.38)$$

Therefore theoretically there exists $n < N$ such that the approximate mapping condition is satisfied: $\sum_{k=1}^N \|\mathcal{N}_{G,n}(\mathbf{x}_k) - \mathbf{t}_k\|^2 < \epsilon$. This condition can be written in a compact formula as $\|\mathcal{N}_{G,n} - \mathbf{T}\|^2 < \epsilon$. The SLFN is implemented by randomly choosing the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution. Then the output unit parameters is uniquely determined by solving the generalized linear system of equations in Lemma 3.4 as

$$\beta^* = \mathbf{H}(\mathbf{X})^\dagger \mathbf{T} \quad (3.39)$$

This completes the proof. $\square$

3.3 Comparison of ELM with Regularization Theory and SVMs

So far the theorems for **ELM** in the case of exact mapping and approximate mapping have been introduced. In Theorem 3.1, the universal approximation capability of **ELM** is proven. It shows that with probability one, **ELM** can provide

exact mapping by using as many hidden nodes as the number of training samples. In this case, the exact mapping theorem of **ELM** using RBF activation is similar to the regularization theory [18]. In the regularization theory, prior information about the function to be approximated is embedded through the introduction of an additional regularizing term and the regularization parameter λ . A special network structure named the regularization network is derived for the implementation of the theory. The regularization network is composed of three layers, viz., input, hidden, and output layer. Specially in the hidden layer, the Green's functions (RBF function is one particular example.) are used as the activation. The number of hidden nodes is also equal to the number of training samples. In stead of randomly choosing the centers as done by **ELM**, the regularization theory set them equal to the inputs. Compared the exact mapping theorem of **ELM** with the regularization theory in the case where RBF activation function is used, **ELM** is coincident with the regularization theory when λ is set equal to zero and the hidden parameters in **ELM** equal to the inputs.

The approximate mapping using **ELM** is addressed in Theorem 3.2 since in practical applications, the generalization performance of an algorithm is concerned. In the theorem, ϵ is the function of n , which is determined through some model selection procedure as introduced in Session 4.4.1. Given a certain structure of SLFNs with n hidden nodes, the theorem guarantees that the optimal generalization performance can be obtained by minimizing the training error and the norm of the solution β simultaneously for a certain problem. The study by Bartlett [4] has been theoretically shown that the magnitude of the output weights (viz., the output unit parameters) affects the generalization performance of an algorithm. It suggests the use of an algorithm that minimizes the training error while keeping the norm of the output unit parameters as small as possible in order to achieve a proper generalization performance. From this point of view, the performance obtained by using **ELM** for approximate mapping is optimal.

Further, we compare **ELM** with Support Vector Machines (SVMs) [61, 62] as fol-

lows. SVMs are classified as one kind of universal feedforward networks. The scheme used by the support vector learning algorithm is an approximate implementation of the method of structural risk minimization, which simultaneously minimizes the training error and the Euclidean norm of its weight vector (parameters) subject to some constraints. When compare ELM with SVMs, it is found that the use of **ELM** and SVM requires some model parameter to be chosen a priori, viz., the number of hidden nodes (**ELM**) and the parameter C (SVM), respectively. While the computational complexity is concerned, SVMs require to solve a quadratic optimization problem; **ELM** only evokes a set of linear equations. Numerical simulations for the comparison of **ELM** and SVMs can be found on benchmark problems from the areas of regression and classification [24] and in the application of mental task classification [38].

3.4 Algorithm Implementation

The theorems of **ELM** have been reviewed. In the following we will summarize the algorithm of **ELM** including some issues related to its practical implementation.

For exact mapping, we have mentioned that the square matrix $\mathbf{H}$ may approach singularity in the cases where input-output samples are densely spaced. In order to solve this problem, a strategy proposed by Ferrari in [14] is introduced. The sigmoidal activation function is assumed to be used in the hidden layer and is defined by $\sigma(n) = \frac{e^{-n}-1}{e^{-n}+1}$. Then the input weight vectors are selected according to:

$$\mathbf{a}_i = c\mathbf{r}_i \quad (3.40)$$

where $\mathbf{r}_i$ is randomly chosen according to any continuous probability distribution (e.g., a continuous uniform distribution with zero mean and variance one). c is a scalar that can be adjusted based on how densely the input-output samples are

spaced. The biases are then computed by setting:

$$b_i = -\mathbf{a}_i^T \mathbf{x}_k, \forall i = k \quad (3.41)$$

where $1 \leq i, k \leq N$ since in the case of exact mapping we use as many hidden nodes as the number of training samples, $n = N$. Using Equation (3.41), the hidden sigmoidal computational units are distributed across the input space such that the diagonal of the resulted $\mathbf{H}$ is composed of zero elements. Further by adjusting the scalar c in Equation (3.40), we have the off-diagonal elements of $\mathbf{H}$ being highly saturated. Hence the produced $\mathbf{H}$ is well-conditioned. A well-conditioned $\mathbf{H}$ contributes to the exact mapping with the property of smoothness. The arguments can be simply extended to the case of RBF activations.

For approximate mapping, the estimation of output unit parameters involves the calculation of the generalized inverse of $\mathbf{H}$. Several practical methods have been devised for the estimation of $\mathbf{H}^\dagger$. They include orthogonalization, iterative method, and Singular Value Decomposition (SVD) [51]. The orthogonalization and iterative method have limitations since searching and iteration are used. SVD is widely used for calculating the generalized inverse of $\mathbf{H}$, and will be applied in the implementation of **ELM**.

In the end of this section, we summarize the implementation of **ELM** as follows:

Batch ELM Algorithm: Given a SLFN with n hidden units using an infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. For N distinct input-output samples $\{(\mathbf{x}_k, \mathbf{t}_k) | \mathbf{x}_k \in \mathbb{R}^d, \mathbf{t}_k \in \mathbb{R}^q\}_{k=1}^N$:

step 1 Assign random values for the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ according to some continuous probability distribution.

step 2 Calculate the hidden layer output matrix $\mathbf{H}$.

step 3 Estimate the output unit parameters $\beta^* = \mathbf{H}^\dagger \mathbf{T}$.

Remark 1: For exact mapping, $n = N$; Otherwise we estimate n through some model selection procedure. A model selection procedure will be introduced in Session 4.4.1.

For details of the performance of **ELM** in real world applications, we refer interested readers to [24, 38].

3.5 Summary

In this chapter, the constructive algorithm **ELM** proposed by Huang *et al.* is introduced. On the basis of a finite set of input-output samples, **ELM** can construct SLFNs with both ridge and radial basis activation function in a unified batch training framework.

Theoretically, provided that the number of hidden units is equal to the number of training samples, **ELM** can reproduce the training samples perfectly. However, an approximate mapping may be preferred to the exact one from the viewpoint of computational complexity and generalization performance in real world applications. Thus, **ELM** extends to the case of approximate mapping where the number of hidden units is selected a prior and in general less than the number of training samples. As analyzed, **ELM** achieves improved generalization performance since the algorithm minimizes magnitude of the output weights while keeps the training square error small.

In the end of this chapter, the issues related to the practical implementation of **ELM** have been discussed. For exact mapping, the scheme for choosing hidden unit parameters is introduced to prevent the hidden layer output matrix approaching singularity. For approximate mapping, SVD technique is used in the estimation of generalized inverse of the hidden layer output matrix.

Chapter 4

Online Sequential ELM

4.1 Motivation

The constructive methods reviewed in the previous chapters include those for SLFNs with specific activation functions as well as **ELM** which deals with SLFNs in an activation function unified framework. And all these algorithms operate in a batch learning mode. For batch learning mode, the parameters of the network are estimated based on the entire training samples. Therefore they are preferred for off-line applications where the whole set of training data is available a priori.

For on-line applications, the batch constructive algorithms should retrain the SLFN whenever a new sample is presented. In other words, previous computational effort goes to waste. On the other hand, sequential algorithms iteratively update the parameters of the network based on current data while preserving previous computation effort. From this viewpoint, sequential algorithms are preferred to batch algorithms for on-line applications. A brief review of several popular sequential learning algorithms is presented here.

Stochastic BP based Sequential Algorithms: Back-Propagation (BP) [55] algorithm and its variants have been the backbone learning algorithms for layered feedforward neural networks with differentiable ridge activation function. In the

learning process of BP, the parameters of the network are iteratively updated by minimizing a cost function. Such learning process can be classified into batch and sequential mode depending upon the definition of the cost function. In batch mode the cost function is defined over the entire training data, whereas in sequential mode the cost function is estimated using the current training data and referred to as the instantaneous cost function.

The technique of BP based on the instantaneous cost function is referred to as stochastic BP based sequential algorithms. The early work of stochastic BP based sequential algorithms is focused on the use of first order information of the instantaneous cost function for updating the parameters of the network, for example, Stochastic Gradient descent Back-Propagation algorithms (SGBPs) [55]. However these algorithms suffer from slow convergence [18]. To overcome this deficiency, some researchers exploited second order information of the instantaneous cost function to accelerate convergence. Recursive Levenberg-Marquardt (LM) algorithms for approximating second order information have been intensively investigated [44, 3]. However, the calculation or approximation of second order information increases the computational complexity in each update iteration.

The model parameters of SGBPs and recursive LMs include the number of hidden units, learning rate and/or momentum constant. These parameters should be carefully selected for successful online learning.

RAN based Sequential Algorithms: Resource Allocation Network (RAN) based algorithms are popular for training SLFNs with radial basis function, viz., RBF neural networks. These algorithms include RAN [50] and its extensions, e.g., RANEKF [34], MRAN [66, 67], GAP-RBF [27], and GGAP-RBF [28]. For all these algorithms, the parameters of the network are iteratively updated on the basis of a single training sample. Although GAP-RBF and GGAP-RBF learn faster than others by using the winner neuron idea, simulation results showed their learning speed was still slow in large-scale online applications. Hence their online capability is limited.

The number of RBF hidden units in these algorithms is iteratively updated. RAN and RANEKF determine whether to add a unit based on the novelty of incoming data. Besides growing units, MRAN, GAP-RBF, and GGAP-RBF prune insignificant units from the current network. However, these algorithms involve many control parameters that need to be determined off-line. Additionally, GAP-RBF and GGAP-RBF require the input distribution information a priori.

From the above review of contemporary sequential learning algorithms, we see the following. First, they are all activation function specific, e.g., stochastic BP based sequential algorithms construct the networks with differentiable ridge activation functions, whereas RAN based sequential algorithms implement RBF neural networks. Second, the computational complexity of these algorithms is intensive for large-scale online applications. Third, the use of these algorithms involves many learning parameters and control parameters, which should be carefully selected in order to achieve proper generalization performance. This makes the learning procedure much more complicated. Fourth, the learning mode of these algorithms (one-by-one, viz., one sample is learned at one time) may not adapt to some on-line applications, where a chunk of training samples is available at a time. All these facts motivate us to develop an efficient and activation function unified sequential learning algorithm, which can construct SLFNs both in one-by-one and chunk-by-chunk (viz., a chunk of data is learned at one time) learning mode.

4.2 Derivation of Online Sequential ELM

Online Sequential **ELM** (**OS-ELM**) [26, 37] stems from batch **ELM**. Among all the batch constructive algorithms reviewed in Chapter 2 and Chapter 3, batch **ELM** efficiently constructs SLFNs in an activation function unified framework and achieves improved generalization performance with reduced computational complexity. For approximate mapping, the use of batch **ELM** only involves the selection of one model parameter, i.e, the number of hidden units. Hence batch

ELM is the choice, from which an efficient sequential algorithm may be derived.

OS-ELM iteratively updates the parameters of a SLFN based on current input-output sample(s). As analyzed, **OS-ELM** can operate both in one-by-one and chunk-by-chunk learning mode. We generalize the sequential learning mode of one sample per iteration (viz., one-by-one) to a chunk of samples per iteration (viz., chunk-by-chunk). Furthermore the chunk size need not be fixed and may vary in each iteration. The flexibility of learning mode is illustrated via numerical experiments.

4.2.1 Online Sequential ELM Theory

OS-ELM encompasses the following principles [26, 37]:

- 1) The generalized inverse of the hidden layer output matrix in batch **ELM** is proven to be estimated by the left pseudoinverse of the matrix $\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$, such that the unique solution for the output unit parameters is defined by $\beta^* = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T}$, which is referred to as the least-squares solution to the linear system of equations $\mathbf{H}\beta = \mathbf{T}$.
- 2) The derivation of the one-by-one formula of recursive least-squares (RLS) algorithm results in **OS-ELM** in the learning mode of one-by-one. This kind of learning mode is suitable for the online applications where the training samples arrive one at a time.
- 3) The derivation of the chunk-by-chunk formula of RLS algorithm results in **OS-ELM** in the learning mode of chunk-by-chunk. Additionally, the chunk size in the algorithm need not be fixed and may vary in each update iteration. This kind of learning mode is suitable for the online applications where a chunk of training samples arrives at a time.

We proceed further with the following lemma. It shows that under the condition $\text{rank}(\mathbf{H}) = n$ if $\mathbf{H} \subset \mathbb{R}^{N \times n}$, the generalized inverse of H has the form of $\mathbf{H}^\dagger =$

$(\mathbf{H}^T \mathbf{H})^{(-1)} \mathbf{H}^T$. Further under this rank condition, $\mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H} = 0$ and hence $\tilde{\beta} = \beta^*$ (see Lemma 3.4). This result is stated as follows:

Lemma 4.1. *Given a linear system of equations $\mathbf{H}\beta = \mathbf{T}$, where $\mathbf{H} \in \mathbb{R}^{N \times n}$ and $\mathbf{T} \in \mathbb{R}^{N \times q}$. If $\text{rank}(\mathbf{H}) = n$, then $\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$. Further its general solutions are simplified and identical with its special solution $\tilde{\beta} = \beta^*$.*

Proof: If $\mathbf{H} \in \mathbb{R}^{N \times n}$ and $\text{rank}(\mathbf{H}) = n$, then the following equalities hold:

$$\begin{aligned}
 \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{H} &= \mathbf{H} \\
 (\mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T)^T &= \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \\
 (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{H}(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T &= (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \\
 ((\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{H})^T &= (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{H}
 \end{aligned} \tag{4.1}$$

Considering Definition 3.1, the generalized inverse of $\mathbf{H}$, under this condition, has the following form:

$$\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \tag{4.2}$$

Further Lemma 3.4 provides the formula of the generalized solutions to this linear system as $\tilde{\beta} = \mathbf{H}^\dagger \mathbf{T} + (\mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H})\mathbf{Z}$ with $\mathbf{Z} \in \mathbb{R}^{n \times q}$. Using Equation (4.2), the second term of $\tilde{\beta}$ becomes

$$\begin{aligned}
 \mathbf{I}_n - \mathbf{H}^\dagger \mathbf{H} &= \mathbf{I}_n - (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{H} \\
 &= \mathbf{I}_n - \mathbf{I}_n \\
 &= 0
 \end{aligned} \tag{4.3}$$

Hence the formula of the general solutions is simplified as $\tilde{\beta} = \mathbf{H}^\dagger \mathbf{T}$ identical with the special solution as given in Lemma 3.4. $\square$

For a linear system $\mathbf{H}\beta = \mathbf{T}$, where $\mathbf{H} \in \mathbb{R}^{N \times n}$ and $\text{rank}(\mathbf{H}) = n$, it has been proven $\mathbf{H}^\dagger \mathbf{H} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{H} = \mathbf{I}_n$ (refer to Equation (4.3)). From this sense, $(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$ is called the left pseudoinverse of $\mathbf{H}$. Correspondingly, the solution in (4.5) is called the least-squares solution of the linear system.

Lemma 4.1 shows, if $\mathbf{H} \in \mathbb{R}^{N \times n}$ satisfies the condition $\text{rank}(\mathbf{H}) = n$, then the generalized inverse of the matrix is estimated as the left pseudoinverse of the matrix. The following theorem proves that such a condition is satisfied by the hidden layer output matrix of batch **ELM**. Then combining with Theorem 3.2, the formula of the solution of the output unit parameters is presented such that the approximate mapping condition is satisfied, $\sum_{k=1}^N \|\mathcal{N}_{G,n}(\mathbf{x}_k) - \mathbf{t}_k\|^2 < \epsilon$.

Theorem 4.1. *Given N distinct input-output samples $\{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^q\}_{i=1}^N$ and $\epsilon > 0$. For $n < N$, $\mathcal{N}_{G,n}(\mathbf{x})$ realized by a SLFN with n hidden units using infinitely differentiable activation function $G: \mathbb{R}^d \rightarrow \mathbb{R}$. Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution. Then the output unit parameters are uniquely determined by $\beta^* = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T}$ such that $\mathcal{N}_{G,n}(\mathbf{x})$ approximates the input-output correspondences with $\sum_{k=1}^N \|\mathcal{N}_{G,n}(\mathbf{x}_k) - \mathbf{t}_k\|^2 < \epsilon$.*

Proof: Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution, then the hidden layer output matrix $\mathbf{H} \in \mathbb{R}^{N \times n}$ ($N > n$) is determined.

Let the first n row vectors of $\mathbf{H}$ be denoted by $\mathbf{H}_0$:

$$\mathbf{H}_0 = \begin{bmatrix} \mathbf{G}(\mathbf{x}_1, \mathbf{a}_1, b_1) & \mathbf{G}(\mathbf{x}_1, \mathbf{a}_2, b_2) & \cdots & \mathbf{G}(\mathbf{x}_1, \mathbf{a}_n, b_n) \\ \mathbf{G}(\mathbf{x}_2, \mathbf{a}_1, b_1) & \mathbf{G}(\mathbf{x}_2, \mathbf{a}_2, b_2) & \cdots & \mathbf{G}(\mathbf{x}_2, \mathbf{a}_n, b_n) \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{G}(\mathbf{x}_n, \mathbf{a}_1, b_1) & \mathbf{G}(\mathbf{x}_n, \mathbf{a}_2, b_2) & \cdots & \mathbf{G}(\mathbf{x}_n, \mathbf{a}_n, b_n) \end{bmatrix}_{n \times n} \quad (4.4)$$

According to Lemma 3.1 and Lemma 3.2, $\mathbf{H}_0$ is square and nonsingular. Therefore $\text{rank}(\mathbf{H}_0) = n$ and also $\text{rank}(\mathbf{H}) = n$.

