Optimization based extreme learning machine : applications and data-driven extensions

Zong, Weiwei

2013

Zong, W. (2013). Optimization based extreme learning machine : applications and data-driven extensions. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/54858>

<https://doi.org/10.32657/10356/54858>

Optimization Based Extreme Learning Machine: Applications and Data-Driven Extensions

ZONG WEIWEI

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

2013

Statement of Originality

I hereby certify that the content of this thesis is the result of work done by me and has not been submitted for a higher degree to any other University or Institution.

.....

Date

.....

ZONG WEIWEI

Acknowledgments

First of all, my sincerest gratitude goes to my supervisor, Prof. Huang Guangbin, for his academic guidance and spiritual encouragement. He has witnessed how I am growing as a beginner in the research field until I complete the thesis. I really appreciate the confidence he has shown in me at bad times and the applause from him every time when I have a little bit of achievement. Indeed, he is an excellent model for me in the field of research, for his enthusiasm in new ideas and new heights, and the attitude of conscientious.

Secondly, I would like to thank my friends Mu Yang, Lan Yuan, Zhang Rui, and Zhou Hongming for their valuable technical discussions and support. I would also like to thank my friends and labmates for the fun and joy we have had during the past four years.

Furthermore, I owe a big thank to technical staff, Mr. Tan Peng Chye from Machine Learning Research Laboratory, in the School of Electrical and Electronic Engineering for being helpful in assisting my research work all the time.

Last but not least, I would like to express the deepest gratitude to my parents and my grandparents for their selfless and constant love over the years. I would never have achieved anything without their support and concern.

Table of Contents

Acknowledgments	i
Summary	vii
List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Research Background	1
1.2 Motivations and Objectives	3
1.3 Major Contributions	5
1.4 Organization of the Thesis	7
2 Literature Review	9
2.1 Machine Learning Techniques	9
2.1.1 Extreme learning machine	10
2.1.2 Support vector machine	15
2.1.3 Least square support vector machine	23
2.2 Optimization Method Based Extreme Learning Machine for Classification	26

2.2.1	Extreme learning machine using optimization method	26
2.2.2	Discussion	27
2.3	Unified Solution Provided by Extreme Learning Machine Framework .	29
2.3.1	Algorithmic view	30
2.3.2	Optimization view	31
2.3.3	Wide type of feature mappings and kernels	31
2.3.4	Comprehensive comparison between SVM, LS-SVM and ELM	33
3	Optimization Method Based ε-ELM	38
3.1	Proposed ε -ELM for Regression Based on Optimization Method	38
3.2	Discussion	41
3.3	Performance Verification	42
3.3.1	User specified parameters	43
3.3.2	Experimental results	43
3.4	Summary	45
4	Face Recognition Based on Extreme Learning Machines	49
4.1	Background	50
4.2	Brief on Dimensionality Reduction Methods	52
4.2.1	Principal component analysis	52
4.2.2	Linear discriminant analysis	53
4.2.3	Discriminative locality alignment	53
4.3	Proposed Multi-class Classifiers Based on ELM in Face Recognition System	56
4.4	Performance Evaluation	58

4.4.1	Face databases	58
4.4.2	Experimental results	59
4.4.3	Discussions	69
4.5	Extension of Work: Face recognition with Kernelized ELM	70
4.5.1	Kernelized ELM	71
4.5.2	Performance evaluation	72
4.6	Summary	74
5	Learning to Rank with Extreme Learning Machine	77
5.1	Introduction	77
5.2	Pointwise ELM	78
5.3	Linear Random Node and Linear Kernel	80
5.3.1	Random linear node	81
5.3.2	Linear kernel	82
5.4	Pairwise ELM	83
5.4.1	Proposed pairwise ELM	83
5.4.2	Discussions with other pairwise approaches	87
5.5	Performance Evaluation	88
5.5.1	Data specification	88
5.5.2	Parameter settings	88
5.5.3	Experiment results	89
5.6	Summary	90
6	Online Sequential Learning Based on Enhanced Extreme Learning Machine Using Left or Right Pseudo-Inverse	93

6.1	Introduction	93
6.2	Brief Review of eELM	95
6.3	The Proposed Online Sequential Learning Algorithm	95
6.3.1	OS-eELM-left	96
6.3.2	The proposed OS-eELM-right	96
6.4	Discussions	100
6.5	Performance Evaluation	101
6.5.1	Data specification	101
6.5.2	Parameter settings	101
6.5.3	Comparison of average testing accuracy	102
6.5.4	Comparison of training time	102
6.6	Summary	105
7	Weighted Extreme Learning Machine for Imbalance Learning	106
7.1	Introduction	106
7.2	Evaluation Metrics	107
7.3	Weighted ELM	109
7.3.1	Effect of imbalance on classification performance of ELM	110
7.3.2	Proposed weighted ELM as binary classifier	111
7.3.3	Proposed weighted ELM as multiclass classifier	113
7.4	Discussions	114
7.4.1	Essence of the difference between unweighted ELM and weighted ELM	115
7.4.2	Movement of boundary with/without weight	115