Applying Lemma 4.1, the output unit parameters are then uniquely determined by

$$\beta^* = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{T} \quad (4.5)$$

Invoking Theorem 3.2, the approximation mapping condition is satisfied also: $\sum_{k=1}^N \|\mathcal{N}_{G,n}(\mathbf{x}_k) - \mathbf{t}_k\|^2 < \epsilon$. $\square$

Theorem 4.1 is the foundation for **OS-ELM**. The formula of the solution to output unit parameters in Theorem 4.1 makes the application of recursive least-squares (RLS) algorithm for iteratively updating the parameters possible. We present the derivation of **OS-ELM** as follows.

OS-ELM is derived on the base of the recursive update version of the output unit parameters shown in Equation (4.5). From Theorem 4.1, it has been shown that the output unit parameters are uniquely defined as the least-squares solution to a linear system of equations. The recursive formula of the solution is referred to as recursive least-squares (RLS). Depending on the update formula of RLS, the proposed **OS-ELM** can operate in the learning mode of one-by-one and chunk-by-chunk. First we consider **OS-ELM** in one-by-one learning mode.

OS-ELM in One-by-One Learning Mode: Given a series of N distinct input-output samples $\{(\mathbf{x}_k, \mathbf{t}_k)\}_{k=1}^N$ and assume one sample is presented at a time. We

may denote the k^{th} hidden layer output row vector $[G(\mathbf{x}_k, \mathbf{a}_1, b_1), G(\mathbf{x}_k, \mathbf{a}_2, b_2), \dots, G(\mathbf{x}_k, \mathbf{a}_n, b_n)]$ by $\mathbf{h}_k$ with respect to the k^{th} input sample. For convenience we further define a series of matrices $\{(\mathbf{X}_j, \mathbf{H}_j, \mathbf{T}_j)\}_{j=0}^m$ as follows:

For $j = 0$:

$$\mathbf{X}_0 = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_{n_0}^T \end{bmatrix}_{n_0 \times d} \quad \mathbf{H}_0 = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_{n_0} \end{bmatrix}_{n_0 \times n} \quad \mathbf{T}_0 = \begin{bmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \\ \vdots \\ \mathbf{t}_{n_0}^T \end{bmatrix}_{n_0 \times q} \quad (4.6)$$

For $j \neq 0$:

$$\mathbf{X}_j = \begin{bmatrix} \mathbf{X}_{j-1} \\ \mathbf{x}_{n_0+j}^T \end{bmatrix} \quad \mathbf{H}_j = \begin{bmatrix} \mathbf{H}_{j-1} \\ \mathbf{h}_{n_0+j} \end{bmatrix} \quad \mathbf{T}_j = \begin{bmatrix} \mathbf{T}_{j-1} \\ \mathbf{t}_{n_0+j}^T \end{bmatrix} \quad (4.7)$$

where $n_0 \geq n$ and $N = n_0 + m$. When $j = m$, $\mathbf{X} = \mathbf{X}_m$, $\mathbf{H} = \mathbf{H}_m$, and $\mathbf{T} = \mathbf{T}_m$.

Given the model of a SLFN with n hidden units with an infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. It represents a set of input-output mapping functions defined by:

$$\mathcal{N}_{G,n}(x) = \sum_{i=1}^n \beta_i \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i) \quad (4.8)$$

Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution, and then fixed throughout the learning procedure. Only the output unit parameters $\{\beta_i\}_{i=1}^n$ are iteratively updated according to the one-by-one formula of RLS algorithm [9]:

For $j = 0$, an initial set of n_0 distinct input-output samples is accumulated. The initial output unit parameters are estimated by solving the linear system of equa-

tions:

$$\mathbf{H}_0\beta = \mathbf{T}_0 \quad (4.9)$$

where $\mathbf{H}_0 \in \mathbb{R}^{n_0 \times n}$, $n_0 \geq n$, and $\text{rank}(\mathbf{H}_0) = n$.

According to Theorem 4.1, the solution for the initial output unit parameters are uniquely determined by:

$$\beta^{(0)} = \mathbf{K}_0^{-1}\mathbf{H}_0\mathbf{T}_0 \quad (4.10)$$

where $\mathbf{K}_0 = \mathbf{H}_0^T\mathbf{H}_0$.

Suppose the $(n_0 + 1)^{st}$ input-output sample is presented. Then the output unit parameters are estimated by solving the current linear system of equations:

$$\mathbf{H}_1\beta = \mathbf{T}_1 \quad (4.11)$$

Similarly we have the solution:

$$\begin{aligned} \beta^{(1)} &= \mathbf{K}_1^{-1}\mathbf{H}_1^T\mathbf{T}_1 \\ &= \mathbf{K}_1^{-1}(\mathbf{H}_0^T\mathbf{T}_0 + \mathbf{h}_{n_0+1}^T\mathbf{t}_{n_0+1}^T) \end{aligned} \quad (4.12)$$

where

$$\begin{aligned} \mathbf{K}_1 &= \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{h}_{n_0+1} \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{h}_{n_0+1} \end{bmatrix} \\ &= \mathbf{H}_0^T\mathbf{H}_0 + \mathbf{h}_{n_0+1}^T\mathbf{h}_{n_0+1} \\ &= \mathbf{K}_0 + \mathbf{h}_{n_0+1}^T\mathbf{h}_{n_0+1} \end{aligned} \quad (4.13)$$

For $\mathbf{K}_0 \mathbf{K}_0^{-1} = \mathbf{I}_{n_0}$, Equation (4.12) may be rewritten as:

$$\begin{aligned}\beta^{(1)} &= \mathbf{K}_1^{-1}(\mathbf{K}_0 \mathbf{K}_0^{-1} \mathbf{H}_0^T \mathbf{T}_0 + \mathbf{h}_{n_0+1}^T \mathbf{t}_{n_0+1}^T) \\ &= \mathbf{K}_1^{-1}(\mathbf{K}_0 \beta^{(0)} + \mathbf{h}_{n_0+1}^T \mathbf{t}_{n_0+1}^T)\end{aligned}\quad (4.14)$$

From Equation (4.13), we have $\mathbf{K}_0 = \mathbf{K}_1 - \mathbf{h}_{n_0+1}^T \mathbf{h}_{n_0+1}$ and substitute it into Equation (4.14) and obtain:

$$\begin{aligned}\beta^{(1)} &= \mathbf{K}_1^{-1}(\mathbf{K}_1 \beta^{(0)} - \mathbf{h}_{n_0+1}^T \mathbf{h}_{n_0+1} \beta^{(0)} + \mathbf{h}_{n_0+1}^T \mathbf{t}_{n_0+1}^T) \\ &= \beta^{(0)} + \mathbf{K}_1^{-1} \mathbf{h}_{n_0+1}^T (\mathbf{t}_{n_0+1}^T - \mathbf{h}_{n_0+1} \beta^{(0)})\end{aligned}\quad (4.15)$$

where $\mathbf{K}_1 = \mathbf{K}_0 + \mathbf{h}_{n_0+1}^T \mathbf{h}_{n_0+1}$.

Observe that the solution $\beta^{(1)}$ is modified by adding to $\beta^{(0)}$ a correction term $\mathbf{K}_1^{-1} \mathbf{h}_{n_0+1}^T (\mathbf{t}_{n_0+1}^T - \mathbf{h}_{n_0+1} \beta^{(0)})$. The above argument can be generalized in a straightforward way. For instance, when the $(n_0 + j)^{th}$ input-output sample is presented, the solution $\beta^{(j)}$ is updated by:

$$\beta^{(j)} = \beta^{(j-1)} + \mathbf{K}_j^{-1} \mathbf{h}_{n_0+j}^T (\mathbf{t}_{n_0+j}^T - \mathbf{h}_{n_0+j} \beta^{(j-1)}) \quad (4.16)$$

where $\mathbf{K}_j = \mathbf{K}_{j-1} + \mathbf{h}_{n_0+j}^T \mathbf{h}_{n_0+j}$.

Further the recursive algorithm for updating $\mathbf{K}_j^{-1}$ using the Sherman-Morrison formula [23](p.51) is given by:

$$\mathbf{K}_j^{-1} = \mathbf{K}_{j-1}^{-1} - \frac{\mathbf{K}_{j-1}^{-1} \mathbf{h}_{n_0+j}^T \mathbf{h}_{n_0+j} \mathbf{K}_{j-1}^{-1}}{1 + \mathbf{h}_{n_0+j} \mathbf{K}_{j-1}^{-1} \mathbf{h}_{n_0+j}^T} \quad (4.17)$$

Let $\mathbf{P}_j = \mathbf{K}_j^{-1}$, then the update equations (4.17) and (4.16) can be rewritten as:

$$\mathbf{P}_j = \mathbf{P}_{j-1} - \frac{\mathbf{P}_{j-1} \mathbf{h}_{n_0+j}^T \mathbf{h}_{n_0+j} \mathbf{P}_{j-1}}{1 + \mathbf{h}_{n_0+j} \mathbf{P}_{j-1} \mathbf{h}_{n_0+j}^T} \quad (4.18)$$

and

$$\beta^{(j)} = \beta^{(j-1)} + \mathbf{P}_j \mathbf{h}_{n_0+j}^T (\mathbf{t}_{n_0+j}^T - \mathbf{h}_{n_0+j} \beta^{(j-1)}), \quad (4.19)$$

These update equations comprise the one-by-one formulation of the RLS algorithm, which is applied for iteratively updating the output unit parameters in **OS-ELM**. More important, when $j = m$, the final linear system is given by:

$$\mathbf{H}_m \beta = \mathbf{T}_m \quad (4.20)$$

Consequently the output unit parameters estimated by **OS-ELM** are identical with those determined by batch **ELM**:

$$\beta^* = \beta^{(m)} \quad (4.21)$$

for $\mathbf{H} = \mathbf{H}_m$ and $\mathbf{T} = \mathbf{T}_m$.

To proceed further, we define a series of functions $\{\mathcal{N}_{G,n}^{(j)}(\mathbf{x})\}_{j=0}^m$ as follows:

Definition 4.1.

$$\mathcal{N}_{G,n}^{(j)}(\mathbf{x}) = \sum_{i=1}^n \beta_i^{(j)} \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i) \quad (4.22)$$

And the present square training error (summed over all the training samples learned so far) can be defined as:

Definition 4.2.

$$\epsilon^{(j)} = \sum_{k=1}^{n_0+j} \|\mathcal{N}_{G,n}^{(j)}(\mathbf{x}_k) - \mathbf{t}_k\|^2 \quad (4.23)$$

Now we can formally state the theorem of **OS-ELM** in the learning mode of one-by-one as follows:

Theorem 4.2. *Given a series of N distinct input-output samples $\{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbb{R}^d, \mathbf{t}_i \in \mathbb{R}^q\}_{i=1}^N$ and $\epsilon > 0$. Let $\mathcal{N}_{G,n} = \sum_{i=1}^n \beta_i \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i)$ be realized by a SLFN with n hidden units using infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. Let the first n_0 input-output samples be the initial data, where $N \geq n_0 \geq n$. If the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ are randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution, then the output unit parameters $\{\beta_i\}_{i=1}^n$ are iteratively updated according to the one-by-one formula of RLS such that the present square training error approaches the approximate mapping condition $\lim_{j \rightarrow m} \epsilon^{(j)} = \epsilon$.*

Proof: Given the first n_0 distinct input-output samples. Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution, and $N \geq n_0 \geq n$. Then the initial linear system of equations for the output unit parameters is given by $\mathbf{H}_0 \beta = \mathbf{T}_0$. By solving the initial linear system, the initial output unit parameters are uniquely determined by:

$$\beta^{(0)} = (\mathbf{H}_0^T \mathbf{H}_0)^{-1} \mathbf{H}_0^T \mathbf{T}_0 \quad (4.24)$$

Without loss of generality, we may assume the output of the SLFN is scalar (i.e., $q = 1$). Then the initial square training error is given by:

$$\begin{aligned} \epsilon^{(0)} &= [\mathbf{H}_0 \beta^{(0)} - \mathbf{T}_0]^T [\mathbf{H}_0 \beta^{(0)} - \mathbf{T}_0] \\ &= (\mathbf{H}_0 \beta^{(0)})^T (\mathbf{H}_0 \beta^{(0)}) - 2(\mathbf{H}_0 \beta^{(0)})^T \mathbf{T}_0 + \mathbf{T}_0^T \mathbf{T}_0 \end{aligned} \quad (4.25)$$

Suppose the $(n_0+1)^{st}$ input-output sample is presented, the output unit parameters are updated according to:

$$\beta^{(1)} = \beta^{(0)} + \Delta\beta^{(1)} \quad (4.26)$$

where $\Delta\beta^{(1)} = (\mathbf{H}_0^T \mathbf{H}_0 + \mathbf{h}_{n_0+1}^T \mathbf{h}_{n_0+1})^{-1} \mathbf{h}_{n_0+1}^T (\mathbf{t}_{n_0+1}^T - \mathbf{h}_{n_0+1} \beta^{(0)})$ is the correction term for the output unit parameters in the first update iteration.

Consequently the current square training error is given by:

$$\begin{aligned} \epsilon^{(1)} &= [\mathbf{H}_1 \beta^{(1)} - \mathbf{T}_1]^T [\mathbf{H}_1 \beta^{(1)} - \mathbf{T}_1] \\ &= (\mathbf{H}_1 \beta^{(1)})^T (\mathbf{H}_1 \beta^{(1)}) - 2(\mathbf{H}_1 \beta^{(1)})^T \mathbf{T}_1 + \mathbf{T}_1^T \mathbf{T}_1 \end{aligned} \quad (4.27)$$

Inserting Equation (4.7), (4.25), and (4.26), we have:

$$\epsilon^{(1)} = \epsilon^{(0)} + \Delta\epsilon^{(0)} + \Delta\epsilon^{(1)} \quad (4.28)$$

where

$$\Delta\epsilon^{(0)} = (\mathbf{H}_0 \Delta\beta^{(1)})^T [(\mathbf{H}_0 \beta^{(0)} - \mathbf{T}_0) + (\mathbf{H}_0 \beta^{(1)} - \mathbf{T}_0)] \quad (4.29)$$

and

$$\Delta\epsilon^{(1)} = [\mathbf{h}_{n_0+1} \beta^{(1)} - \mathbf{t}_{n_0+1}]^T [\mathbf{h}_{n_0+1} \beta^{(1)} - \mathbf{t}_{n_0+1}] \quad (4.30)$$

$\epsilon^{(0)}$ is modified by the term $\Delta\epsilon^{(0)}$ caused by the correction for the output unit parameters in the first update iteration. Since $\epsilon^{(0)}$ is the least square error with respect to the initial linear system, $\Delta\epsilon^{(0)} \geq 0$. The additional square training error $\Delta\epsilon^{(1)}$ is introduced by the current training sample and not less than zero as well.

The above arguments can be generalized in a straightforward manner. For instance when the $(n_0 + j)^{th}$ training sample is presented, we have the output unit

parameters updated according to:

$$\beta^{(j)} = \beta^{(j-1)} + \Delta\beta^{(j)} \quad (4.31)$$

where $\Delta\beta^{(j)} = (\mathbf{H}_{j-1}^T \mathbf{H}_{j-1} + \mathbf{h}_{n_0+j}^T \mathbf{h}_{n_0+j})^{-1} \mathbf{h}_{n_0+j}^T (\mathbf{t}_{n_0+j}^T - \mathbf{h}_{n_0+j} \beta^{(j-1)})$. And the current square training error is given by:

$$\epsilon^{(j)} = \epsilon^{(j-1)} + \Delta\epsilon^{(j-1)} + \Delta\epsilon^{(j)} \quad (4.32)$$

where

$$\Delta\epsilon^{(j-1)} = (\mathbf{H}_{j-1} \Delta\beta^{(j)})^T [(\mathbf{H}_{j-1} \beta^{(j-1)} - \mathbf{T}_{j-1}) + (\mathbf{H}_{j-1} \beta^{(j)} - \mathbf{T}_{j-1})] \geq 0 \quad (4.33)$$

and

$$\Delta\epsilon^{(j)} = [\mathbf{h}_{n_0+j} \beta^{(j)} - \mathbf{t}_{n_0+j}]^T [\mathbf{h}_{n_0+j} \beta^{(j)} - \mathbf{t}_{n_0+j}] \geq 0 \quad (4.34)$$

Using the same arguments as in the first update iteration, we can imply that $\{\epsilon^{(j)}\}_{j=0}^m$ is a non-decreasing series.

For $j = 0$, if $n_0 = n$, then $\epsilon^{(0)} = 0$; Otherwise, it is a small positive value. For $j = m$, according to Equation (4.21), $\beta^m = \beta^*$, we obtain $\mathcal{N}_{G,n}^{(m)}(\mathbf{x}) = \mathcal{N}_{G,n}(\mathbf{x})$. Therefore, using Theorem 3.2, the approximate mapping condition is then satisfied, i.e., $\epsilon^{(m)} = \epsilon$. $\square$

OS-ELM in the learning mode of one-by-one is suitable for the online applications where one training sample is available at a time. However in other online applications, a chunk of training samples is presented at a time instead. Next **OS-ELM** in chunk-by-chunk learning mode is considered.

OS-ELM in Chunk-by-Chunk Learning Mode: For convenience, we first define a series of matrices $\{(\mathbf{X}_{n_j}, \mathbf{H}_{n_j}, \mathbf{T}_{n_j})\}_{j=0}^m$ and $\{(\mathbf{X}_{N_j}, \mathbf{H}_{N_j}, \mathbf{T}_{N_j})\}_{j=0}^m$ as follows:

For $j = 0$, $N_0 = n_0 \geq n$, and

$$\mathbf{X}_{n_0} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_{n_0}^T \end{bmatrix}_{n_0 \times d} \quad \mathbf{H}_{n_0} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_{n_0} \end{bmatrix}_{n_0 \times n} \quad \mathbf{T}_{n_0} = \begin{bmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \\ \vdots \\ \mathbf{t}_{n_0}^T \end{bmatrix}_{n_0 \times q} \quad (4.35)$$

For $j \neq 0$, $N_j = N_{j-1} + n_j$, $n_j \geq 1$, and

$$\mathbf{X}_{N_j} = \begin{bmatrix} \mathbf{X}_{N_{j-1}} \\ \mathbf{X}_{n_j} \end{bmatrix} \quad \mathbf{H}_{N_j} = \begin{bmatrix} \mathbf{H}_{N_{j-1}} \\ \mathbf{H}_{n_j} \end{bmatrix} \quad \mathbf{T}_{N_j} = \begin{bmatrix} \mathbf{T}_{N_{j-1}} \\ \mathbf{T}_{n_j} \end{bmatrix} \quad (4.36)$$

When $j = m$, $N_m = N$, $\mathbf{X}_{N_m} = \mathbf{X}$, $\mathbf{H}_{N_m} = \mathbf{H}$, and $\mathbf{T}_{N_m} = \mathbf{T}$.

Given a model of the SLFN with n hidden units using infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. Then it represents a set of input-output mapping functions defined by:

$$\mathcal{N}_{G,n}(x) = \sum_{i=1}^n \beta_i \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i) \quad (4.37)$$

Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ and fixed. Then the output unit parameters $\{\beta_i\}_{i=1}^n$ are iteratively updated according to the chunk-by-chunk formula of RLS algorithm. Since the similar manner is used in the derivation of the chunk-by-chunk formula of RLS algorithm as that of the one-by-one formula, we give the resulting update equations here. The details of derivation can be found in Appendix A.

$$\beta^{(N_j)} = \beta^{(N_{j-1})} + \mathbf{P}_{N_j} \mathbf{H}_{n_j}^T (\mathbf{T}_{n_j} - \mathbf{H}_{n_j} \beta^{(N_{j-1})}) \quad (4.38)$$

and

$$\mathbf{P}_{N_j} = \mathbf{P}_{N_{j-1}} - \mathbf{P}_{N_{j-1}} \mathbf{H}_{n_j}^T (\mathbf{I} + \mathbf{H}_{n_j} \mathbf{P}_{N_{j-1}} \mathbf{H}_{n_j}^T)^{-1} \mathbf{H}_{n_j} \mathbf{P}_{N_{j-1}} \quad (4.39)$$

Equations (4.38) and (4.39) give the chunk-by-chunk formula of RLS algorithm, which is used in **OS-ELM** for iteratively updating the output unit parameters of the SLFN. More importantly, when $j = m$, the final linear system is given by:

$$\mathbf{H}_{N_m} \beta = \mathbf{T}_{N_m} \quad (4.40)$$

and consequently the output unit parameters estimated by **OS-ELM** are identical with those determined by batch **ELM**:

$$\beta^* = \beta^{(N_m)} \quad (4.41)$$

for $\mathbf{H} = \mathbf{H}_{N_m}$ and $\mathbf{T} = \mathbf{T}_{N_m}$.

To proceed further, we define a series of functions $\{\mathcal{N}_{G,n}^{(N_j)}(\mathbf{x})\}_{j=0}^m$ as follows:

Definition 4.3.

$$\mathcal{N}_{G,n}^{(N_j)}(\mathbf{x}) = \sum_{i=1}^n \beta_i^{(N_j)} \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i) \quad (4.42)$$

and the present square training error (summed over all the training samples learned so far) can be defined by:

Definition 4.4.

$$\epsilon^{(N_j)} = \sum_{k=1}^{N_j} \|\mathcal{N}_{G,n}^{(N_j)}(\mathbf{x}_k) - \mathbf{t}_k\|^2 \quad (4.43)$$

Now we can formally state the theorem of **OS-ELM** in the learning mode of chunk-by-chunk as follows:

Theorem 4.3. *Given a series of $m + 1$ chunks of distinct input-output samples $\{(\mathbf{X}_{n_i}, \mathbf{T}_{n_i}) | \mathbf{X}_{n_i} \in \mathbb{R}^{n_i \times d}, \mathbf{T}_{n_i} \in \mathbb{R}^{n_i \times q}\}_{i=0}^m$ and $\epsilon > 0$. Let $\mathcal{N}_{G,n} = \sum_{i=1}^n \beta_i \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i)$ be realized by a SLFN with n hidden units using infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. Let the first n_0 samples be the initial data, where $N \geq n_0 \geq n$. If the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ are randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution, then the output unit parameters $\{\beta_i\}_{i=1}^n$ are iteratively updated according to the chunk-by-chunk formula of RLS such that the present square training error approaches the approximate mapping condition: $\lim_{j \rightarrow m} \epsilon^{(N_j)} = \epsilon$.*

Proof: The proof is similar as in Theorem 4.2. For $j = 0$, **OS-ELM** in chunk-by-chunk learning mode is identical with **OS-ELM** in one-by-one learning mode, i.e., given the first n_0 distinct input-output samples. Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution and $N \geq n_0 \geq n$. Then the initial linear system of equations for the output unit parameters is given by $\mathbf{H}_{N_0} \beta = \mathbf{T}_{N_0}$, where $N_0 = n_0$. By solving the initial linear system, we have

$$\beta^{(N_0)} = (\mathbf{H}_{N_0}^T \mathbf{H}_{N_0})^{-1} \mathbf{H}_{N_0}^T \mathbf{T}_{N_0} \quad (4.44)$$

Again we assume the output of the SLFN is scalar (i.e., $q = 1$) such that the initial square training error is given by:

$$\begin{aligned} \epsilon^{(N_0)} &= [\mathbf{H}_{N_0} \beta^{(N_0)} - \mathbf{T}_{N_0}]^T [\mathbf{H}_{N_0} \beta^{(N_0)} - \mathbf{T}_{N_0}] \\ &= (\mathbf{H}_{N_0} \beta^{(N_0)})^T (\mathbf{H}_{N_0} \beta^{(N_0)}) - 2(\mathbf{H}_{N_0} \beta^{(N_0)})^T \mathbf{T}_{N_0} + \mathbf{T}_{N_0}^T \mathbf{T}_{N_0} \end{aligned} \quad (4.45)$$

For $j = 1$, we assume the first chunk of n_1 input-output samples is presented, and the output unit parameters are updated according to:

$$\beta^{(N_1)} = \beta^{(N_0)} + \Delta\beta^{(n_1)} \quad (4.46)$$

where $N_1 = N_0 + n_1$, and $\Delta\beta^{(n_1)} = (\mathbf{H}_{N_0}^T \mathbf{H}_{N_0} + \mathbf{H}_{n_1}^T \mathbf{H}_{n_1})^{-1} \mathbf{H}_{n_1}^T (\mathbf{T}_{n_1}^T - \mathbf{H}_{n_1} \beta^{(N_0)})$ is the correction term for the output unit parameters caused by the first chunk of samples.

Consequently the current square training error is given by:

$$\begin{aligned} \epsilon^{(N_1)} &= [\mathbf{H}_{N_1} \beta^{(N_1)} - \mathbf{T}_{N_1}]^T [\mathbf{H}_{N_1} \beta^{(N_1)} - \mathbf{T}_{N_1}] \\ &= (\mathbf{H}_{N_1} \beta^{(N_1)})^T (\mathbf{H}_{N_1} \beta^{(N_1)}) - 2(\mathbf{H}_{N_1} \beta^{(N_1)})^T \mathbf{T}_{N_1} + \mathbf{T}_{N_1}^T \mathbf{T}_{N_1} \end{aligned} \quad (4.47)$$

Using Equations (4.36), (4.45), and (4.46), we have:

$$\epsilon^{(N_1)} = \epsilon^{(N_0)} + \Delta\epsilon^{(N_0)} + \Delta\epsilon^{(n_1)} \quad (4.48)$$

where

$$\Delta\epsilon^{(N_0)} = (\mathbf{H}_{N_0} \Delta\beta^{(n_1)})^T [(\mathbf{H}_{N_0} \beta^{(N_0)} - \mathbf{T}_{N_0}) + (\mathbf{H}_{N_0} \beta^{(N_1)} - \mathbf{T}_{N_0})] \quad (4.49)$$

and

$$\Delta\epsilon^{(n_1)} = [\mathbf{H}_{n_1} \beta^{(N_1)} - \mathbf{T}_{n_1}]^T [\mathbf{H}_{n_1} \beta^{(N_1)} - \mathbf{T}_{n_1}] \quad (4.50)$$

$\epsilon^{(N_0)}$ is modified by the term $\Delta\epsilon^{(N_0)}$ caused by the correction for the output unit parameters in the first update iteration. Since $\epsilon^{(N_0)}$ is the least square error with respect to the initial linear system, $\Delta\epsilon^{(N_0)} \geq 0$. The additional square training error $\Delta\epsilon^{(n_1)}$ is introduced by the chunk of current training samples and not less than zero as well.

The above arguments can be generalized in a straightforward manner. For instance when the j^{th} chunk of n_j training samples is presented, we have the output unit parameters updated according to:

$$\beta^{(N_j)} = \beta^{(N_{j-1})} + \Delta\beta^{(n_j)} \quad (4.51)$$

where $\Delta\beta^{(n_j)} = (\mathbf{H}_{N_{j-1}}^T \mathbf{H}_{N_{j-1}} + \mathbf{H}_{n_j}^T \mathbf{H}_{n_j})^{-1} \mathbf{H}_{n_j}^T (\mathbf{T}_{n_j}^T - \mathbf{H}_{n_j} \beta^{(N_{j-1})})$. And the current square training error is given by:

$$\epsilon^{(N_j)} = \epsilon^{(N_{j-1})} + \Delta\epsilon^{(N_{j-1})} + \Delta\epsilon^{(n_j)} \quad (4.52)$$

where

$$\Delta\epsilon^{(N_{j-1})} = (\mathbf{H}_{N_{j-1}} \Delta\beta^{(n_j)})^T [(\mathbf{H}_{N_{j-1}} \beta^{(N_{j-1})} - \mathbf{T}_{N_{j-1}}) + (\mathbf{H}_{N_{j-1}} \beta^{(N_j)} - \mathbf{T}_{N_{j-1}})] \geq 0 \quad (4.53)$$

and

$$\Delta\epsilon^{(n_j)} = [\mathbf{H}_{n_j} \beta^{(N_j)} - \mathbf{T}_{n_j}]^T [\mathbf{H}_{n_j} \beta^{(N_j)} - \mathbf{T}_{n_j}] \geq 0 \quad (4.54)$$

$\{\epsilon^{(N_j)}\}_{j=0}^m$ is a non-decreasing series. For $j = 0$, if $N_0 = n$, then $\epsilon^{(N_0)} = 0$; Otherwise, it is a small positive value. For $j = m$, according to Equation (4.41), $\beta^{N_m} = \beta^*$, we obtain $\mathcal{N}_{G,n}^{(N_m)}(\mathbf{x}) = \mathcal{N}_{G,n}(\mathbf{x})$. According to Theorem 3.2, the approximate mapping condition is then satisfied, i.e., $\epsilon^{(m)} = \epsilon$. $\square$

Remark 1: Theorem 4.2 and Theorem 4.3 show that **OS-ELM** both in the learning mode of one-by-one and chunk-by-chunk randomly assigns the hidden unit parameters without any adjustment throughout the learning evolution. It only iteratively updates the output unit parameters according to the algorithm of RLS

based on the current training sample(s). Such strategy produces a great reduction of computational complexity in each update iteration.

Remark 2: When $n_0 = N$, **OS-ELM** becomes the same as batch **ELM**. When $N > n_0 > n$ and $n_j = 1$ ($j \neq 0$), **OS-ELM** operates in one-by-one learning mode. For other cases, **OS-ELM** operates in chunk-by-chunk learning mode.

Remark 3: Theorem 4.2 and Theorem 4.3 have proven that **OS-ELM** realizes the identical input-output mapping function with batch **ELM** given the same training samples and initial model of a SLFN. Therefore, from the theoretical viewpoint, all the desirable properties of batch **ELM** are inherited by **OS-ELM**, viz., improved generalization performance, reduced computational complexity, and activation function unified training framework.

Remark 4: As far as the stability of **OS-ELM** is concerned, **OS-ELM** preserves the desirable convergence properties of RLS algorithm [6, 17]. Further Theorem 4.2 has proven that $\lim_{j \rightarrow m} \beta^{(j)} = \beta^*$ and $\lim_{j \rightarrow m} \epsilon^{(j)} = \epsilon$ in case of **OS-ELM** implemented in one-by-one learning mode. They also hold true for **OS-ELM** in the chunk-by-chunk learning mode as stated in Theorem 4.3.

4.2.2 Algorithm Implementation

Given a model of the SLFN with n hidden units using some infinitely differentiable activation function, the learning process of **OS-ELM** consists of two phases, viz., initialization and update phase. In the initialization phase, the number of initial samples is at least equal to the number of hidden units. According to Lemma 3.1 and Lemma 3.2, when $n_0 = n$, $\text{rank}(\mathbf{H}_0) = n$ provided that the initial input samples are distinct. Otherwise, more samples are required. In the update phase, **OS-ELM** operates in the sequential learning mode of one-by-one or chunk-by-chunk. For the former, the hidden layer output row vector $\mathbf{h}_{n_0+j}$ is estimated and the output unit parameters are updated according to the one-by-one formula of RLS algorithm in each iteration. For the latter, the *partial* hidden layer output

matrix $\mathbf{H}_{n_j}$ is calculated and the output unit parameters are updated according to the chunk-by-chunk formula of RLS algorithm in each iteration. And the chunk size $n_j \geq 1$ for $j \neq 0$ may vary in each iteration.

Now we can summarize the implementation of **OS-ELM** in one-by-one and chunk-by-chunk learning mode as follows:

OS-ELM in One-by-One Learning Mode: Given a SLFN with n hidden units using an infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. For a series of input-output samples $\{(\mathbf{x}_k, \mathbf{t}_k) | \mathbf{x}_k \in \mathbb{R}^d, \mathbf{t}_k \in \mathbb{R}^q, k = 1, 2, \dots\}$,

step 1 Initialization Phase: An initial training data set $\{(\mathbf{x}_k, \mathbf{t}_k)\}_{k=1}^{n_0}$ ($n_0 \geq n$) is accumulated.

- (a) Assign random values for the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$.
- (b) Calculate the initial hidden layer output matrix $\mathbf{H}_0$:

$$\mathbf{H}_0 = \begin{bmatrix} G(\mathbf{x}_1, \mathbf{a}_1, b_1) & \cdots & G(\mathbf{x}_1, \mathbf{a}_n, b_n) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{n_0}, \mathbf{a}_1, b_1) & \cdots & G(\mathbf{x}_{n_0}, \mathbf{a}_n, b_n) \end{bmatrix}_{n_0 \times n} \quad (4.55)$$

- (c) Estimate the initial output unit parameters $\beta^{(0)}$ according to:

$$\mathbf{P}_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1} \quad (4.56)$$

$$\beta^{(0)} = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{T}_0 \quad (4.57)$$

- (d) Set $j = 1$.

step 2 Update Phase: When the $(n_0 + j)^{th}$ training sample $\{(\mathbf{x}_{n_0+j}, \mathbf{t}_{n_0+j})\}$ is available, do:

- (a) Calculate the hidden layer output row vector $\mathbf{h}_{n_0+j}$:

$$\mathbf{h}_{n_0+j} = [G(\mathbf{x}_{n_0+j}, \mathbf{a}_1, b_1), G(\mathbf{x}_{n_0+j}, \mathbf{a}_2, b_2), \dots, G(\mathbf{x}_{n_0+j}, \mathbf{a}_n, b_n)] \quad (4.58)$$

(b) Update the output unit parameters $\beta^{(j)}$ according to:

$$\begin{aligned}\mathbf{P}_j &= \mathbf{P}_{j-1} - \frac{\mathbf{P}_{j-1} \mathbf{h}_{n_0+j}^T \mathbf{h}_{n_0+j} \mathbf{P}_{j-1}}{1 + \mathbf{h}_{n_0+j} \mathbf{P}_{j-1} \mathbf{h}_{n_0+j}^T} \\ \beta^{(j)} &= \beta^{(j-1)} + \mathbf{P}_j \mathbf{h}_{n_0+j}^T (\mathbf{t}_{n_0+j}^T - \mathbf{h}_{n_0+j} \beta^{(j-1)})\end{aligned}\quad (4.59)$$

(c) Set $j = j + 1$. Repeat step 2.

OS-ELM in Chunk-by-Chunk Learning Mode: Given a SLFN with n hidden units using an infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$. For a series of chunks of input-output samples $\{(\mathbf{X}_{n_k}, \mathbf{T}_{n_k}) | \mathbf{X}_{n_k} \in \mathbb{R}^{n_k \times d}, \mathbf{T}_{n_k} \in \mathbb{R}^{n_k \times q}, k = 1, 2, \dots\}$,

step 1 Initialization Phase: An initial training data set $\{(\mathbf{x}_k, \mathbf{t}_k)\}_{k=1}^{n_0}$ ($n_0 \geq n$) is accumulated.

(a) Assign random values for the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$.

(b) Calculate the initial hidden layer output matrix $\mathbf{H}_{n_0}$:

$$\mathbf{H}_{n_0} = \begin{bmatrix} G(\mathbf{x}_1, \mathbf{a}_1, b_1) & \cdots & G(\mathbf{x}_1, \mathbf{a}_n, b_n) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{n_0}, \mathbf{a}_1, b_1) & \cdots & G(\mathbf{x}_{n_0}, \mathbf{a}_n, b_n) \end{bmatrix}_{n_0 \times n} \quad (4.60)$$

(c) Estimate the initial output unit parameters $\beta^{(N_0)}$ according to:

$$\mathbf{P}_{N_0} = (\mathbf{H}_{n_0}^T \mathbf{H}_{n_0})^{-1} \quad (4.61)$$

$$\beta^{(N_0)} = \mathbf{P}_{N_0} \mathbf{H}_{n_0}^T \mathbf{T}_{n_0} \quad (4.62)$$

(d) Set $j = 1$.

step 2 Update Phase: When the j^{th} chunk of training samples $\{(\mathbf{x}_k, \mathbf{t}_k)\}_{k=1}^{n_j}$ ($n_j \geq 1$) is presented, do:

(a) Calculate the partial hidden layer output matrix $\mathbf{H}_{n_j}$:

$$\mathbf{H}_{n_j} = \begin{bmatrix} G(\mathbf{x}_1, \mathbf{a}_1, b_1) & \cdots & G(\mathbf{x}_1, \mathbf{a}_n, b_n) \\ \vdots & \cdots & \vdots \\ G(\mathbf{x}_{n_j}, \mathbf{a}_1, b_1) & \cdots & G(\mathbf{x}_{n_j}, \mathbf{a}_n, b_n) \end{bmatrix}_{n_j \times n} \quad (4.63)$$

(b) Update the output unit parameters $\beta^{(N_j)}$ according to:

$$\begin{aligned} \mathbf{P}_{N_j} &= \mathbf{P}_{N_{j-1}} - \mathbf{P}_{N_{j-1}} \mathbf{H}_{n_j}^T (\mathbf{I} + \mathbf{H}_{n_j} \mathbf{P}_{N_{j-1}} \mathbf{H}_{n_j}^T)^{-1} \mathbf{H}_{n_j} \mathbf{P}_{N_{j-1}} \\ \beta^{(N_j)} &= \beta^{(N_{j-1})} + \mathbf{P}_{N_j} \mathbf{H}_{n_j}^T (\mathbf{T}_{n_j} - \mathbf{H}_{n_j} \beta^{(N_{j-1})}) \end{aligned} \quad (4.64)$$

(c) Set $j = j + 1$. Repeat step 2.