7.4.3	How to determine the weight	117
7.4.4	Difference with other weighted ELMs and weighted least square algorithms	118
7.5	Performance Evaluation	121
7.5.1	Data specification	121
7.5.2	Parameter settings	123
7.5.3	Experiment results	124
7.6	Summary	131
8	Conclusions and Future Work	136
8.1	Conclusions	136
8.2	Future work	139
	Author's Publications	142
	Bibliography	144

Summary

Artificial neural network, or commonly referred to as “neural network”, is a successful example of how human nature has led technology. Due to the massively parallel computing capability, neural network is able to model quite complex relationship. However, traditional learning algorithms in neural network require iterative parameter tuning and often suffer from problems like local minimum and slow convergence.

Extreme learning machine (ELM) is able to overcome all the problems above. Proposed as a learning algorithm for the single-hidden layer feedforward neural networks (SLFNs), ELM was later extended to the “generalized” SLFNs where the hidden nodes might take wide types of forms not limited to neuron type. The main feature of ELM lies in the random hidden nodes. To elaborate, the parameters of the hidden layer are randomly generated according to some continuous distribution and the output weights connecting the hidden layer and output layer are then analytically determined. The unique feature of ELM has rendered ELM an efficient training algorithm for “generalized” SLFNs with very simple implementation. Moreover, the universal approximation theorem of ELM has guaranteed good performance as long as the hidden layer mapping is any bounded piecewise continuous function. The solution to minimize the least square loss function as well as the norm of output weight actually attempts to achieve the promising generalization performance on unknown testing data. However, compared with another powerful machine learning technique, support vector machine (SVM), original ELM might not have comparable performance, especially on classification problems of small scale.

Hence, researchers on ELM have been seeking for some other methods to improve the

generalization performance. Standard optimization method was thus considered in the realization of ELM. Not only better performance in classification was achieved, but also a fact was revealed that ELM and SVM are actually consistent from optimization point of view. The resultant ELM classifier based on standard optimization method was found with comparable performance as SVM. What's more, the implementation of ELM is much easier since the performance is insensitive to parameters. Afterwards, ELM was further analyzed from optimization point of view and solution of kernel version was derived. So far the unified framework of ELM has been formed that includes traditional neural networks, support vector networks, and regularized networks.

Since the ELM theory is only developed since very recent years, there are plenty of places ELM can be applied. In this thesis, works of ELM successfully applied in real world applications, such as face recognition system and relevance ranking for information retrieval are elaborated in detail in the following chapters.

In real world applications, the natural data is with different characteristics. For example, situations when data is not available at once or data is of large scale often arise. In this case, online sequential learning model of a machine learning technique is generally regarded as one typical solution. In this thesis, online sequential model based on ELM framework is provided so that not only all the advantages of ELM over other machine learning techniques are pertained but also the aforementioned problems are solved. Another situation happens quite often is that the training data is not well balanced. Any normal machine learning technique that assumes well balanced data distribution is supposed with the tendency to bias the performance. In this case, weighted version of ELM is proposed as the most straightforward and efficient method to tackle such problem.

List of Figures

2.1	An example of linear support vector machine.	17
2.2	An example of linear support vector regression and its ε -insensitive loss function.	20
3.1	(a) Primal ELM network for regression: 1) L hidden nodes; 2) The output of the i -th hidden node is $G(\mathbf{a}_i, b_i, \mathbf{x}_i)$; 3) the output weight vector linking the hidden layer to the output node is $\beta = \sum_{i=1}^N (\alpha_i - \alpha_i^*) K_{\text{ELM}}(\mathbf{x}, \mathbf{x}_i)$. (b) Dual ELM network for regression: 1) N_s hidden nodes where N_s is the number of support vectors; 2) The output of the s -th hidden node is $K_{\text{ELM}}(\mathbf{x}, \mathbf{x}_s)$; 3) the output weight connecting the s -th hidden node to the output node is $(\alpha_s - \alpha_s^*)$, where $K_{\text{ELM}}(\mathbf{x}, \mathbf{x}_s) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \dots, G(\mathbf{a}_L, b_L, \mathbf{x})]^T \cdot [G(\mathbf{a}_1, b_1, \mathbf{x}_s), \dots, G(\mathbf{a}_L, b_L, \mathbf{x}_s)]^T$	47
3.2	The relationship of testing RMSE to the parameter selection of ε -SVR (C, γ) and ε -ELM (C, L)	48
4.1	The main procedures in a face recognition system.	50
4.2	Face samples: Row 1 for YALE, Row 2 for ORL, Row 3 for UMIST. . .	59
4.3	Face samples from BioID database.	59
4.4	Two-dimensional representation of the images of selected individuals from YALE.	61