Remark 5: In order to handle the case where the terms of $\mathbf{H}_0^T \mathbf{H}_0$ and $\mathbf{I} + \mathbf{H}_{n_j} \mathbf{P}_{N_j} \mathbf{H}_{n_j}^T$ are nearly ill-conditioned and to make the implementation of **OS-ELM** in finite precision arithmetic numerically stable, the update for β and $\mathbf{P}$ can be calculated using singular value decomposition (SVD) [51] as done in our simulations.

4.3 Quantitative Comparison With Other Sequential Learning Algorithms

We compare **OS-ELM** with other popular sequential learning algorithms with respect to the mode of learning, type of activation functions, and model selection.

4.3.1 Mode of Learning

We observe that SGBP and recursive Marquardt-Levenberg algorithm regularly work in the learning mode of one-by-one, i.e., the learning is based on the information of a single sample in each update iteration. RAN-based algorithms also operate in this learning mode. In the derivation of **OS-ELM**, we have shown that the algorithm can operate not only in the learning mode of one-by-one, but

also chunk-by-chunk. In the learning mode of chunk-by-chunk, the parameters of a SLFN are iteratively updated on the basis of a chunk of incoming samples. Further the chunk size may vary in each iteration. Thus **OS-ELM** is more flexible than other sequential learning algorithms with respect to the mode of learning.

4.3.2 Type of Activation Functions

SGBP and recursive Marquardt-Levenberg algorithm are specific for training feed-forward neural networks with differentiable ridge activation function. RAN, RANEKF, MRAN, GAP-RBF, and GGAP-RBF are specific for training RBF neural networks. **OS-ELM** can handle both SLFNs with ridge activation functions and SLFNs with RBF in a unified framework.

4.3.3 Model Selection

The application of the algorithms for a given problem requires model selection in order to achieve a proper generalization performance. Model selection is the procedure for selecting the network size, learning parameters, and/or control parameters. For SGBP and recursive Marquardt-Levenberg algorithm, it includes the selection of the network size (e.g., the number of hidden units for SLFNs), and learning parameters (e.g., learning rate and momentum constant, etc). RAN, RANEKF, MRAN, GAP-RBF, and GGAP-RBF involve many control parameters, which include distance parameters ($\epsilon_{\max}$, γ , $\epsilon_{\min}$) and impact factor adjustment parameter κ . Besides these common control parameters, MRAN introduces the growing and pruning control parameters; GAP-RBF and GGAP-RBF require the information of the input sampling distributions. The application of **OS-ELM** only involves the selection of the hidden unit number. Thus the model selection procedure for **OS-ELM** is relatively simple compared with other sequential learning algorithms.

4.4 Performance Evaluation of OS-ELM

In this section, we provide the performance comparison of **OS-ELM** with other algorithms in the learning mode of one-by-one on seven benchmark problems. These algorithms include SGBP, RAN, RANEKF, MRAN, GAP-RBF, and GGAP-RBF. Moreover, the performance of **OS-ELM** in the learning mode of chunk-by-chunk is evaluated and compared with batch **ELM** on the same benchmark problems.

Dataset	# Input Attributes	# Classes	# Training Data	# Testing Data
Auto-MPG	7	-	320	72
Abalone	8	-	3,000	1,177
California Housing	8	-	8,000	12,640
Image Segment	19	7	1,500	810
Satellite Image	36	6	4,435	2,000
DNA	180	3	2,000	1,186
Mackey Glass	4	-	4,000	500

Table 4.1: Specifications of the benchmark problems.

Table 4.1 provides the specifications of the benchmark problems, which include regression (Auto-MPG, Abalone, California Housing) [7], classification (Image Segment, Satellite Image, DNA) [7], and time series prediction application (Mackey-Glass time series) [40]. It is to be noted that the input-output samples for regression and classification problems are stationary processes; while for time series prediction problem, they are nonstationary. In our simulations, the input and output attributes in regression application are normalized into the range $[0, 1]$, while the input attributes in classification application are normalized into the range $[-1, 1]$. For the time series prediction application, no scaling is applied.

The activation functions used in the algorithms are specified as follows: SGBP is simulated to train SLFNs using sigmoidal activation function: $Sigmoid(\mathbf{a}^T \mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{a}^T \mathbf{x} + b)}}$. RAN, RANEKF, MRAN, GAP-RBF, and GGAP-RBF are simulated to construct SLFNs using gaussian activation function: $Gauss(\mathbf{x}, \mathbf{a}, b) = e^{-\frac{\|\mathbf{x} - \mathbf{a}\|^2}{b}}$. For **OS-ELM**, it is simulated to construct both SLFNs using sigmoidal activation function and SLFNs using gaussian activation function.

All the simulations have been conducted in MATLAB 6.5 environment running on an ordinary PC with 1.9 GHZ CPU.

4.4.1 Model Selection in Simulations

As analyzed in Section 4.3.3, **OS-ELM** requires the number of hidden units to be specified. SGBP requires the optimal combination of the hidden unit number, learning rate and momentum constant. For RAN, RANERF, MRAN, GAP-RBF, and GGAP-RBF, their control parameters, including distance parameters ($\epsilon_{\max}$, γ , $\epsilon_{\min}$) and impact factor adjustment parameter κ , should be tuned by trial and error. Additionally, for MRAN, the growing and pruning thresholds need to be determined; for GAP-RBF and GGAP-RBF, the input sampling distribution has to be estimated as well.

Here we give the example for the selection of the number of hidden units for **OS-ELM** and SGBP. The selection procedure makes use of the training and validation set method. Training data set is separated into two non-overlapped subsets: one for training and the other for validation. The training and validation error are then estimated on the basis of these two data sets (For regression application and Mackey-Glass time series prediction, root mean square error (RMSE) is estimated; while for classification, the classification error rate is calculated). The optimal number of hidden units is selected as the one which results in the lowest average validation error over 50 trials. An example of model selection in the Auto-MPG case is provided in Figure 4.1. In the figure, the top thin dotted and solid curves represent the average training and validation RMSEs, respectively, for SGBP. And the bottom thick dotted and solid curves are for **OS-ELM** using sigmoidal activation function and training in the learning mode of one-by-one. As observed from the average validation curves in the figure, the lowest average validation RMSEs are achieved when the number of hidden units are in the ranges [15-35] and [13-24] for **OS-ELM** (Sigmoid) and SGBP, respectively. Therefore the estimated optimal number of hidden units for **OS-ELM** (Sigmoid) and SGBP are falling into these

ranges in the Auto-MPG case.

Figure 4.1: An example of model selection for **OS-ELM**(Sigmoid) and SGBP in the learning mode of one-by-one in the Auto-MPG case: Average RMSEs are showed as a function of the number of hidden units.

Figure 4.2: The learning evolution patterns for **OS-ELM** (with 25 hidden units using sigmoidal activation function) and for SGBP (with 13 hidden units) in the learning mode of one-by-one on a single trial in model selection for the Auto-MPG case.

From Figure 4.1 we observe that the average RMSE curves for **OS-ELM** are smoother compared with SGBP, which implies that **OS-ELM** is less sensitive to the network size. Figure 4.2 unwraps such behavior in the learning evolution pattern for **OS-ELM** (with 25 hidden units using the sigmoidal activation function and training in the learning mode of one-by-one) and SGBP (with 13 hidden units) on a single trial in the model selection in the Auto-MPG case. The dotted curves represent the training RMSEs, which are estimated on the basis of the training samples learned so far in each step. The solid curves represent the validation RMSEs, which are estimated over the entire validation samples. From the figure, it shows the validation RMSE of **OS-ELM** decreases much faster than SGBP at the begin of the learning evolution. And roughly after 175 training samples, it converges to the optimal value of validation RMSE. Contrarily, the validation RMSE of SGBP “ridges” throughout its learning evolution.

The above procedure is repeated to search the optimal number of hidden units for **OS-ELM** and SGBP in all benchmark problems. Further the optimal combination of the learning parameters for SGBP as well as the optimal combination of the control parameters for the RAN-based sequential algorithms are also estimated. All the selected model parameters are applied in the following performance evaluation of the algorithms.

In each section that follows, we first evaluate the performance of **OS-ELM** and compare it with other algorithms in the the learning mode of one-by-one on the benchmark problems. Then we evaluate the performance of **OS-ELM** in the learning mode of chunk-by-chunk and compare it with batch **ELM** on the same benchmark problems. The performance of the algorithms are averaged over 50 independent trials for each problem. For regression and time series prediction applications, average training time, training and testing root mean square error (RMSE) are considered. For classification problems, average training time, training and testing correct classification rate (accuracy) are estimated.

4.4.2 OS-ELM Performance Evaluation on Regression Problems

In this section, we present the application of **OS-ELM** to scalar regression problems, which can be viewed as scalar function approximation using SLFNs in Figure 2.1.

Three benchmark problems of regression have been studied, viz., Auto-MPG, Abalone, and California Housing [7]. The Auto-MPG problem is to predict the fuel consumption (miles per gallon) of different models of car. The Abalone problem is the estimation of the age of abalone from the physical measurements. And the California Housing problem is to predict the median California housing price based on the information collected using all the chunk groups in California from the 1990 Census. For all the problems studied here, the training and testing data are randomly generated for each trial.

(i) One-by-One Learning Mode:

Table 4.2 summarizes the performance comparison of **OS-ELM** with other algorithms in the learning mode of one-by-one on these regression problems. In the table, the average training time, training and testing RMSE, and number of hidden units are presented. For **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and SGBP, the number of hidden units is chosen from the optimal range as estimated in model selection procedure. For RAN, RANEKF, MRAN, and GGAP-RBF, it is learned by the algorithm itself. Additionally, for **OS-ELM** (Sigmoid) and **OS-ELM** (Gauss), the number of initial training samples is set equal to the optimal number of hidden units. From the table, we observe that the training time taken by **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and SGBP is much less than RAN, RANEKF, MRAN, GAP-RBF, and GGAP-RBF. Further, out of all algorithms, **OS-ELM** (Sigmoid) and **OS-ELM** (Gauss) produce the lowest testing RMSEs. Additionally the performances of both **OS-ELMs** are comparable to each other except that **OS-ELM** (Gauss) requires more training time than **OS-ELM** (Sigmoid).

Datasets	Algorithms	Time (seconds)	RMSE		# hidden units
			Training	Testing	
Auto-MPG	OS-ELM (Sigmoid)	0.0326	0.0695	0.0696	25
	OS-ELM (Gauss)	0.0987	0.0694	0.0695	25
	SGBP	0.0875	0.1112	0.1028	13
	GAP-RBF	0.4520	0.1144	0.1404	3.12
	MRAN	1.4644	0.1086	0.1376	4.46
	RANEKF	1.0103	0.1088	0.1387	5.14
	RAN	0.8042	0.2923	0.3080	4.44
Abalone	OS-ELM (Sigmoid)	0.3154	0.0762	0.0775	25
	OS-ELM (Gauss)	0.9776	0.0760	0.0773	25
	SGBP	0.7472	0.0996	0.0972	11
	GAP-RBF	83.784	0.0963	0.0966	23.62
	MRAN	1500.4	0.0836	0.0837	87.571
	RANEKF	90806	0.0738	0.0794	409
	RAN	105.17	0.0931	0.0978	345.58
California Housing	OS-ELM (Sigmoid)	1.8400	0.1305	0.1335	50
	OS-ELM (Gauss)	5.3331	0.1291	0.1312	50
	SGBP	1.6866	0.1688	0.1704	9
	GGAP-RBF	115.34	0.1417	0.1386	18
	MRAN	2891.5	0.1598	0.1586	64
	RANEKF	14181	0.0736	0.1495	200
	RAN	3505.2	0.1083	0.1531	3552

Table 4.2: Performance comparison between **OS-ELM** and other algorithms in the learning mode of one-by-one on the regression problems. The number of hidden units for RAN, RANEKF, MRAM, GAP-RBF, and GGAP-RBF is the average over 50 trials.

Figure 4.3, 4.4, and 4.5 detail the learning evolution patterns for **OS-ELM** (Sigmoid) and SGBP on a single trial in the cases of Auto-MPG, Abalone, and California Housing, respectively. In the figures, the training and testing RMSEs is represented as functions of the number of training samples. The dotted curves illustrate the training RMSEs, which are estimated over the training samples learned so far in each iteration. The solid curves show the testing RMSEs, which are calculated over the entire testing samples.

As observed from Figure 4.3, in the Auto-MPG problem, the learning evolution for **OS-ELM** (Sigmoid) starts after the first 25 training samples have been used to complete the initialization phase. At this moment, the training RMSE of the algorithm is zero, while testing RMSE is relatively large. Afterwards the update phase of the algorithm commences: the training RMSE increases slightly as more training samples present, whereas testing RMSE drops sharply at the beginning of update phase. And subsequently training and testing RMSEs converge smoothly. Roughly after 175 training samples have been learned, no more obvious change can be observed with the training and testing RMSEs. On the Contrary, for SGBP the parameters of the SLFN are updated at the presentation of the first training sample. Its training and testing RMSEs are alike. The convergence speed of the testing RMSE is slower than **OS-ELM**. And the final testing RMSE is larger than **OS-ELM** as well. Moreover, there are “ridges” randomly distributed throughout the learning evolution process of SGBP. Similar patterns can be observed in Figure 4.4 and 4.5, which correspond to Abalone and California Housing problems.

Figure 4.3: The learning evolution patterns for **OS-ELM** (with 25 hidden units using sigmoidal activation function) and for SGBP (with 13 hidden units) in the learning mode of one-by-one on a single trial in the Auto-MPG case.

Figure 4.4: The learning evolution patterns for **OS-ELM** (with 25 hidden units using sigmoidal activation function) and for SGBP (with 11 hidden units) in the learning mode of one-by-one on a single trial in the Abalone case.

Figure 4.5: The learning evolution patterns for **OS-ELM** (with 50 hidden units using sigmoidal activation function) and SGBP (with 9 hidden units) in the learning mode of one-by-one on a single trial in the California Housing case.

(ii) Chunk-by-Chunk Learning Mode:

The performance of **OS-ELM** in chunk-by-chunk learning mode is described here. For other sequential learning algorithms, they may not operate in the the learning mode of chunk-by-chunk. For instance, SGBP learning algorithm operates only in one-by-one learning mode. RAN-based learning algorithms also work in the learning mode of one-by-one only. From this viewpoint, we can not conduct the comparison of **OS-ELM** with these algorithms. In stead we compare **OS-ELM** with batch **ELM**.

Table 4.3 presents the performance of **OS-ELM** in various learning modes, viz., one-by-one, 20-by-20, and [10,30] (the number of samples for each iteration is randomly picked from the range between 10 and 30). From theoretical analysis, the performance of **OS-ELM** should be comparable with batch **ELM**. Again the average training time, training and testing RMSEs, and number of hidden units are considered here.

In the table, the simulation results show that the average training and testing RMSEs obtained by batch **ELM** and **OS-ELM** (both for sigmoidal and gaussian activation functions in various learning modes) are very close. This is consistent with the theoretical analysis in Section 4.2. The slight difference of the average training and testing RMSEs between batch **ELM** and **OS-ELM** is probably caused by the different initial values for the hidden unit parameters, which are randomly chosen for each learning mode and each trial from a uniform distribution in the range $[-1,1]$.

As far as the training time is concerned, **OS-ELM** in the learning mode of one-by-one takes the longest time followed by [10, 30], 20-by-20 and finally batch **ELM**. This implies that the learning mode of one-by-one takes the longest time while the batch learning mode requires the shortest time and any other learning mode with different chunk size falls in between. If the chunk size is large, it approaches the training time taken by the batch learning mode.

However, there is an interesting exception in the California Housing problem. In

Datasets	Activation Functions	Algorithms	Learning Mode	Time (seconds)	RMSE		# hidden units
					Training	Testing	
Auto - MPG	Sigmoid	ELM	Batch	0.0053	0.0697	0.0694	25
		OS-ELM	1-by-1	0.0326	0.0695	0.0696	25
			20-by-20	0.0075	0.0696	0.0692	25
			[10,30]	0.0119	0.0693	0.0695	25
	Gauss	ELM	Batch	0.0100	0.0691	0.0694	25
		OS-ELM	1-by-1	0.0987	0.0694	0.0695	25
			20-by-20	0.0147	0.0692	0.0698	25
			[10,30]	0.0206	0.0689	0.0698	25
Abalone	Sigmoid	ELM	Batch	0.0497	0.0763	0.0771	25
		OS-ELM	1-by-1	0.3154	0.0762	0.0775	25
			20-by-20	0.0682	0.0762	0.0771	25
			[10,30]	0.1365	0.0763	0.0770	25
	Gauss	ELM	Batch	0.0972	0.0761	0.0770	25
		OS-ELM	1-by-1	0.9776	0.0760	0.0773	25
			20-by-20	0.1291	0.0760	0.0776	25
			[10,30]	0.1966	0.0761	0.0772	25
California Housing	Sigmoid	ELM	Batch	0.5122	0.1306	0.1333	50
		OS-ELM	1-by-1	1.8400	0.1305	0.1335	50
			20-by-20	0.3506	0.1300	0.1330	50
			[10,30]	0.4787	0.1302	0.1333	50
	Gauss	ELM	Batch	1.0210	0.1292	0.1312	50
		OS-ELM	1-by-1	5.3331	0.1291	0.1312	50
			20-by-20	0.7422	0.1291	0.1314	50
			[10,30]	0.9828	0.1291	0.1312	50

Table 4.3: Performance comparison between batch **ELM** and **OS-ELM** in the learning mode of chunk-by-chunk on the regression problems. [10,30] means that the number of samples is randomly picked in the range of 10 to 30 for each iteration.

this problem, the training time taken by the learning mode of 20-by-20 and [10,30] is even smaller than the batch learning mode. One possible explanation is that: In the case where the size of the entire training data set is large, the dimension of the hidden layer output matrix is huge as well. Therefore more RAM space is required to manipulate the matrix for batch learning mode. When this space exceeds a limit, e.g., cache memory limit, the operations slow down, and the training time increases. On the other hand, for the learning mode of chunk-by-chunk, it does not require to manipulate with the whole hidden layer output matrix but the partial hidden layer output matrix in each iteration. The space required to store the partial hidden layer output matrix is relatively small. Therefore the training data size has very little effect on the operation speed. Thus the training time taken by the learning mode of chunk-by-chunk is less than the batch mode in such situation.

Figure 4.6 shows the learning evolution patterns of **OS-ELM** in various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Auto-MPG case. For the comparison purpose, the training and testing RMSEs produced by batch **ELM** (on a single trial) are also shown in the figure. In the figure, y-axis is confined to the range [0, 0.2] in order to focus on the details of update phase of the algorithm. The training and testing RMSEs are presented as the functions of the number of training samples. The thin dotted and solid curves present the training and testing RMSEs of **OS-ELM** with sigmoidal activation function in the learning mode of one-by-one. The thick dotted and solid curves are for **OS-ELM** with sigmoid activation function in the learning mode of 20-by-20. The thin and thick dash-dotted curves correspond to **OS-ELM** in the learning mode of [10,30]. The asterisks in small and large size represent the training and testing RMSEs of batch **ELM** and are depicted in the end of learning evolution of **OS-ELM**.

In the initialization phase of **OS-ELM** in various learning modes, the number of the initial training samples is set equal to the optimized number of hidden units. As shown in the figure, the learning evolution of **OS-ELM** (Sigmoid) starts after the first 25 training samples have been used for completing the initialization. At this

moment, the training RMSEs of **OS-ELM** in all learning modes are zero, while testing RMSEs are relatively large. In the update phase followed by, the changes in the training and testing RMSE curves for all the learning modes are alike: training RMSEs increase slightly as more training samples present, whereas testing RMSEs drop sharply at the beginning of update phase. Subsequently training and testing RMSEs converge smoothly. And observed from the figure, their final training and testing RMSEs are identical with those of batch **ELM**, which are marked by the symbol of asterisks in the small and large size, respectively. The differences among the learning evolution patterns of **OS-ELM** in various learning modes are caused by two facts: the order of the training samples presented as well as the initial values for the hidden unit parameters. And the tiny differences of the final training and testing RMSEs between **OS-ELM** and batch **ELM** are caused by the initial values for the hidden unit parameters only. Similar behaviors have been observed in Figure 4.7 and Figure 4.8 for Abalone and California Housing problem, respectively.

Figure 4.6: The learning evolution patterns of **OS-ELM** (with 25 hidden units using sigmoidal activation function) in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Auto-MPG case.

Figure 4.7: The learning evolution patterns of **OS-ELM** (with 25 hidden units using sigmoidal activation function) in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Abalone case.

Figure 4.8: The learning evolution patterns of **OS-ELM** (with 50 hidden units using sigmoidal activation function) in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the California Housing case.

4.4.3 OS-ELM Performance Evaluation on Classification Problems

Another application of **OS-ELM** is multiple-category classification. Such application can be viewed as vector function approximation using SLFNs in Figure 3.1, where each output node stands for one class. For this study, three benchmark problems of classification have been considered, viz., Image Segmentation, Satellite Image, and DNA [7].

The Image Segmentation problem consists of a database of images drawn randomly from 7 outdoor images and consists of 2310 regions of 3 x 3 pixels. The aim is to recognize each region into one of the 7 categories, viz., brick facing, sky, foliage, cement, window, path and grass using 19 attributes extracted from each square region. A pair of attributes and their corresponding category label consists of one sample in the database. Training and testing samples are randomly drawn from the entire database.

The Satellite Image problem consists of a database generated from Landsat Multi-Spectral Scanner. One frame of Landsat Multi-Spectral Scanner imagery consists of four digital images of the same scene in four different spectral bands. The database is a (tiny) sub-area of a scene, consisting of 82 x 100 pixels. Each sample in the database corresponds to a region of 3x3 pixels. The aim is to classify of the central pixel in a region into 6 categories, viz., red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble and very damp grey soil using 36 spectral values (attributes) for each region. The training and test samples are fixed according to [7], but their order is randomly shuffled for each trial.

The “prediction of primate splice-junction gene sequences” is known as the DNA problem. Splice junctions are points on a DNA sequence at which ‘superfluous’ DNA is removed during the process of protein creation in higher organisms. The aim of the DNA problem is, given a sequence of DNA, to recognize the boundaries between exons (the parts of the DNA sequence retained after splicing) and introns

(the parts of the DNA sequence that are spliced out). This consists of 3 subtasks: recognizing exon/intron boundaries, intron/exon boundaries, and neither. A given sequence of DNA consists of 60 elements (called “ucleotides” or “base-pairs”). The symbolic variables representing nucleotides were replaced by 3 binary indicator variables, thus resulting in 180 binary attributes. The training and test samples are also fixed according to [7], but their order is randomly shuffled for each trial.

(i) One-by-One Learning Mode:

For all simulations on the classification problems, RAN and RANEFK produced too many hidden units since there is no pruning mechanism in these algorithms. Further they required much longer training time and frequently the system was out of memory. Hence they are not included in the comparison. It is also found that the input sampling distribution for some of these problems is difficult to estimate. Hence, for the Satellite Image and DNA problems, no simulation has been done with GAP-RBF, which requires the calculation of input sampling distribution. Here the results of RAN-based sequential learning algorithms on these classification applications are originally documented in [28] and represented here.

Table 4.4 summarizes the performances of **OS-ELM** and other algorithms in the learning mode of one-by-one with respect to the average training time, training and testing accuracy (correct classification rate, %), and number of hidden units. Again, the optimal number of hidden units of **OS-ELM** and SGBP are chosen from the the range estimated in model selection procedure. And the optimal learning parameters and control parameters are applied for SGBP and RAN-based algorithms as well. Additionally, the number of initial samples used in the initialization phase for **OS-ELM** is set equal to the optimal number of hidden units found during model selection. From the table, it can be see that **OS-ELM** (Sigmoid) and **OS-ELM** (Gauss) achieve similar training and testing accuracies. More importantly, their testing accuracies rank highest among all the algorithms. And their training time is much less than RAN-based algorithms. SGBP completes the training at the fastest speed; however its accuracies are much less than other algorithms.

Datasets	Algorithms	Time (seconds)	Accuracy (%)		# hidden units
			Training	Testing	
Image Segmentation	OS-ELM (Sigmoid)	9.1365	96.74	95.06	180
	OS-ELM (Gauss)	11.264	96.23	94.97	180
	SGBP	2.5776	83.71	82.55	80
	GAP-RBF	3291.3	-	88.42	55.4
	MRAN	7004.5	-	93.30	53.1
Satellite Image	OS-ELM (Sigmoid)	305.64	91.99	88.93	400
	OS-ELM (Gauss)	323.12	92.95	88.95	400
	SGBP	3.1415	85.23	83.75	25
	MRAN	2469.4	-	86.36	20.4
DNA	OS-ELM (Sigmoid)	17.315	96.93	94.21	200
	OS-ELM (Gauss)	24.068	95.82	92.24	200
	SGBP	1.0840	85.64	82.11	12
	MRAN	6079.0	-	86.85	5

Table 4.4: Performance comparison between **OS-ELM** and other algorithms in the learning mode of one-by-one on the classification problems. The number of hidden units MRAN and GAP-RBF is the average over 50 trials.

Figure 4.9, 4.10, 4.11 show the learning evolution patterns of **OS-ELM** (Sigmoid) and SGBP on a single trial in the cases of Image Segment, Satellite Image, and DNA, respectively. In the figures, the training and testing accuracies is represented as functions of the number of training samples. The dotted curves illustrate the training accuracies, which are estimated over the training samples learned so far in each iteration. The solid curves show the testing accuracies, which are calculated over the entire testing samples.

As observed from Figure 4.9, the learning evolution for **OS-ELM** (Sigmoid) in the Image Segment case starts after the first 180 training samples have been presented to the algorithm to complete the initialization phase. At this moment, the training accuracy of the algorithm is 100%, while testing accuracy is relatively low. Afterwards the update phase of the algorithm commences: the training accuracy decreases slightly as more training samples present, whereas testing accuracy increases fast at the beginning of update phase. And subsequently training and testing accuracies converge smoothly. Around after 1000 training samples have been learned, no obvious changes could be observed with the training and testing

Figure 4.9: The learning evolution patterns for **OS-ELM** (with 180 hidden units using sigmoidal activation function) and for SGBP (with 80 hidden units) in the learning mode of one-by-one on a single trial in the Image Segment case.

accuracies. On the contrary, the learning evolution for SGBP starts at the presentation of the first training sample. The training and testing evolution patterns of the algorithm are alike. From the figure, we observe that the testing accuracy convergence speed of SGBP is slower than **OS-ELM**. And its final testing accuracy is lower than **OS-ELM** as well. Moreover, the learning evolution pattern of **OS-ELM** is much clearer than that of SGBP. Similar observations can be found in Figure 4.10 and 4.11, which correspond to Satellite Image and DNA classification problem.

Figure 4.10: The learning evolution patterns for **OS-ELM** (with 400 hidden units using sigmoidal activation function) and for SGBP (with 25 hidden units) in the learning mode of one-by-one on a single trial in the Satellite Image case.

Figure 4.11: The learning evolution patterns for **OS-ELM** (with 200 hidden units using sigmoidal activation function) and for SGBP (with 12 hidden units) in the learning mode of one-by-one on a single trial in the DNA case.

(ii) **Chunk-by-Chunk Learning Mode:**

Datasets	Activation Functions	Algorithms	Learning Mode	Time (seconds)	Accuracy (%)		# hidden units
					Training	Testing	
Image Segmentation	Sigmoid	ELM	Batch	0.6384	96.75	95.07	180
		OS-ELM	1-by-1	9.1365	96.74	95.06	180
			20-by-20	0.8303	96.72	95.02	180
			[10,30]	0.8228	96.72	95.10	180
	Gauss	ELM	Batch	1.6300	96.22	94.91	180
		OS-ELM	1-by-1	11.2642	96.23	94.97	180
			20-by-20	1.1012	96.17	95.02	180
			[10,30]	1.2216	96.22	94.95	180
Satellite Image	Sigmoid	ELM	Batch	7.1816	91.95	88.97	400
		OS-ELM	1-by-1	305.6416	91.99	88.93	400
			20-by-20	18.6037	91.98	88.91	400
			[10,30]	18.6825	91.97	88.92	400
	Gauss	ELM	Batch	24.8091	92.94	89.03	400
		OS-ELM	1-by-1	323.1163	92.95	88.95	400
			20-by-20	22.1444	92.92	89.02	400
			[10,30]	24.4513	92.93	89.08	400
DNA	Sigmoid	ELM	Batch	0.9748	96.90	94.30	200
		OS-ELM	1-by-1	17.3148	96.93	94.21	200
			20-by-20	1.4362	96.84	94.30	200
			[10,30]	1.4806	96.88	94.25	200
	Gauss	ELM	Batch	8.2998	95.87	92.33	200
		OS-ELM	1-by-1	24.0682	95.82	92.24	200
			20-by-20	3.6288	95.79	92.31	200
			[10,30]	3.8557	95.72	92.29	200

Table 4.5: Performance comparison between batch **ELM** and **OS-ELM** in the learning mode of chunk-by-chunk on the classification problems. [10,30] means that the number of samples is randomly picked in the range of 10 to 30 for each iteration.

The performance of **OS-ELM** in the learning mode of chunk-by-chunk is described here. We do not conduct the comparison of **OS-ELM** with SGBP and RAN-based algorithms as they can not operate in the the learning mode of chunk-by-chunk. Table 4.5 summarized the performance of **OS-ELM** for both sigmoidal and gaussian activation functions in various learning modes on the same classification problems. For comparison purpose, we provide the performance of batch **ELM** in the table as well. The average training time, training and testing accuracies (correct classification rate, %), and number of hidden units are considered.

The simulation results show that the average training and testing accuracies achieved by batch **ELM** and **OS-ELM** in various learning modes are similar. The tiny difference among them is probably due to the different initial values for the hidden unit parameters, which are randomly chosen for each learning mode and each trial from a uniform distribution in the range $[-1,1]$.

From the table, we also observe that the training time taken by **OS-ELM** reduces as the chunk size increases. For instance, in the Satellite Image classification problem, the training time reduces from 300 (using one-by-one learning mode) to 20 seconds (using 20-by-20 learning mode). However, we also find the exception examples that the batch mode takes more time than the sequential learning mode of 20-by-20 and $[10,30]$ when gaussian activation function is used for these classification problems. This may be due to the same reason as discussed in regression application in Section 4.4.2.

Figure 4.12 shows the learning evolution patterns of **OS-ELM** in various learning modes (viz., one-by-one, 20-by-20, and $[10,30]$) on a single trial in the Image Segment case. In the figure, y-axis is confined to the range $[70\%, 100\%]$ in order to focus on the details of update phase of the algorithm. And the training and testing accuracies are presented as the functions of the number of training samples. The thin dotted and solid curves present the training and testing accuracies of **OS-ELM** with sigmoidal activation function in the learning mode of one-by-one. The thick dotted and solid curves are for **OS-ELM** with sigmoid activation function in the learning mode of 20-by-20. And the thin and thick dash-dotted curves correspond to **OS-ELM** in the learning mode of $[10,30]$. For comparison purpose, the training and testing accuracies produced by batch **ELM** (on a single trial) are also presented in the figure. They are marked by the symbol of asterisks in small and large size in the end of the learning evolution of **OS-ELM**.

In the initialization phase of the algorithm, the number of initial training samples is set equal to the number of hidden units estimated through model selection. As shown in the figure, the learning evolution of **OS-ELM** (Sigmoid) starts after

the first 180 training samples have been used to complete the initialization phase of **OS-ELM**. At this moment, the training accuracies for various learning modes are 100%, while testing accuracies are relatively small. In the update phase, the changes in the training and testing accuracy curves for all the learning modes are alike: training accuracies decrease slightly as more training samples present, whereas testing accuracies increase fast at the beginning of update phase. Subsequently training and testing accuracies converge smoothly. And at the end of the learning evolution of the algorithm, it appears that the training and testing accuracies for various learning modes are identical with those achieved by batch **ELM**. The differences among their learning evolution patterns are caused by two facts: the order of the training samples presented as well as the initial values for the hidden unit parameters. And the differences of the final training and testing accuracies between **OS-ELM** and batch **ELM** are caused by the initial values for the hidden unit parameters only. Similar behaviors have been observed in Figure 4.13 and Figure 4.14, which correspond to the Satellite Image and DNA problem.

Figure 4.12: The learning evolution patterns for **OS-ELM** with 180 hidden units using sigmoidal activation function in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Image Segment case.

Figure 4.13: The learning evolution patterns for **OS-ELM** with 400 hidden units using sigmoidal activation function in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Satellite Image case.

Figure 4.14: The learning evolution patterns for **OS-ELM** with 200 hidden units using sigmoidal activation function in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the DNA case.

4.4.4 OS-ELM Performance Evaluation on Mackey-Glass Time Series Prediction Problem

Another important application of **OS-ELM** is time series prediction, which is a special case of function approximation. The need for time series prediction arises in many real-world problems such as detecting arrhythmia in heartbeats, stock market indices etc. One of the classical benchmark problems in literature is Mackey-Glass time series prediction. The time series is defined by Mackey-Glass differential delay equation and expressed as [40]:

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x^{10}(t-\tau)} - bx(t) \quad (4.65)$$

Integrating the equation over the time interval $[t, t + \Delta t]$ by the trapezoidal rule yields:

$$x(t + \Delta t) = \frac{2 - b\Delta t}{2 + \Delta t}x(t) + \frac{a\Delta t}{2 + b\Delta t} \left[\frac{x(t + \Delta t - \tau)}{1 + x^{10}(t + \Delta t - \tau)} + \frac{x(t - \tau)}{1 + x^{10}(t - \tau)} \right] \quad (4.66)$$

In our simulation, the time series is generated under the condition $x(t - \tau) = 0.3$ for $0 \leq t \leq \tau$. Moreover, we set $a = 0.2, b = 0.1, \tau = 17$, and $\Delta t = 1$. The problem is to predict with $v = 50$ steps ahead using the four past samples: $s_{n-v}, s_{n-v-6}, s_{n-v-12}$, and s_{n-v-18} . Hence the n^{th} input-output sample of the problem is given by:

$$\begin{aligned} \mathbf{x}_n &= [s_{n-v}, s_{n-v-6}, s_{n-v-12}, s_{n-v-18}]^T \\ y_n &= s_n \end{aligned}$$

The training data is from $n = 1$ to 4000 and testing from $n = 4001$ to 4500. Figure 4.15 depicts the first 1000 training targets on the generated Mackey-Glass time series. From the figure, we observe that the samples around the “peaks” in the time series are densely spaced, which may cause the initial hidden layer output

Figure 4.15: The first 1000 training targets on the generated Mackey-Glass time series.

matrix ill-conditioned. In order to solve this problem, we slightly increase the number of initial samples used in the initialization phase in our simulation with **OS-ELM**.

(i) **One-by-One Learning Mode:**

Algorithms	Time (seconds)	Training RMSE	Testing RMSE	# hidden units
OS-ELM (Sigmoid)	9.9963	0.0229	0.0231	120
OS-ELM (Gauss)	14.2517	0.0205	0.0196	120
GGAP-RBF	24.326	0.0700	0.0368	13
MRAN	57.205	0.1101	0.0337	16
RANEKF	62.674	0.0726	0.0240	23
RAN	58.127	0.1006	0.0466	39

Table 4.6: Performance comparison between **OS-ELM** and other algorithms in the learning mode of one-by-one on the Mackey-Glass time series prediction problem.

Performance comparison of **OS-ELM** and other algorithms in the learning mode of one-by-one on this problem is summarized in Table 4.6 with respect to the average training time, training and testing RMSEs, and number of hidden units.