4.5	Comparison of the generalization performance of NN, SVM-OAA and ELM-OAA on YALE.	66
4.6	Comparison of the training time of SVM-OAA and ELM-OAA on YALE.	66
4.7	Comparison of the training time of SVM-OAO and ELM-OAO on YALE.	67
4.8	The effect of tunable parameters ((C, γ) for SVM and (C, L) for ELM) on the generalization performance in holistic learning with 3 training samples from YALE.	67
4.9	The recognition rate for each person from database YALE using ELM-OAA.	68
4.10	Row 1: images of Identity 3 from YALE; Row 2: images of Identity 11 from YALE.	68
4.11	Display of sample images from database YALE.	72
4.12	Testing accuracy with respect to various parameter selections of (C, γ) for LS-SVM and ELM with Gaussian kernels.	75
5.1	Illustration of input space and output space for pointwise learning and pairwise learning.	84
5.2	Relationship of the MAP with choices of the parameters tested on Folder 1 from OHSUMED.	89
6.1	Testing RMSE of abalone with respect to C and L	102
7.1	An example of ROC graph.	109
7.2	Effect of imbalanced class distribution on synthetically generated dataset, with 3 settings of trade-off constant C , where the red cross represents the majority class, the blue circle represents the minority class.	111

7.3	The decision boundary produced by unweighted ELM and weighted ELM on dataset <i>banana</i> with imbalance ratio 1:43 on Row 1 or 3:43 on Row 2.	116
7.4	Display of G-mean distribution of yeast1 with different choices of C and L for Sigmoid node or C and γ for Gaussian kernel.	124
7.5	Display of testing result in terms of accuracy (white bar) and G-mean (blue bar), using unweighted ELM, weighted ELM W1 and weighted ELM W2 , with Sigmoid node or Gaussian kernel.	134
7.6	ROC graph of three ELM classifiers using Sigmoid node on three datasets with different imbalance ratios.	135

List of Tables

3.1	Specification of regression problems	43
3.2	Comparison of the average performance between ε -SVR and ε -ELM. . .	44
4.1	The optimal parameter (P -dim, k_1 , k_2 , C_{DLA}) of DLA in different systems.	60
4.2	Comparison between the NN, SVM-OAA, SVM-OAO, ELM-OAA and ELM-OAO on database YALE.	62
4.3	Comparison between the NN, SVM-OAA, SVM-OAO, ELM-OAA and ELM-OAO on database UMIST.	63
4.4	Comparison between the NN, SVM-OAA, SVM-OAO, ELM-OAA and ELM-OAO on database ORL.	64
4.5	Comparison between the NN, SVM-OAA, SVM-OAO, ELM-OAA and ELM-OAO on database BioID.	65
4.6	Comparison of performance of LS-SVM and ELM on database YALE. . .	72
5.1	MAP value obtained from various pointwise learning methods and pair- wise learning methods of every folder from dataset OHSUMED, TD2003 and TD2004.	90
5.2	NDCG value obtained from various pointwise learning methods and pairwise learning methods of every folder from dataset OHSUMED. . .	91

5.3	Precision value obtained from various pointwise learning methods and pairwise learning methods of every folder from dataset TD2003 and TD2004.	91
6.1	Specification of benchmarking datasets.	101
6.2	Performance comparison on regression problems	103
6.3	Performance comparison on multiclass classification problems	104
7.1	Specification of binary classification problems.	122
7.2	Specification of binary and multiclass problems from UCI.	122
7.3	Performance result of binary problems with imbalance ratio $\in (0, 0.2)$	128
7.4	Performance result of binary problems with imbalance ratio $\in (0.2, 1)$	129
7.5	Performance result of multiclass problems.	130
7.6	P-value of the t-test for pairwise comparisons between the algorithms over binary problems.	131
7.7	Performance result in terms of accuracy for binary problems with imbalance ratio $\in (0, 0.2)$	132
7.8	Performance result in terms of accuracy for binary problems with imbalance ratio $\in (0.2, 1)$	133

Chapter 1

Introduction

1.1 Research Background

As a popular branch of artificial intelligence, machine learning has been studied by numerous researchers from various fields over the years. Machine learning is essentially a collection of algorithms to interpret and reveal the unknown relationship or distribution hidden in the real world empirical data. The application areas include function approximation (regression analysis), classification or data processing (such as clustering, filtering, etc.). A common challenge confronted by any machine learning technique is how to learn the complex pattern from given training data, which is usually a subset of all possible samples, so as to generalize to the vast unknown data.

Artificial neural networks (ANNs) [35], or “neural networks” for short, are one type of machine learning techniques that have attracted many researchers. Designed to imitate human brain, the structure and processing of a neural network is found parallel to that of human neural system. The input information is processed in the basic units, referred to as “neurons”, where the powerful computation in terms of all kinds “activation functions” is carried out. Afterwards, a network with complex interconnections between the neurons is built up to store the learning information for any specific application. The

massively parallel distributed structure renders the neural network capable of solving complex and intractable problems.