SGBP is excluded from this simulation as we found that its validation RMSE was high even after trying out a large number of hidden units in the procedure of model selection. This indicates that SGBP is not suitable for this application. Hence no result for SGBP is reported here. For **OS-ELM**, the number of initial samples is chosen slightly larger than the number of hidden units. It is due to the observation that the training samples around the “peaks” in the generated Mackey-Glass time series are densely spaced. In our numerical experiment with this problem, we set the number of initial samples equal to 125 samples. From the table, it shows **OS-ELM** (Gauss) achieves the lowest testing RMSE followed by **OS-ELM** (Sigmoid) and RANEKF. The training time of both **OS-ELM**(Sigmoid) and **OS-ELM**(Gauss) is lowest.

Figure 4.16 shows the learning evolution patterns of both **OS-ELM** (Sigmoid) and **OS-ELM**(Gauss) in the learning mode of one-by-one on a single trial in this problem. In the figure, y-axis uses logarithmic scale for the plot. The thin dotted and solid curves are training and testing RMSEs of **OS-ELM** (Sigmoid), which are presented as functions of the number of training samples. And the thick dotted and solid curves correspond to **OS-ELM** (Gauss). From the figure, we observe the learning evolution patterns of both **OS-ELM** (Sigmoid) and **OS-ELM**(Gauss) are similar. The difference among them is due to the different initial values for the hidden unit parameters, which are randomly chosen from a uniform distribution in the range $[-1,1]$ for each trial. We also observe that at the moment of the initialization phase, the training RMSEs of both **OS-ELM** (Sigmoid) and **OS-ELM** (Gauss) are some small positive values. It is due to the number of initial training samples is more than the number of hidden units.

Figure 4.17 illustrates the test targets (circles) on the generated time series and the predicted values (dots) using **OS-ELM** (Sigmoid) in the mode of one-by-one. From the figure, it can be seen that the predicted ones are much close to the real ones.

Figure 4.16: The learning evolution patterns for **OS-ELM** with 120 hidden units using sigmoidal and gaussian activation functions in the learning mode of one-by-one on a single trial in the Mackey-Glass time series prediction problem.

Figure 4.17: The testing targets (circles) on the generated Mackey-Glass time series and predicted values (dots) by **OS-ELM** with 120 hidden units using the sigmoidal function in the learning mode of one-by-one.

(ii) **Chunk-by-Chunk Learning Mode:**

Activation Functions	Algorithms	Learning Mode	Time (seconds)	RMSE		# hidden units
				Training	Testing	
Sigmoid	ELM	Batch	1.1664	0.0183	0.0187	120
	OS-ELM	1-by-1	9.9963	0.0229	0.0231	120
		20-by-20	0.8612	0.0226	0.0228	120
		[10,30]	0.9613	0.0229	0.0231	120
Gauss	ELM	Batch	2.1794	0.0185	0.0180	120
	OS-ELM	1-by-1	14.2517	0.0205	0.0196	120
		20-by-20	1.2541	0.0205	0.0196	120
		[10,30]	1.5372	0.0205	0.0196	120

Table 4.7: Performance comparison between batch **ELM** and **OS-ELM** in the learning mode of chunk-by-chunk on the Mackey-Glass time series prediction problem. [10,30] means that the number of samples is randomly picked in the range of 10 to 30 for each iteration.

Again the performance of **OS-ELM** in the learning mode of chunk-by-chunk is described here. We do not conduct the comparison of **OS-ELM** with RAN-based algorithms as they do not operate in the the learning mode of chunk-by-chunk. Table 4.7 summarized the performance of **OS-ELM** using both sigmoidal and gaussian activation functions in various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on this problem. For comparison purpose, we provide the performance of batch **ELM** in the table as well. The average training time, training and testing RMSEs, and number of hidden units are considered.

As observed from the table, the testing RMSEs achieved by **OS-ELM** in various learning modes are close but slightly higher than that of batch **ELM**. It may be due to the matrix $\mathbf{H}_{n_0}$ approaching singularity in some trials. From the table, we also observe that the batch mode takes more training time than the learning mode of 20-by-20 and [10,30], which may be due to the similar reason as discussed in California Housing regression problem in Section 4.4.2.

Figure 4.18 shows the learning evolution patterns of **OS-ELM** (Sigmoid) in various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in this time series prediction problem. In the figure, y-axis uses logarithmic scale and is

confined to the range $[10^{-3}, 10^0]$ to focus on the details of the learning evolution. The training and testing RMSEs are presented as the functions of the number of training samples. The thin dotted and solid curves present the training and testing RMSEs of **OS-ELM** with sigmoidal activation function in the learning mode of one-by-one. The thick dotted and solid curves are for **OS-ELM** in the learning mode of 20-by-20. And the thin and thick dash-dotted curves correspond to **OS-ELM** in the learning mode of [10,30]. For the comparison purpose, the training and testing RMSEs produced by batch **ELM** (on a single trial) are also shown in the figure. They are marked by the symbol of asterisks in the small and large size in the end of the learning evolution of **OS-ELM**.

From the figure, we observe that the learning evolution patterns of **OS-ELM** (Sigmoid) in various learning modes are similar: at the moment of the initialization phase, the training RMSEs are small positive values, while testing RMSEs are relatively larger. Further, at the beginning of the update phase, testing RMSEs converges fast, and there is no obvious change observed with them after 1000 samples have been learned. From the figure, it shows that the final RMSEs of both batch **ELM** and **OS-ELM** are close to each other (for the initial hidden layer output matrix $\mathbf{H}_{n_0}$ of **OS-ELM** is well-conditioned in this trial). The differences among their learning evolution patterns are probably caused by the initial values for the hidden unit parameters only.

4.5 Summary

In this chapter, **OS-ELM** has been derived from batch **ELM** for approximate mapping. From a theoretical viewpoint, **OS-ELM** inherits all the characteristics of batch **ELM**, viz., unified framework for ridge and radial basis activation function in a SLFN, reduced computational complexity (no adjustment involves with the hidden unit parameters throughout the learning evolution), and improved generalization performance (the norm of the solution for the output unit parameters is

Figure 4.18: The learning evolution for **OS-ELM** with 120 hidden units using sigmoidal activation function in the various learning modes (viz., one-by-one, 20-by-20, and [10,30]) on a single trial in the Mackey-Glass time series prediction problem.

minimized with least-squares training error). Further we have theoretically proven that the input-output mapping functions learned by **OS-ELM** and batch **ELM** are identical given the same training samples and initial model of a SLFN.

In the beginning of this chapter, we derive **OS-ELM** in the learning mode of one-by-one, which is commonly adopted by other sequential learning algorithms, e.g., stochastic BP-based and RAN-based algorithms. Further, we introduce the learning mode of chunk-by-chunk to **OS-ELM**. The algorithm can handle chunk size that may vary in each update iteration. **OS-ELM** in chunk-by-chunk learning mode iteratively updates the parameters of the SLFN on the basis of the chunk of current samples. In theory, we have proven that batch **ELM** and **OS-ELM** in the learning mode of one-by-one are two special cases of **OS-ELM** in the learning mode of chunk-by-chunk. Other sequential algorithms may not operate in the learning mode of chunk-by-chunk. Hence **OS-ELM** offers greater flexibility of learning mode than other sequential learning algorithms.

In the later part of this chapter, numerical experiments have been performed: **OS-ELM** is simulated for both sigmoidal and gaussian activation functions. Their performances are compared with other sequential algorithms in the learning mode of one-by-one in the areas of regression, classification, and chaotic time series prediction applications. The simulation results show that the proposed **OS-ELM** achieves better generalization performance (lowest testing RMSE for regression and chaotic time series prediction applications and highest testing accuracy for classification application) with much less training time than other sequential learning algorithms for all the benchmark problems.

Further the performance of **OS-ELM** in the learning mode of chunk-by-chunk on the same problems has been evaluated and compared with batch **ELM**. The chunk size simulated here includes one sample, 20 samples, and the number of samples randomly picked from the range [10,30]. The simulation results show that the generalization performance of both batch **ELM** and **OS-ELM** in various learning modes are similar, which coincides with the theoretical study. Considering

the training time, we observe that **OS-ELM** in the learning mode of one-by-one takes the highest training time and batch **ELM** the shortest training time. Other chunk size falls in between these two cases. However there are exceptions where **OS-ELM** in the learning modes of 20-by-20 and [10,30] is even faster than batch **ELM**. It maybe due to the huge dimension of the hidden layer output matrix in batch **ELM**.

In summary, we have in detail explored the properties of **OS-ELM** in the theoretical study. Further, from the simulation results, it illustrates that **OS-ELM** outperforms the other sequential learning algorithms in terms of training time, generalization performance, and flexible learning modes. Hence both from the theoretical and practical viewpoint, **OS-ELM** is efficient and suited for online applications.

Chapter 5

OS-ELM Application in BCI

A Brain Computer Interface (BCI) is devised to provide a nonmuscular communication channel for those with severe neuromuscular disorder. Its main component involves with mental task classification through the recorded brain activity [65, 63]. Recently, the need for online learning of the classifier in a BCI system has been recognized for the dynamical nature of brain activity [13]. Several important requirements of the online learning method used in a BCI system have been stressed: fast convergence speed (requiring minimal number of training samples), good generalization performance (producing high rate of correct respond of the system), and inexpensive computation (such that the system can adapt and respond fast enough for practical use). In Chapter 4, we have shown that **OS-ELM** is characterized by fast convergence speed, improved generalization performance, and reduced computation complexity when compared with other sequential learning algorithms. This motivates us to apply **OS-ELM** into the online classifier design in BCI systems. The outline of a BCI system is first presented. The simulation results of **OS-ELM** in BCI application is then provided.

Figure 5.1: The diagram shows the general scheme of a BCI System.

(Available at [http :](http://www.bci-info.tugraz.at/ResearchInfo/documents/articles/briefintro)

[//www.bci – info.tugraz.at/ResearchInfo/documents/articles/briefintro](http://www.bci-info.tugraz.at/ResearchInfo/documents/articles/briefintro))

5.1 BCI System Outline

As depicted in Figure 5.1, a BCI system is generally composed of components of signal acquisition, signal processing, and application interface. The idea behind the validation of BCIs is that different mental tasks (e.g., motor/cognitive mental tasks) cause changes in the brain activity [15, 48, 49, 43]. Through a certain kind of signal acquisition technique, the brain activity is monitored and recorded as the input of the system. The type of tasks is recognized using advanced signal processing techniques. In the application interface, the code of task type is converted into an appropriate signal for representing the intend of the subject, e.g., controlling or communicating with external object(s).

The popular **signal acquisition** techniques used in BCI are those for collecting electrical brain signals, e.g., EEG (electroencephalograph), ECoG (electrocortical graph), and that making use of intracortical electrodes. Since EEG is noninvasive and less expensive, it is widely used in practice [32]. And in the following discussion, we assume that EEG recording is used as the input of the system.

Next, advanced **signal processing** techniques are applied for exploiting the embedded information of input signals in order to recognize the type of tasks. Signal processing is the crucial component in a BCI system. As shown in Figure 5.1, it can be further divided into three subcomponents, viz., preprocessing, feature extraction, and classification/detection.

5.1.1 Preprocessing

Since the amplitude of electrical brain activity recorded on the scalp surface is tiny and normally in order of μV . On the other hand, the voltage of the noise/artifact in EEG signal is relatively large. For instance, eye blinks can produce voltage changes of much higher amplitude (over $100\mu V$ s) than the endogenous brain activity (generally in the range of -50 to $50\mu V$ s). In such situation, the signal-to-noise ratio (SNR) of the signals is much low, and thus the rate of correct classification of the

tasks may decrease. Therefore the signals contaminated by the noise should be discarded/rejected; otherwise we can remove/correct the noise through advanced signal processing techniques, such as those based on regression models [8, 64, 59], principal component analysis (PCA) [36], and independent component analysis (ICA) [10, 33].

5.1.2 Feature Extraction

It is difficult to classify the mental tasks directly based on the recorded signals. Commonly certain kinds of discriminable features are extracted prior to the mental task classification.

The frequency rather than the amplitude of the brain waves is widely used for interpreting the ongoing EEG signals. In Figure 5.2, four categories of brain waves in the typical frequency bands of Beta, Alpha, Theta, and Delta are presented. Hence brain oscillation patterns (i.e., frequency features) are commonly extracted and used for the mental task classification in BCI domains. Table 5.1 summarizes the event-related patterns of brain oscillations in the main frequency bands of EEG signals. From the table, it can be seen that the brain oscillation patterns (including the specific frequency band and distribution) is associated with the subject's state as well as the task being performed.

The patterns of brain oscillations are estimated in frequency domain, such as the power spectral density (PSD) of the recorded signals. Various classical signal processing techniques may be applied for this purpose, e.g., fast Fourier transform (FFT) and autoregressive (AR) spectral estimation. Sometimes, the coefficients of an AR model are directly used as the feature input for the mental task classification in a BCI system.

Brain wave (frequency)	Distribution	Subjective states	Associated tasks & behaviors
Delta (0.1-3Hz)	widespread	deep sleep, trance, unconscious	lethargic, not moving, not attentive
Theta (3.5-7.5Hz)	regional	creative, recall, fantasy, imagery, dreamlike thoughts; drowsy	creative but may also be distracted
Alpha (7.5-13Hz)	regional	tranquil, conscious	meditation, no action
Low Beta (12-15Hz)	localized by side and by lobe	relaxed yet focused	lack of focused attention
Midrange Beta (15-18Hz)	localized by one electrode	thinking, aware	mental activity
High Beta (above 18Hz)	localized (may be very focused)	alertness, agitation	mental activity, e.g. math, planning, etc.
Gamma (above 36Hz)	very localized	thinking	high-level information processing

Table 5.1: The patterns of brain waves are related to the mental states and tasks

Figure 5.2: A diagram shows four categories of EEG waves in the typical frequency bands: Beta, Alpha, Theta, and Delta.

5.1.3 Classification/Detection

The classifier of mental tasks plays an important role within a BCI system, which is responsible for predicting the type of mental tasks after properly training. Hence the training method for the classifier is a crucial algorithm of the system.

Recently, a number of linear and nonlinear classifiers in the application of BCIs have been studied [42]. The linear classifiers include those based on statistical disciplines, such as Linear Discriminant Analysis (LDA), Hidden Markov Classifier, and Z-scale base Discriminant Analysis (ZDA) [19]. The nonlinear classifiers include BP-based neural networks and Support Vector Machines (SVMs). It has been shown the nonlinear methods can achieve improved classification performance compared with linear techniques in BCIs [16].

However, these nonlinear classifiers require training off-line. Because of the dynamic nature of the brain activity, the need for online learning of the classifier in BCIs has been recognized [13]. Further the online classifier in BCI application should be characterized by fast convergence speed, good generalization performance, and inexpensive computation such that the system can adapt quickly and respond accurately for practical applications.

In Chapter 4, the study has shown that **OS-ELM** can achieve improved generalization performance with faster convergence speed and less computation complexity compared with other sequential learning algorithms. This motivates us to apply **OS-ELM** into the online classifier design in BCIs and carry out simulations of **OS-ELM** in the application of cognitive mental task classification in the following section.

After properly training online, the classifier can be used for predicting the type of tasks based on the unseen feature input. Then, in the **application interface**, the classifier's output is transformed into an appropriate signal, which is used for representing the intend of the subject, such as, controlling or communicating with certain external object(s).

5.2 Performance Evaluation

Motivated by the need for the online learning of the classifier in BCI domain, we propose the application of **OS-ELM** into the online classifier design. Here simulations of the algorithm are carried out based on a well-known EEG database of cognitive mental task classification. The performance of **OS-ELM** is evaluated and compared with other sequential learning algorithms session by session.

5.2.1 EEG Database

A well-known EEG database is used for the purpose of performance evaluation and comparison. This EEG database was recorded by Zak Keirn [35] and distributed by Anderson [1]. The recordings (C3, C4, P3, P4, O1, and O2) were made with reference to mastoids (A1 and A2). Additionally EOG was recorded as the difference between the forehead electrode above the left brow line and another one on the left cheekbone. All recordings were sampled at 250Hz. There were 7 healthy subjects and 13 sessions in the database for each subject performed one to three sessions of recordings. And in each session, five 10-second trials of five pre-defined cognitive mental tasks (viz., Relax, Letter Composition, Multiplication, Counting, and Rotation) were executed.

5.2.2 Preprocessing & Feature Extraction

To extract the useful information for the cognitive mental task classification, we first detect and discard the signals contaminated by ocular artifact. Then a 6-order AR model is fitted into the resulted signals. The coefficients of the model are estimated using Burg method. In Figure 5.3, a flowchart shows the steps involved with the procedure for ocular artifact removal and feature extraction for each trial of a task.

In the flowchart, it is shown that the EEG signals in each trial of a task are first

Figure 5.3: A flowchart shows the procedure for ocular artifact removal and feature extraction from each trial of a task.

normalized with zero mean and variance one, and then divided into 1/2-second segments with an overlap of 1/4-second between adjacent segments, which results in 39 segments. The segments contaminated by ocular artifacts are detected and discarded using the scheme in [8] through EOG recording. The scheme is based on a fixed-weight leakage normalized stochastic least mean fourth algorithm in a predictor mode. The detector recognizes abrupt patterns in EOG recording caused by eye movements or blinks. Its output is composed of 1s and 0s, with possible 0s between 1s. Then morphological operations are used for smoothing the output of the detector, which results in two separated sets of EOG signals: eye-movement and non-eye-movement. We discard the EEG segments when an eye-movement set of EOG signals is identified. Finally, there are m segments of EEG signals produced from each trial of a task.

To proceed further, certain discriminable features are extracted from the resulted segments in frequency domain. For this purpose, autoregressive (AR) modelling technique [45] is used. The order of the model is six, the same as that in [16, 2] for cognitive mental task classification using the same database. As shown in the flowchart, the 6-order AR model is used to fit into each channel of EEG signals for each segment. And the coefficients of the model are estimated by Burg method. The resulted coefficient vectors from six channels are then concatenated into a feature vector of 36-dimension (6 channels * 6 coefficients) for each segment [16, 2]. Finally there are m feature vectors of 36-dimension generated from each trial of a task.

One feature vector and its corresponding task label compose of a data sample. The samples generated from the first 4 trials of tasks compose of training data set; while those from the last trial compose of testing data set. Training and testing data set are then used in the following performance evaluation and comparison of **OS-ELM** with other sequential learning algorithms.