According to the way the neurons are connected, there are different architectures of a neural network to fit in different kinds of situations such as single layer feedforward neural network, multi layer feedforward neural network, recurrent neural network and so on. Among those architectures, feedforward neural networks (FNNS) are extensively studied. The simplest form of an FNN is single-hidden layer feedforward neural network (SLFN), where there are three layers: input layer to absorb the input data, hidden layer to process the data and the output layer to display the output value of the system.

Traditionally, the learning algorithms of SLFNs require iterative tuning of parameters in the hidden layer. Take the representative algorithm backpropagation (BP) [79] based on gradient-descent for example. The training process is tedious and time consuming. Furthermore, such algorithms normally suffer from problems like local minimum, and slow convergence rate.

Extreme learning machines (ELMs) were proposed originally as the learning algorithms for the single-hidden layer feedforward neural networks [58, 57, 50], and then extended to the “generalized” SLFNs which may not be neuron alike [47, 48]. The main feature of ELMs lies in that we generate random computational hidden nodes instead of iteratively tuning the hidden nodes’ parameters as in the traditional way of training SLFNs and the universal approximation capability is guaranteed. And then output weight is analytically determined as the solution of a linear system. Hence ELMs have extreme fast speed and convenience in implementation. And ELMs have been successfully applied in various function approximation and classification applications.

In addition to neural networks, another machine learning technique is well known for its classification performance: support vector machine (SVM). With the kernel tricks, the underlying relationship between input and output can be approximated as the linear combination of the training samples. Instead of least square method in ELM, the original SVM makes use of standard optimization method to determine the value of those

linear coefficients, called “Lagrange multipliers”. To reduce computational complexity of original SVM caused by the quadratic programming tools, least square SVM (LS-SVM) was proposed where the least square loss function is of researchers’ interest and the optimization problem is solved using least square method. Note that, the architecture of the SVMs model, namely support vector network, can be considered as one special type of SLFN [52].

1.2 Motivations and Objectives

Three main approaches exist in the literature of training of feedforward networks: (1) gradient-descent based method like backpropagation (BP) [79] for multi-layer feedforward neural networks; (2) standard optimization method based like support vector machines (SVMs) [21] for a specific network called support vector network which is essentially one type of single-hidden layer feedforward networks (SLFNs); (3) least square based such as extreme learning machines (ELMs) [58, 57, 50, 47, 48] for the generalized SLFNs.

As a popular machine learning method for (binary) classification, SVMs have been studied over the decades. With the aim to maximize the marginal distance of two different classes and minimize the training errors, SVMs obtain the solution by using the standard optimization method.

However, compared with traditional SVMs: (1) for most medium and large applications, ELMs usually achieve the same generalization performance as SVM but at much faster learning speed and only one learning parameter (the number of hidden nodes L) is required to be tuned; (2) for classification applications especially those of small scale, ELM may not perform as well as SVM although SVM may require longer time spent on learning and parameter tuning.

Huang et al. [52] have shown that SVMs and ELMs are consistent from optimization point of view. Further ELM has been extended to support vector network for classifi-

cation applications where standard optimization method instead of least square method was used to compute the solution. This optimization method based ELM was verified to have better generalization performance with insensitive parameter which results in less intensive human intervention.

Optimization method was proved feasible on ELM for classification applications [52], it is natural that optimization based ELM is extended to regression problems, which is the first objective of this thesis.

To the best of our knowledge, there are few classification applications of ELMs on real world biometric problems, such as face recognition, finger recognition. On the other hand, SVMs have been widely applied in face recognition applications. That is one objective of my research work to propose ELM-based classifier on real world biometric problems. Since optimization method based ELM was shown with promising classification capability comparable as the powerful SVM and much easier to be implemented, it becomes feasible and reasonable to apply ELM in the interesting real world task of face recognition.

Later, a breakthrough was reported in [55], where least square based ELM was further analyzed from optimization point of view. And it was found that ELM actually provides a unified solution for the “generalized” SLFNs where there are many choices for the hidden layer feature mappings. In other words, the framework of ELM is so vast that traditional neural networks, support vector networks and regularized networks are all included and unified. The advantage of ELMs over SVMs in terms of generalization performance and computational efficiency on regression, binary classification, especially multiclass classification problems was proved and validated on large number of benchmarking datasets [55].

Since the ELM theory, especially the newly proposed ELM framework, is only recently developed, there are large amount of real world applications where ELMs have not been under consideration. Among those applications, we pick out face recognition to test the classification performance and relevance ranking to test the regression capability

of ELM. Because both applications are very important and quite popular during recent years, it would be meaningful to demonstrate performance of ELM on such tasks. This is another objective of this thesis.

Speaking of the algorithm itself, improvements are always needed to satisfy all kinds of demands. Natural data are usually not as standard as assumed. Sometimes, a dataset with too large data size is given, which has been a hot topic recently and has caused trouble for lots of machine learning techniques. Or when the dataset is sequentially arriving, online sequential learning mode instead of the batch mode is highly desired. Another case we need to pay special attention to is when the data distribution is imbalanced. It is well known that imbalanced data would bias the performance of any algorithm designed only based on balanced data. Therefore, according to characteristic of natural data, the original ELM algorithm needs to be modified or further improved, which is another objective of this thesis.

1.3 Major Contributions

The major contributions of the thesis are summarized as follows.

In the first work, ε -ELM for regression applications is proposed where the solution of SLFNs is computed through standard optimization method. The idea is based on the consistency between ELM and SVM from optimization point of view. And the resultant method is verified to have similar performance to SVM but with less sensitive parameters resulting in less intensive human intervention. So far, optimization method based ELM can be realized as a classifier as well as a regressor. To test the regression and classification capability of ELM, two interesting real world applications are studied in the following works.

In the second work, optimization method based ELM for classification is applied on face recognition (one of the most important and popular biometric recognition applica-

tions), which is new in literature of both ELM and face recognition to the best of our knowledge. Comparison with the popular classifiers nearest neighbor (NN) and support vector machine (SVM) is conducted in different systems where the face representation method is varied. The experimental results show that the proposed multiclass classifiers based on ELM has comparable generalization performance to SVM together with less sensitive user specified parameters which makes the implementation much easier. Later, after the ELM framework proposed in [55], kernelized ELM is applied in face recognition as an extension of the previous work. Compared with optimization method based ELM and SVMs, the latest ELM framework is able to tackle multiclass classification problems directly, which is computationally efficient. The performance of ELM is demonstrated much better than that of LS-SVM with much easier implementation and less computational cost.

All the rest of works are based on the unified ELM framework [55].

In the third work, ELM is extended to the field of relevance ranking as a regressor for the first time, where pointwise RankELM and pairwise RankELM are proposed respectively in accordance with two different trends to perceive the ranking problems. The proposed ranking algorithms actually have a large variety of choices when it comes to the feature mapping functions, namely any type of piecewise continuous function or in the form of a kernel. In the experiment, as far as computation cost is concerned, the linear random node and linear kernel are used. The experimental results show that ranking algorithms based on ELM outperform state-of-art linear ranking algorithms on most of the cases.

It is found that data in real applications is of different characteristics, such as large scale, imbalanced distribution, and etc., which has posed limits on regular machine learning techniques. To deal with large scale or sequentially arriving data, in the fourth work, an online sequential learning algorithm based on non-kernel version of the ELM framework (referred to as enhanced ELM or eELM for short) was proposed, which is capable of learning data on a one-by-one basis or chunk-by-chunk basis. Simulations on six benchmarking datasets have shown that, by adding a positive value to the diagonal of

$\mathbf{H}\mathbf{H}^T$ and $\mathbf{H}^T\mathbf{H}$, the generalization performance of our proposed methods outperform the original online sequential ELM (OS-ELM). Moreover, the proposed OS-eELM actually provides solutions for both SLFNs and regularized networks.

In the fifth work, a weighted ELM is proposed for binary and multiclass classification tasks. To benefit from unweighted ELM, the proposed method is simple in theory and implementation; wide types of feature mapping or kernels are available as options; comparable or better generalization performance is achievable compared to the conventional machine learning techniques. On the other hand, by incorporating the information of imbalance class distribution, the proposed method is able to deal with data with imbalanced class distribution.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows:

Chapter 2 gives the literature review on current popular machine learning techniques for feedforward networks.

Chapter 3 shows the first work where ε -ELM based on optimization method is proposed for regression problems.

Chapter 4 presents the second work where a multi-class classifier based on ELMs for the face recognition application is proposed. The extension of the work is also included where kernelized ELM is applied in face recognition.

Chapter 5 exhibits the third work where ranking algorithms based on ELM framework are proposed.

Chapter 6 presents the fourth work where the online sequential mode based on enhanced ELM is proposed to deal with large scale data or sequentially arriving data.

Chapter 7 displays the last work where unified ELM framework is extended to cope with situation of imbalanced data distribution.

Chapter 8 concludes all the works and addresses potential future work.

Chapter 2

Literature Review

2.1 Machine Learning Techniques

Motivated by the human brain, which is a highly complex, nonlinear and parallel computational processor, neural network is designed to model the way in which the brain performs a particular task or function of interest. There are two information processing capabilities that render neural networks capable of solving complex problems that are normally intractable. Firstly, the massively parallel distributed structure of neural network makes it powerful in computation. Secondly, neural network has the ability to learn and therefore generalize [35].