5.2.3 Classification

OS-ELM is applied to construct a SLFN online for the cognitive mental task classification. For comparison purpose, simulation with other sequential learning algorithms is supposed to carry out. However, SGBP was found unsuitable for this application due to the same reason as discussed in the case of Mackey-Glass time series prediction: its validation accuracy was low after trying out a large number of hidden units. For RAN-based algorithms of Chapter 4, very long training time was required and frequently the system was out of memory. It is possibly caused by the high dimension of the input space as well as the complexity of this application. Therefore, we use an improved RAN-based algorithm, which is developed recently and called FGAP-RBF[68]. FGAP-RBF differs from GAP-RBF of Chapter 4 by the way the update process is carried out and generally results in reduced computational complexity. Hence in our simulations, the performance of **OS-ELM** (both using sigmoidal and gaussian activation function) is evaluated and compared with FGAP-RBF session by session (5 trials * 5 tasks). Further the learning mode used by **OS-ELM** is one-by-one in that the samples of this application are available one at a time.

Model Selection

As analyzed in Section 4.4.1, the number of hidden units in **OS-ELM** is problem specific and should be selected using model selection session by session. On the other hand, FGAP-RBF involves similar control parameters as GAP-RBF. Therefore the selection of the optimal combination of control parameters is also required for FGAP-RBF.

Here we instead focus on the selection of the optimal number of hidden units for **OS-ELM**. Training and validation set method is applied for each session. First, the training data set is divided into two non-overlapped subsets: one for training is composed of the samples generated from the first 3 trials of tasks; and the other for validation is from the 4th trial. Then on the basis of these two subsets, the

training and validation accuracies (correct classification rate, %) are estimated and averaged over 50 independent trials. In each trial, the initial values for the hidden unit parameters of the SLFN are randomly chosen from a uniform distribution in the range $[-1, 1]$. Then the optimal number of hidden units is selected as the one which results in the highest average validation accuracy over trials.

Figure 5.4 provides an example of model selection for **OS-ELM** both using sigmoidal and gaussian activation function in session 1. In the figure, the average accuracies are shown as functions of the number of hidden units. The thin dotted and solid curves represent the average training and validation accuracies, respectively, for **OS-ELM** (Sigmoid); the thick dotted and solid curves are for **OS-ELM** (Gauss). As observed from the average validation curves, the highest average validation accuracies are achieved when the number of hidden units are in the ranges $[32, 48]$ and $[40, 65]$ for **OS-ELM** (Sigmoid) and **OS-ELM** (Gauss), respectively. Therefore the estimated optimal number of hidden units for **OS-ELM** (Sigmoid) and **OS-ELM** (Gauss) fall into these ranges in session 1.

The above procedure is repeated to search the optimal number of hidden units for **OS-ELM** (Sigmoid) and **OS-ELM** (Gauss) in all sessions. For FGAP-RBF, the optimal combination of control parameters is also estimated for each session using the same train and validation set method. All the estimated model parameters are applied in the following performance evaluation and comparison of **OS-ELM** with FGAP-RBF.

Table 5.2 summarizes, session by session, the performance of **OS-ELM** and FGAP-RBF in terms of training time per sample (milliseconds/sample), training and testing accuracy (correct classification rate, %), and the number of hidden units. The last column gives the number of training samples generated from EEG signals in each session. Session 6 and Session 7 are excluded from our simulations because of saturation or invalidation of the EEG signals in several trials of tasks in these two sessions.

The number of hidden unit for **OS-ELM** is chosen from the range estimated by

Figure 5.4: An example of model selection for **OS-ELM**(Sigmoid) and **OS-ELM** (Gauss) in session 1: Average accuracies are showed as a function of the number of hidden units.

Session No.	Algorithms	Time (ms/sample)	Accuracy (%)		# hidden units	# training samples
			Training	Testing		
1	OS-ELM (Sigmoid)	0.2	76.21	71.26	40	625
	OS-ELM (Gauss)	0.8	76.78	69.68	50	
	FGAP-RBF	24.4	97.12	61.54	619	
2	OS-ELM (Sigmoid)	0.2	79.62	69.66	40	674
	OS-ELM (Gauss)	0.8	78.68	67.62	45	
	FGAP-RBF	26.0	94.96	49.08	669	
3	OS-ELM (Sigmoid)	0.7	81.31	60.75	70	606
	OS-ELM (Gauss)	1.3	80.28	56.12	70	
	FGAP-RBF	24.9	88.94	58.33	591	
4	OS-ELM (Sigmoid)	0.3	69.47	18.10	45	661
	OS-ELM (Gauss)	1.0	71.52	19.77	60	
	FGAP-RBF	27.5	95.61	19.11	637	
5	OS-ELM (Sigmoid)	0.9	89.03	47.08	80	691
	OS-ELM (Gauss)	1.7	88.15	47.34	80	
	FGAP-RBF	27.3	92.62	42.77	679	
8	OS-ELM (Sigmoid)	0.2	78.61	29.12	40	634
	OS-ELM (Gauss)	0.7	77.23	28.17	45	
	FGAP-RBF	24.8	95.74	30.00	627	
9	OS-ELM (Sigmoid)	0.1	63.48	50.10	30	635
	OS-ELM (Gauss)	0.3	57.11	47.72	20	
	FGAP-RBF	25.1	90.55	45.22	632	
10	OS-ELM (Sigmoid)	0.2	73.81	36.25	40	645
	OS-ELM (Gauss)	1.7	79.00	40.16	80	
	FGAP-RBF	24.7	96.59	34.18	638	
11	OS-ELM (Sigmoid)	0.2	73.18	68.57	40	602
	OS-ELM (Gauss)	0.7	72.39	65.51	45	
	FGAP-RBF	23.4	95.85	55.33	592	
12	OS-ELM (Sigmoid)	0.3	75.89	48.70	45	660
	OS-ELM (Gauss)	0.7	73.23	45.94	45	
	FGAP-RBF	25.4	91.21	34.55	654	
13	OS-ELM (Sigmoid)	0.4	81.65	47.34	50	602
	OS-ELM (Gauss)	0.8	79.70	44.36	50	
	FGAP-RBF	23.1	98.17	45.81	594	

Table 5.2: Performance comparison between **OS-ELM** and FGP-RBF on the cognitive mental task classification.

model selection. And the training time per sample, training and testing accuracies for **OS-ELM** are estimated and averaged over 50 trials. In each trial, the initial values for the hidden unit parameters are randomly drawn from a uniform distribution in the range $[-1,1]$. For FGAP-RBF, the optimal combination of control parameters is applied. However, only one trial of simulation is carried out for FGAP-RBF since there is no stochastic factor involving with the use of FGAP-RBF. And the number of hidden units of FGAP-RBF is learned by the algorithm itself.

From the table, it can be observed that **OS-ELM** both using sigmoidal and gaussian activation function achieve similar training and testing accuracy. However, **OS-ELM** (Gauss) takes more training time and more hidden units than **OS-ELM** (Sigmoid) in most sessions. Further, considering **OS-ELM** and FGAP-RBF together, it is found that both algorithms can learn in real time in that the training time per sample taken by both algorithms is less than 250 milliseconds (each sample is generated using 1/2-second raw EEG signals with 1/4-second overlapping). Moreover **OS-ELM** spends less training time than FGAP-RBF in order of 10 to 100 times on average. Considering the number of hidden units learned by FGAP-RBF, it is much close to the number of training samples in each session, which might account for the higher training accuracy of FGAP-RBF compared with **OS-ELM**. On the other hand, testing accuracy is an important index of the performance of the algorithm. The simulation results show that **OS-ELM** obtains higher testing accuracies than FGAP-RBF in all sessions but except session 8. So far the simulation results indicate that **OS-ELM** outperforms FGAP-RBF in terms of the training time and testing accuracy.

Further the online learning nature of the algorithms is characterized by their learning evolution on a single trial. Figure 5.5 to 5.15 depict the learning evolution patterns for **OS-ELM** and FGAP-RBF in all sessions. In the figures, the classification accuracies are shown as a function of the number of training samples. The thin dot and solid curves represent the training accuracy and testing accuracies,

respectively, for **OS-ELM** (Sigmoid). The thick dot and solid curves correspond to **OS-ELM** (Gauss). And the thin and thick dash-dot curves are for FGAP-RBF.

In Figure 5.5, the learning evolution patterns of **OS-ELM** and FGAP-RBF is given for session 1. It is observed that the learning evolution of **OS-ELM** (Sigmoid) starts after the initialization phase is completed, in which a certain number of initial training samples are required. In our simulation, the number of training samples used in initialization phase of **OS-ELM** is set equal to the optimal number of hidden units as estimated in model selection for the algorithm for each session. At this moment, the training accuracy of **OS-ELM** (Sigmoid) is 100%, while the testing accuracy is relatively low. As more training samples have been learned by the algorithm, training accuracy decreases, whereas testing accuracy increases. For **OS-ELM** (Gauss), similar learning evolution pattern can be observed. Moreover, from previous experiments in Chapter 4, we may infer that the training and testing accuracies should converge to each other asymptotically, if more training samples are available and all the samples are consistent.

As far as FGAP-RBF is considered, its learning evolution comments at the presence of the first training sample. At this moment, its training and testing accuracies are similar with **OS-ELM**: training accuracy of 100% with low testing accuracy. And there is no change observed with the training and testing accuracies of the algorithm until a certain number of training samples have been learned. Subsequently, obvious increase trend of testing accuracy occurs, while the changes in training accuracy are still subtle. At the end of the learning evolution, we can observe that the final testing accuracy of FGAP-RBF is obviously lower than **OS-ELM** (both with sigmoidal and gaussian activation function) in this session.

The similar patterns can be observed in Figure 5.6 and 5.14 for session 2 and 12, respectively. For session 3,5,9,11 and 13, the alike behaviors are also observed except that the final testing accuracy achieved by the algorithms are comparable to each other.

However, in the session 4, 8 and 10, there is no obvious increase in the testing

accuracy of FGAP-RBF or **OS-ELM**. Even in session 4, decrease in the testing accuracy occurs after around 180 training samples have been learned by the algorithms. Such phenomenon is probably due to the subject. As we know the successful operation of a BCI system (i.e., the correct classification of the predefined mental tasks) is the result of the adequate adaptation between the subject and the system. Given the five predefined cognitive mental tasks, the subjects in these sessions may not produce the reliable/reproducible patterns of brain oscillation in that they may be not familiar with these tasks. In such case, one solution is to assign the subject the tasks that he/she knows well; otherwise some instructions should be provided such that he/she can perform the tasks in a correct way. Another solution is to introduce feedback into the system. Through feedback, the subject has the chance to learn to control certain characteristics of the brain patterns.

Figure 5.5: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session1.

Figure 5.6: A diagram shows the learning evolution for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session2.

Figure 5.7: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session3.

Figure 5.8: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session4.

Figure 5.9: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session5.

Figure 5.10: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session8.

Figure 5.11: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session9.

Figure 5.12: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session10.

Figure 5.13: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session11.

Figure 5.14: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session12.

Figure 5.15: A diagram shows the learning evolution patterns for **OS-ELM** (Sigmoid), **OS-ELM** (Gauss), and FGAP-RBF in the learning mode of one-by-one on a single trial of cognitive mental task classification in session13.

5.3 Summary

In this chapter, we propose the application of **OS-ELM** in the online classifier design in BCIs. Simulations are carried out based on a well-known EEG database of cognitive mental task classification. The performance of the algorithm is evaluated and then compared with FGAP-RBF session by session.

The results show that **OS-ELM** both using sigmoidal and gaussian activation function outperforms FGAP-RBF with better generalization performance and less computational complexity. Further, the online learning characteristic of the algorithms is illustrated by learning evolution on a single trial of simulation. From the learning evolution patterns, an increase trend in the testing accuracy can be found as more training samples have been learned. Moreover, **OS-ELM** achieves higher testing accuracy than FGAP-RBF at the end of the learning evolution. From these viewpoints, **OS-ELM** is a potential algorithm for the online classifier design in BCI domain.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

The core of the thesis has been laid on the derivation and implementation of **OS-ELM**, which can sequentially train both SLFNs with ridge function and SLFNs with radial basis function (i.e., RBF neural networks) in a unified framework. The algorithm employs the same strategy as batch **ELM** for reducing the computational complexity: the value for the hidden unit parameters of a SLFN is randomly assigned according to some probability density function (e.g., uniform distribution function) in the initialization phase of the algorithm, and does not require any adjustment in the update phase. On the other hand, **OS-ELM** differs from batch **ELM** by the way the estimation of the output unit parameters is performed: **OS-ELM** iteratively updates the output parameters of the SLFN on the basis of current sample(s) using RLS algorithm; while batch **ELM** estimates the output parameters based on the entire set of training samples in one step, which involves the calculation of the pseudoinverse of the hidden layer output matrix. Theoretically, **OS-ELM** both in the learning mode of one-by-one and chunk-by-chunk realizes identical mapping function as batch **ELM** given the same training samples and initial model of a SLFN. Thus **OS-ELM** inherits the excellent features from batch **ELM**, such as improved generalization performance, reduced compu-

tational complexity, and unified training framework for accommodating different activation functions used in the hidden layer of SLFNs. Further, the implementation of the algorithm both in learning mode of one-by-one and chunk-by-chunk is provided, and the related issues with the properties of the initial hidden layer output matrix are discussed.

Through seven benchmark problems in the areas of regression, classification, and time series prediction applications, the simulation results show that **OS-ELM** in the learning mode of one-by-one is superior than other sequential learning algorithms, i.e., SGBP, RAN, RANEKF, MRAN, GAP-RBF, and GGAP-RBF, in terms of training time and generalization performance. Moreover, the learning evolution patterns show that **OS-ELM** achieves fast convergence compared with SGBP in these applications. Other sequential algorithms may not operate in the learning mode of chunk-by-chunk, while **OS-ELM** can achieve comparable generalization performance with batch **ELM** on all the benchmark problems. Considering the training time between various learning modes of **OS-ELM** and batch **ELM**, it is observed that **OS-ELM** in one-by-one learning mode takes the longest training time and batch **ELM** the shortest training time. And various chunk sizes fall in between the one-by-one and batch case.

These advantages of **OS-ELM** lead to the application of the algorithm into the online classifier design in BCIs. The performance of **OS-ELM** is evaluated and compared with FGAP-RBF on a well-known EEG database consisting of the cognitive mental task classification. The simulation results show that **OS-ELM** achieves better classification accuracy with less computational complexity than FGAP-RBF in all sessions except session 8 in the database. From this viewpoint, **OS-ELM** is a potential algorithm for the online classifier design in BCI systems by providing a framework for the investigation of the optimal number of training samples as well as type of features and mental tasks.

6.2 Recommendations for Further Research

The main recommendation for further research is to expend **OS-ELM** such that it can automatically learn the optimal number of hidden units online. In the cases where the underlying system is nonlinear but stationary, the optimal number of hidden units could be determined in the initialization phase of the algorithm. For example, it could be estimated as the largest rank of initial hidden layer output matrix and then fixed in the update phase. However, for the nonlinear and non-stationary cases, it could be much more complex. We recommend the combination of RLS algorithm with adaptive exponential forgetting factor [17]. Within the adaptive memory span of the algorithm, the process can be considered as ‘semi-stationary’. Thus, the same trick as in the stationary cases could be applied for estimating the optimal number of hidden units in each update iteration. If the estimated number of hidden units is larger than the one estimated in previous iteration, then growing criterion should be introduced to initialize the parameters of the new hidden unit(s); otherwise, pruning criterion would be used for estimating the least significant unit(s). Finally, the output unit parameters should be adjusted in order to achieve fine adaptation of the process. This is an important direction of research and is likely to require substantial modification of the algorithm.

Like kernel learning techniques, the selection of optimal activation function is problem specific. For batch **ELM** and **OS-ELM**, the choice of a specific activation function is relevant to the property of the hidden layer output matrix. Further investigation of the relationship between the choice of the type of activation function and the property of hidden layer output matrix could be a key ingredient in the direction of activation function design in batch **ELM** and **OS-ELM**.

The valuable extension would be the application of **OS-ELM** into other BCI systems, such as those driven by ECoG and by intracortical electric recording. In these BCI systems, the input signal with better quality (higher SNR) can be recorded. Therefore more accurate respond has been achieved than traditional

EEG driven BCI systems. However, the input signal is very intensive and high dimensional, which results in the even more complicate online classifier design in these BCI systems. Finally, other real-life online applications are considerable interest for the validation of the theory of **OS-ELM**, as well as the discovery of hidden limitations of the algorithm.

Author's Publications

Journal Paper

- (1) Nan-Ying Liang, Paramasivan Saratchandran, Guang-Bin Huang, and Narasimhan Sundararajan, "Classification of Mental Tasks from EEG Signals using Extreme Learning Machine", *International Journal of Neural System*, Vol.16, No.1, pp.29-38, 2006.
- (2) Nan-Ying Liang, Guang-Bin Huang, Paramasivan Saratchandran, and Narasimhan Sundararajan, "A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks", (full paper, in press), *IEEE Transaction on Neural Networks*, vol.17, no.6, 2006.

Conference Paper

- (1) Nan-Ying Liang, Paramasivan Saratchandran, Guang-Bing Huang, and Narasimhan Sundararajan, "Cognitive Mental Task Classification using a Novel On-line Learning Algorithm", *International Conference on Cognitive and Neural System, 2005 (ICCNS05)*, Boston, USA, May 18-21, 2005.
- (2) Guang-Bin Huang, Nan-Ying Liang, Hai-Jun Rong, Paramasivan Saratchandran, and Narasimhan Sundararajan, "On-Line Sequential Extreme Learning Machine", *Proceeding of the IASTED International Conference on Computational Intelligence, 2005 (CI05)*, Calgary, Canada, July 4-6, 2005.