Among all the structures of neural networks, feedforward networks are most popular. Feedforward neural networks have been extensively used in various applications. But the traditional learning algorithm is suffering from the tedious parameter tuning process, where all the parameters of all layers need to be tuned. Sometimes it may take hours even days to train the network. Besides, as one of the popular training methods, gradient descent-based method faces problems like slow convergence rate, local minima and overfitting. Hence, some analytical learning methods were proposed to solve these problems mentioned above. And they are attracting more and more research interests

nowadays. The extreme learning machine (ELM) is one of them.

Except neural networks, there are other popular machine learning techniques for training of feedforward networks. In the early 1990s, Vapnik and coworkers reported their work of Support Vector Machines (SVMs) [21], which were actually powerful supervised networks for solving regression and classification problems. Some results showed that ELMs can operate thousands times faster than SVMs in certain applications and provide comparable performance. Nevertheless, in some other applications, SVMs produce better results than ELMs [58, 57, 50, 47, 48].

Earlier this year, it has been shown that SVMs and ELMs actually share the similar network architecture with consistent purpose. And ELMs were extended to support vector network for classification applications, where standard optimization method instead of original least square method was utilized. It was shown by the simulations that optimization method based ELM has comparable classification capability as SVM but with less sensitive user specified parameters [52].

In this section, we review fundamentals of ELMs and SVMs respectively and present the basis of optimization method based ELM for classification which is actually the combination of SVMs and ELMs in ELMs feature space.

2.1.1 Extreme learning machine

In the beginning extreme learning machines were proposed to train single-hidden layer feedforward neural networks [58, 57, 50], which are one particular type of feedforward neural networks where there is only one layer of hidden nodes. Later ELMs were extended to the “generalized” single-hidden layer feedforward networks (SLFNs) where the hidden nodes can be neuron alike or non-neuron alike [47, 48].

Single-hidden layer feedforward networks and its approximation capability

Without loss of generality, the output of an SLFN with L hidden nodes with respect to input $\mathbf{x}$ can be expressed as:

$$f_L(\mathbf{x}) = \sum_{i=1}^L \beta_i g_i(\mathbf{x}) = \sum_{i=1}^L \beta_i G(\mathbf{x}, \mathbf{a}_i, b_i) \quad (2.1)$$

$$\mathbf{a}_i \in C^d, \mathbf{x}_i \in C^d, b_i \in C, \beta_i \in C.$$

where β_i is the output weight connecting the i th hidden node and the output node, and $g_i(\mathbf{x})$ or $G(\mathbf{x}, \mathbf{a}_i, b_i)$ is the output function of the i th hidden node. The hidden nodes are not limited to neuron type of nodes. Researchers have used different kinds of nodes, including additive or radial basis function (RBF) type of nodes, multiplicative nodes, fuzzy rules, fully complex nodes, hinging functions, high-order nodes, ridge polynomials, wavelets, Fourier terms, etc. The activation function of three popular neuron nodes are shown below.

$$\begin{aligned} \text{Additive hidden nodes: } g_i(\mathbf{x}) &= g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \quad \mathbf{a}_i \in R^d, \mathbf{x} \in R^d, b_i \in R, \\ \text{RBF hidden nodes: } g_i(\mathbf{x}) &= g(b_i |\mathbf{x} - \mathbf{a}_i|), \quad \mathbf{a}_i \in R^d, \mathbf{x} \in R^d, b_i \in R, \\ \text{Fully complex hidden nodes: } g_i(\mathbf{x}) &= \prod_{j=1}^{s_i} g(\mathbf{a}_{ij} \cdot \mathbf{x} + b_i), \quad \mathbf{a}_{ij} \in C^d, \mathbf{x} \in C^d, b_i \in C. \end{aligned} \quad (2.2)$$

where g is the activation function, s_i is an integer constant and $\cdot$ represents inner product of vectors.

Research on the approximation capabilities of SLFNs has mainly focused on two aspects: universal approximation on compact input sets and approximation in a set con-

sisting of a finite number of training samples.

Many researchers have worked on the universal approximation capability of SLFNs. In conventional neural network theories, SLFNs with additive or radial basis function (RBF) hidden nodes are universal approximators as long as all the parameters of the networks can be adjusted. Hornik [43] has shown that SLFNs with additive hidden nodes can approximate any continuous mapping provided that the activation function is continuous, bounded, and non-constant. Leshno [64] further proved that SLFNs with additive hidden nodes and any non-polynomial activation function can approximate any continuous target functions. In addition, Park and Sandberg [73] have shown that RBF networks with appropriately chosen kernels are also able to approximate any continuous target function.

In real applications, where usually only a finite number of training samples are available, the approximation capability of SLFNs has also been studied by researchers. Huang and Babri [46] proved that SLFNs with at most N hidden nodes and with almost any nonlinear activation function can learn N distinct observations with probability one. Moreover, with the input weights chosen randomly, SLFNs with N additive hidden nodes were proved to be able to learn N distinct observations with arbitrarily small error [45].

Later, Huang et al. [50] has proved in theory that the SLFNs with randomly generated hidden nodes are indeed universal approximators, where the activation function for additive nodes can be any bounded non-constant piecewise continuous functions and the activation functions for RBF nodes can be any integrable piecewise continuous functions.

Threshold activation functions in neural networks are highly desirable because the threshold units make it easy for hardware implementation. But many conventional popular learning algorithms like gradient-based learning algorithm cannot be used directly to the learning process because the threshold function is not differentiable. Hence, in literature, many researchers usually approximate the threshold activation functions by using sigmoid functions which are differentiable. Huang et al. [56] proved that ELMs can

be used to train threshold networks without any modifications and better generalization performance is obtained than traditional BP methods.

Brief of extreme learning machines

Extreme learning machines were proposed on the basis of the findings above. An SLFN can be viewed as a linear system with hidden nodes randomly generated which is independent of the training data. Then least square based method is used to compute the output weight. ELMs tend to obtain the smallest training error as well as solution with smallest norm which indicates better generalization performance according to Bartlett's theory [7].

Suppose we have N distinct training samples $(\mathbf{x}_i, t_i)$, the output of an ELM with L hidden nodes is:

$$f(\mathbf{x}) = \sum_{i=1}^L \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) = \boldsymbol{\beta} \cdot \mathbf{h}(\mathbf{x}), \quad (2.3)$$

where $\mathbf{h}(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \dots, G(\mathbf{a}_L, b_L, \mathbf{x})]^T$ is the output vector for the hidden layer with respect to input $\mathbf{x}$. The parameters $(\mathbf{a}_i, b_i, i = 1, \dots, L)$ for hidden layer nodes are randomly assigned and the output weight β_i which connects the i -th hidden node to the output nodes is then analytically determined.

The linear system can be expressed in the matrix form:

$$\mathbf{H} \cdot \boldsymbol{\beta} = \mathbf{T}, \quad (2.4)$$

where

$$\mathbf{H}(\mathbf{a}_1, \dots, \mathbf{a}_L; b_1, \dots, b_L; \mathbf{x}_1, \dots, \mathbf{x}_N) = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times d} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times d}.$$
(2.5)

$\mathbf{H}$ is the hidden layer output matrix of the network with the i th row representing the output vector of hidden layer with respect to input $\mathbf{x}_i$; and the i th column representing the output of the i th hidden node with respect to all the inputs $(\mathbf{x}_1, \dots, \mathbf{x}_N)$.

With the aim to find the least square solution $\hat{\boldsymbol{\beta}}$ of the linear system:

$$|\mathbf{H} \cdot \hat{\boldsymbol{\beta}} - \mathbf{T}| = \min_{\boldsymbol{\beta}} |\mathbf{H} \cdot \boldsymbol{\beta} - \mathbf{T}|, \quad (2.6)$$

we have in ELMs:

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{T}, \quad (2.7)$$

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse [81, 77] of $\mathbf{H}$ which tends to achieve the smallest norm of $\boldsymbol{\beta}$ while keeping the training error minimum. In theory, the least square solution of minimum norm ($\boldsymbol{\beta}$) is supposed to be unique.

If the number of hidden nodes (L) is equal to the number of training samples (N), matrix $\mathbf{H}$ is square and invertible. So the solution of the linear system can be obtained with error zero. But in most of the cases, number of the hidden nodes is much less than the number of training samples. Hence, the Moore-Penrose generalized inverse shown above is adopted to compute the solution with arbitrarily small error.

Besides, according to Bartlett's theory [7], for feedforward networks achieving smallest training error, the smaller the norm of the weight is, the better generalization performance the networks tend to have. Therefore, ELMs not only tend to achieve smallest training error but also good generalization performance.

Learning algorithm of ELM

Given the training data $(\mathbf{x}_i, t_i), i = 1, \dots, N$, the training process of ELM is:

- (1) Generate L hidden nodes with the parameters $(\mathbf{a}, b)$ of each node randomly assigned (the input weights and the hidden node biases in case of additive hidden node or neuron center and impact width in case of RBF hidden node);
- (2) Calculate the hidden layer output matrix $\mathbf{H}$;
- (3) Calculate the output weight $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$, where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of hidden layer output matrix $\mathbf{H}$.

2.1.2 Support vector machine

Support vector machines (SVMs) [21] have been widely used in the last decades in various applications including regression and pattern recognition problems. SVMs are based on the structural risk minimization principle from statistical learning theory [89], which is then solved by the standard optimization method.

Support vector classification

As seen from Figure 2.1 the essence of conventional support vector machines for classification (SVC) is to separate training samples belonging to two classes with some hyperplane after mapping the training samples from the input space to the feature space, resulting in a separating hyperplane having the maximum marginal distance ($\frac{2}{\|\mathbf{w}\|}$) between the two classes. Traditionally the standard optimization method is used to solve this problem.