Bibliography

- [1] C. W. Anderson. Classification of electroencephalogram (EEG) signals for brain-machine interfaces. Computer Science Department, Colorado State University, USA, 2003. Available at <http://www.cs.colostate.edu/~anderson/res/eeg>.
- [2] C. W. Anderson and Z. Sijercic. Classification of EEG signals from four subjects during five mental tasks. In *Proceedings of the Conference on Engineering Applications in Neural Networks, 1996 (EANN96)*, 17-19 June, 1996.
- [3] V. Asirvadam, S. McLoone, and G. Irwin. Parallel and separable recursive Levenberg-Marquardt training algorithm. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 129–138, 4-6 Sep., 2002.
- [4] P. L. Bartlett. The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.
- [5] S. A. Barton. A matrix method for optimizing a neural network. *Neural Computation*, 3:450–459, 1991.
- [6] S. Bittanti, P. Bolzern, M. Campi, and E. Coletti. Deterministic convergence analysis of RLS estimators with different forgetting factors. In *Proceedings of the 27th IEEE Conference on Decision and Control, 1988*, volume 2, pages 1530–1531, Austin, Texas, 7 - 9 Dec. 1988.

-
- [7] C. Blake and C. Merz. UCI Repository of Machine Learning Databases. 1998.
Available at <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
 - [8] P. Celka, B. Boashash, and P. Colditz. Preprocessing and time-frequency analysis of newborn EEG seizures. *IEEE Engineering in Medicine and Biology*, 20:30–39, Sept./Oct. 2001.
 - [9] E. K. P. Chong and S. H. Żak. *An introduction to optimization*. New York: John Wiley, 2001.
 - [10] A. Cichocki and S. Vorobyov. Application of ICA for automatic noise and interference cancellation in multisensory biomedical signals. In *Proceedings of the Second International Workshop on ICA and BSS*, pages 621–626, June, 2000.
 - [11] G. V. Cybenko. Approximations by superpositions of sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.
 - [12] M. S. Dawson, A. K. Fung, and M. T. Manry. Surface parameter retrieval using fast learning neural networks. *Remote Sensing Reviews*, 7:1–18, 1993.
 - [13] J. del R. Millán. On the need for on-line learning in brain-computer interfaces. In *IEEE International Joint Conference on Neural Networks, 2004 (IJCNN2004)*, volume 4, pages 2877–2882, 25-29 July, 2004.
 - [14] S. Ferrari and R. F. Stengel. Smooth function approximation using neural networks. *IEEE Transactions on Neural Networks*, 16(1):24–38, 2005.
 - [15] R. S. J. Frackowiak. *Human brain function*. Elsevier Academic Press, 2004.
 - [16] D. Garrett, D. A. Peterson, C. W. Anderson, and M. H. Thaut. Comparison of linear, nonlinear, and feature selection methods for EEG signal classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11:141–144, June 2003.
 - [17] S. Haykin. *Adaptive filter theory, 4rd edition*. Prentice Hall, 2002.

-
- [18] S. S. Haykin. *Neural networks : a comprehensive foundation*. Prentice Hall, 1999.
- [19] T. Hinterberger, A. Kübler, J. Kaiser, N. Neumann, and N. Birbaumer. A brain-computer interface (BCI) for the locked-in: comparison of different EEG classifications for the thought translation device. *Clinical Neurophysiology*, 114:416–425, Mar. 2003.
- [20] K. Hornik. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- [21] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.
- [22] K. Hornik. Some new results on neural network approximation. *Neural Networks*, 6(9):1069–1072, 1993.
- [23] A. Householder. *The theory of matrices in numerical analysis*. New York: Dover, 1975.
- [24] G.-B. Huang. Benchmarking ELM. School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 2004. Available at http://www.ntu.edu.sg/home/egbhuang/ELM_Benchmarking.htm.
- [25] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive networks. *IEEE Transactions on Neural Networks*, 17(4):879–892, July 2006.
- [26] G.-B. Huang, N.-Y. Liang, H.-J. Rong, P. Saratchandran, and N. Sundararajan. On-line sequential extreme learning machine. In *Proceeding of the IASTED International Conference on Computational Intelligence, 2005 (CI2005)*, Calgary, Canada, 4-6 jul., 2005.
- [27] G.-B. Huang, P. Saratchandran, and N. Sundararajan. An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. *IEEE*

- Transactions on Systems, Man and Cybernetics, Part B*, 34(6):2284–2292, 2004.
- [28] G.-B. Huang, P. Saratchandran, and N. Sundararajan. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Transactions on Neural Networks*, 16(1):57–67, 2005.
- [29] G.-B. Huang and C.-K. Siew. Extreme Learning Machine: RBF Network Case. In *Proceedings of the Eighth International Conference on Control, Automation, Robotics and Vision, 2004 (ICARCV2004)*, Kunming, China, 6-9 Dec., 2004.
- [30] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications (in press). *Neurocomputing*, 2006.
- [31] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. In *IEEE International Joint Conference on Neural Networks, 2004 (IJCNN2004)*, Budapest, Hungary, 25-29 July, 2004.
- [32] J. R. Hughes. *EEG in clinical practice*. Butterworth-Heinemann, 1994.
- [33] T.-P. Jung, C. Humphries, T. Lee, M. McKeown, V. Iragui, S. Makeig, and T. Sejnowski. Removing electroencephalographic artifacts by blind source separation. *Psychophysiology*, 37:163–178, 2000.
- [34] V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5(6):954–975, 1993.
- [35] Z. A. Keirn. Alternative modes of communication between man and machine. Master’s thesis, Purdue University, Dec. 1998.
- [36] T. Lagerlund, F. Sharbrough, and N. Busacker. Spatial filtering of multichannel electroencephalographic recordings through principal component analysis

- by singular value decomposition. *Journal of Clinical Neurophysiology*, 14:73–82, 1997.
- [37] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate on-line sequential learning algorithm for feedforward networks, (in press). *IEEE Transaction on Neural Networks*, 17(6), 2006.
- [38] N.-Y. Liang, P. Saratchandran, G.-B. Huang, and N. Sundararajan. Classification of mental tasks from EEG signals using extreme learning machine. *International Journal of Neural System*, 16(1):29–38, 2006.
- [39] Y. Liao, S.-C. Fang, and H. L. W. Nuttle. Relaxed conditions for radial-basis function networks to be universal approximators. *Neural Networks*, 16(7):1019–1028, 2003.
- [40] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, 1977.
- [41] M. Meltser, M. Shoham, and L. M. Manevitz. Approximating functions by neural networks: a constructive solution in the uniform norm. *Neural Networks*, 9(6):965–978, 1996.
- [42] K.-R. Müller, C. W. Anderson, and G. E. Birch. Linear and nonlinear methods for brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11:165–169, June 2003.
- [43] C. Neuper and G. Pfurtscheller. Event-related dynamics of cortical rhythms: frequency-specific features and functional correlates. *International Journal of Psychophysiology*, 43(1):41–58, 2001.
- [44] L. S. Ngia, J. Sjöberg, and M. Viberg. Adaptive neural nets filter using a recursive Levenberg-Marquardt search direction. In *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers*, volume 1, pages 697–701, 1-4 Nov., 1998.

- [45] J. Pardey, S. Roberts, and L. Tarassenko. A review of parametric modelling techniques for EEG analysis. *Medical Engineering & Physics*, 18:2–11, 1996.
- [46] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.
- [47] J. Park and I. W. Sandberg. Approximation and radial-basis-function networks. *Neural Computation*, 5(2):305–316, 1993.
- [48] G. Pfurtscheller, C. Guger, G. Müller, G. Krausz, and C. Neuper. Brain oscillations control hand orthosis in a tetraplegic. *Neuroscience Letters*, 292(3):211–214, 2000.
- [49] G. Pfurtscheller, G. R. Müller, J. Pfurtscheller, H. J. Gerner, and R. Rupp. ‘Thought’ - control of functional electrical stimulation to restore hand grasp in a patient with tetraplegia. *Neuroscience Letters*, 351(1):33–36, 2003.
- [50] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.
- [51] C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and its Applications*. New York: John Wiley, 1971.
- [52] K. Rohani, M. S. Chen, and M. T. Manry. A mapping approach for designing neural sub-nets. In *Proceedings of the First IEEE-SP Workshop on Neural Networks for Signal Processing*, pages 70–79, Princeton, New Jersey, Sep.29–Oct.2 1991.
- [53] K. Rohani, M. S. Chen, and M. T. Manry. Neural subnet design by direct polynomial mapping. *IEEE Transactions on Neural Networks*, 3:1024–1026, November, 1992.
- [54] W. Rudin. *Real and complex analysis, 3rd edition*. McGraw-Hill, 1987.

-
- [55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.
 - [56] M. A. Sartori and P. J. Antsaklis. A simple method to derive bounds on the size and to train multilayer neural networks. *IEEE Transactions on Neural Networks*, 2:467–471, 1991.
 - [57] F. Scarselli and A. C. Tsoi. Universal approximation using feedforward neural networks: a survey of some existing methods, and some new results. *Neural Networks*, 11(1):15–37, 1998.
 - [58] D. Serre. *Matrices: Theory and Applications*. Springer, 2002.
 - [59] I. O. B. P.-D. Shane M. Haas, Mark G. Frei and J. Radel. EEG ocular artifact removal through ARMAX model system identification using extended least squares. *Communications in Information and Systems*, 3:19–40, June, 2003.
 - [60] S. Tamura. Capabilities of a three layer feedforward neural network. In *IEEE International Joint Conference on Neural Networks, 1991 (IJCNN1991)*, volume 3, pages 2757–2762, Seattle, WA, USA, 18-21 Nov., 1991.
 - [61] V. N. Vapnik. *Statistical learning theory, 2nd ed.* New York : Wiley, 1998.
 - [62] V. N. Vapnik. *The nature of statistical learning theory, 2nd ed.* New York: Springer, 2000.
 - [63] T. M. Vaughan. Guest editorial brain-computer interface technology: A review of the second international meeting. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11:94–109, June 2003.
 - [64] G. L. Wallstroma, R. E. Kassb, A. Millerc, J. F. Cohnd, and N. A. Foxe. Automatic correction of ocular artifacts in the EEG: a comparison of regression-based and component-based methods. *International Journal of Psychophysiology*, 53:105–119, 2004.

-
- [65] J. R. Wolpaw, N. Birbaumer, W. J. Heetderks, D. J. McFarland, P. H. Peckham, G. Schalk, E. Donchin, L. A. Quatrano, C. J. Robinson, and T. M. Vaughan. Brain-computer interface technology: A review of the first international meeting. *IEEE Transactions on Rehabilitation Engineering*, 8:164–173, June 2000.
- [66] L. Yingwei, N. Sundararajan, and P. Saratchandran. A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, 9(2):461–478, 1997.
- [67] L. Yingwei, N. Sundararajan, and P. Saratchandran. Performance evaluation of a sequential minimal radial basis function (RBF) neural network learning algorithm. *IEEE Transactions on Neural Networks*, 9(2):308–318, 1998.
- [68] R.-X. Zhang. *Efficient sequential and batch learning artificial neural network methods for classification problems*. PhD thesis, Nanyang Technological University, 2005.

Appendix

Chunk-by-chunk formula of RLS algorithm

Given a series of chunks of input-output samples $\{(\mathbf{X}_{n_j}, \mathbf{T}_{n_j})\}_{j=1}^m$ where each input-output sample is distinct from others. For the model of a SLFN with n hidden units using infinitely differentiable activation function $G : \mathbb{R}^d \rightarrow \mathbb{R}$, it represents a set of input-output mapping functions defined by:

$$\mathcal{N}_{G,n}(x) = \sum_{i=1}^n \beta_i \mathbf{G}(\mathbf{x}, \mathbf{a}_i, b_i) \quad (\text{A.1})$$

Let the hidden unit parameters $\{(\mathbf{a}_i, b_i)\}_{i=1}^n$ be randomly chosen from $\mathbb{R}^d \times \mathbb{R}$ according to some continuous probability distribution and then fixed throughout the learning procedure. Only the output unit parameters $\{\beta_i\}_{i=1}^n$ are iteratively updated according to the chunk-by-chunk formula of RLS algorithm [9]:

In the initialization phase, we set $j = 0$. Let an initial set of n_0 distinct input-output samples be accumulated. The initial output unit parameters are estimated by solving the linear system of equations:

$$\mathbf{H}_{N_0} \beta = \mathbf{T}_{N_0} \quad (\text{A.2})$$

where $N_0 = n_0$. $\mathbf{H}_{N_0} \in \mathbb{R}^{n_0 \times n}$, $n_0 \geq n$, and $\text{rank}(\mathbf{H}_{N_0}) = n$.

According to Theorem 4.1, the solution for initial output unit parameters are

uniquely determined by:

$$\beta^{(N_0)} = \mathbf{K}_{N_0}^{-1} \mathbf{H}_{N_0}^T \mathbf{T}_{N_0} \quad (\text{A.3})$$

where $\mathbf{K}_{N_0} = \mathbf{H}_{N_0}^T \mathbf{H}_{N_0}$.

In the update phase, we suppose the first chunk of input-output samples is available. The output unit parameters are estimated by solving the current linear system of equations:

$$\mathbf{H}_{N_1} \beta = \mathbf{T}_{N_1} \quad (\text{A.4})$$

where $N_1 = N_0 + n_1$.

Similarly we have the solution:

$$\begin{aligned} \beta^{(N_1)} &= \mathbf{K}_{N_1}^{-1} \mathbf{H}_{N_1}^T \mathbf{T}_{N_1} \\ &= \mathbf{K}_{N_1}^{-1} (\mathbf{H}_{N_0}^T \mathbf{T}_{N_0} + \mathbf{H}_{n_1}^T \mathbf{T}_{n_1}) \end{aligned} \quad (\text{A.5})$$

where

$$\begin{aligned} \mathbf{K}_{N_1} &= \begin{bmatrix} \mathbf{H}_{N_0} \\ \mathbf{H}_{n_1} \end{bmatrix}^T \begin{bmatrix} \mathbf{H}_{N_0} \\ \mathbf{H}_{n_1} \end{bmatrix} \\ &= \mathbf{H}_{N_0}^T \mathbf{H}_{N_0} + \mathbf{H}_{n_1}^T \mathbf{H}_{n_1} \\ &= \mathbf{K}_{N_0} + \mathbf{H}_{n_1}^T \mathbf{H}_{n_1} \end{aligned} \quad (\text{A.6})$$

For $\mathbf{K}_{N_0} \mathbf{K}_{N_0}^{-1} = \mathbf{I}_{N_0}$, Equation (A.5) may be rewritten as:

$$\begin{aligned} \beta^{(N_1)} &= \mathbf{K}_{N_1}^{-1} (\mathbf{K}_{N_0} \mathbf{K}_{N_0}^{-1} \mathbf{H}_{N_0}^T \mathbf{T}_{N_0} + \mathbf{H}_{n_1}^T \mathbf{T}_{n_1}) \\ &= \mathbf{K}_{N_1}^{-1} (\mathbf{K}_{N_0} \beta^{(N_0)} + \mathbf{H}_{n_1}^T \mathbf{T}_{n_1}) \end{aligned} \quad (\text{A.7})$$

From Equation (A.6), we have $\mathbf{K}_{N_0} = \mathbf{K}_{N_1} - \mathbf{H}_{n_1}^T \mathbf{H}_{n_1}$ and substitute it into Equation (A.7) and obtain:

$$\begin{aligned}\beta^{(N_1)} &= \mathbf{K}_{N_1}^{-1}(\mathbf{K}_{N_1}\beta^{(N_0)} - \mathbf{H}_{n_1}^T \mathbf{H}_{n_1}\beta^{(N_0)} + \mathbf{H}_{n_1}^T \mathbf{T}_{n_1}) \\ &= \beta^{(N_0)} + \mathbf{K}_{N_1}^{-1} \mathbf{H}_{n_1}^T (\mathbf{T}_{n_1} - \mathbf{H}_{n_1}\beta^{(N_0)})\end{aligned}\quad (\text{A.8})$$

where $\mathbf{K}_{N_1} = \mathbf{K}_{N_0} + \mathbf{H}_{n_1}^T \mathbf{H}_{n_1}$.

Observe that the solution $\beta^{(N_1)}$ is modified by adding to $\beta^{(N_0)}$ a correction term $\mathbf{K}_{N_1}^{-1} \mathbf{H}_{n_1}^T (\mathbf{T}_{n_1} - \mathbf{H}_{n_1}\beta^{(N_0)})$. The above argument can be generalized in a straightforward way. For instance, when the j^{th} chunk of input-output samples is presented, the solution $\beta^{(N_j)}$ is updated by:

$$\beta^{(N_j)} = \beta^{(N_{j-1})} + \mathbf{K}_{N_j}^{-1} \mathbf{H}_{n_j}^T (\mathbf{T}_{n_j} - \mathbf{H}_{n_j}\beta^{(N_{j-1})}) \quad (\text{A.9})$$

where $\mathbf{K}_{N_j} = \mathbf{K}_{N_{j-1}} + \mathbf{H}_{n_j}^T \mathbf{H}_{n_j}$.

Further the recursive algorithm for updating $\mathbf{K}_{N_j}^{-1}$ using the Sherman-Morrison formula [23](p.51) is given by:

$$\mathbf{K}_{N_j}^{-1} = \mathbf{K}_{N_{j-1}}^{-1} - \mathbf{K}_{N_{j-1}}^{-1} \mathbf{H}_{n_j} (\mathbf{I} + \mathbf{H}_{n_j}^T \mathbf{K}_{N_{j-1}}^{-1} \mathbf{H}_{n_j})^{-1} \mathbf{H}_{n_j}^T \mathbf{K}_{N_{j-1}}^{-1} \quad (\text{A.10})$$

Let $\mathbf{P}_{N_j} = \mathbf{K}_{N_j}^{-1}$, then the update equations (A.10) and (A.9) can be rewritten as:

$$\mathbf{P}_{N_j} = \mathbf{P}_{N_{j-1}} - \mathbf{P}_{N_{j-1}} \mathbf{H}_{n_j} (\mathbf{I} + \mathbf{H}_{n_j}^T \mathbf{P}_{N_{j-1}} \mathbf{H}_{n_j})^{-1} \mathbf{H}_{n_j}^T \mathbf{P}_{N_{j-1}} \quad (\text{A.11})$$

and

$$\beta^{(N_j)} = \beta^{(N_{j-1})} + \mathbf{P}_{N_j} \mathbf{H}_{n_j}^T (\mathbf{T}_{n_j} - \mathbf{H}_{n_j}\beta^{(N_{j-1})}) \quad (\text{A.12})$$

Equations (4.38) and (4.39) are consisted of the chunk-by-chunk formula of RLS algorithm and applied for iteratively updating the output unit parameters of **OS-ELM** in the learning mode of chunk-by-chunk.