Slack variables

Originated to tackle binary classification problems, it is believed that training points are separable after mapped to higher dimensional space, if not separable in the input space. In the case of Gaussian kernel, the dimension of feature space could reach infinity. An ideal situation would be all of the training points fall on the right side and away from the two decision boundaries. However, the data is not necessarily separable in practise. What's more, noises and outliers exist for quite a number of problems. Ergo, Slack variables (ξ in Equation 2.8) are introduced to allow such errors, which generalizes to training data already separable. Hence, decision boundaries with slack variables are also termed “soft margin”.

Given a set of training data $(\mathbf{x}_i, t_i)$, $i = 1, \dots, N$, where $\mathbf{x}_i \in \mathbf{R}^d$ and $t_i \in \{-1, 1\}$, SVM [21] aims to find a hyperplane to separate the data with maximum separating boundary into two classes after mapping these data from input space to the feature space: $\phi(\mathbf{x}) : \mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$.

To maximize the separating margin distance as well as to minimize the training error we

Figure 2.1: An example of linear support vector machine.

have the following optimization function:

$$\begin{aligned}
 &\text{minimize: } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i, \\
 &\text{subject to: } t_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad i = 1, \dots, N, \\
 &\quad \xi_i \geq 0, \quad i = 1, \dots, N,
 \end{aligned} \tag{2.8}$$

where C is a user specified parameter.

According to the Karush-Kuhn-Tucker theorem [29], the optimization problem is equivalent to:

$$\begin{aligned}
 &\text{minimize: } L_D = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N t_i t_j \alpha_i \alpha_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) - \sum_{i=1}^N \alpha_i \\
 &\text{subject to: } \sum_{i=1}^N t_i \alpha_i = 0 \\
 &\quad 0 \leq \alpha_i \leq C, i = 1, \dots, N,
 \end{aligned} \tag{2.9}$$

where each Lagrange multiplier α_i corresponds to a training example $(\mathbf{x}_i, t_i)$.

Kernel trick

Kernel trick was introduced to cure the pain of high computational cost caused feature space of high dimension, which is specifically infinite in some cases. The dot product of two training samples (u, v) in the feature space $\phi(\mathbf{u}) \cdot \phi(\mathbf{v})$ can be computed as if some kernel function in the input space $K(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u}) \cdot \phi(\mathbf{v})$. Meanwhile, the feature mapping function $\phi(\cdot)$ is not necessarily known. Hence, as shown from two representative kernel functions, kernel functions of two training samples can be perceived as an indication of how close these two samples are. Note that (p, σ) in Equation 2.10 are the kernel parameters.

$$\begin{aligned} \text{Polynomial Kernels: } K(\mathbf{u}, \mathbf{v}) &= (\mathbf{u} \cdot \mathbf{v})^p, \mathbf{u}, \mathbf{v} \in \mathbf{R}^{d_L}, \\ \text{Gaussian Kernels: } K(\mathbf{u}, \mathbf{v}) &= e^{-|\mathbf{u}-\mathbf{v}|^2/2\sigma^2}. \end{aligned} \tag{2.10}$$

The dual optimization problem can be rewritten by replacing $\phi(\mathbf{u}) \cdot \phi(\mathbf{v})$ with $K(\mathbf{u}, \mathbf{v})$:

$$\begin{aligned} \text{minimize: } L_D &= \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N t_i t_j K(\mathbf{x}_i, \mathbf{x}_j) \alpha_i \alpha_j - \sum_{i=1}^N \alpha_i, \\ \text{subject to: } \sum_{i=1}^N t_i \alpha_i &= 0, \\ 0 \leq \alpha_i &\leq C, i = 1, \dots, N. \end{aligned} \tag{2.11}$$

A bound on the generalization performance and VC dimensions

There is a weird but reasonable phenomena in supervised/unsupervised machine learning techniques: on one hand, the training error of those data we have in hand is expected to

be as small as possible; on the other hand, the same evaluation criteria is also expected not too small to restrict the generalization performance of the trained model on other unseen testing data. In other words, it is the testing error that is of more interest rather than the empirical training error. However, to predict the testing error on unseen data seems almost impossible. Luckily, an upper bound on the testing error is provided by Vapnik [15] to cast new light into theoretical study of the generalization performance of support vector classifier.

For a binary SVM classifier model trained on a particular set $\{\mathbf{x}_i, y_i\}$, the Lagrange multiplier α_i for each of the l samples $\mathbf{x}_i$ is fixed. R_{emp} represents the empirical training error and $R(\alpha)$ the testing error users are truly interested in. The following bound holds with probability $1 - \eta$:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l}\right)}, \quad (2.12)$$

where h is a non-negative integer termed as Vapnik Chervonenkis (VC) dimension. The right hand side of Equation (2.12) is referred to as the “risk bound” and the second term of right hand side is termed as “VC confidence”. A commonly misunderstanding about the upper bound is that the risk is guaranteed. As a matter of fact the Equation only holds with a certain probability.

Further study has shown that the VC confidence is monotonic in h . That is to say, the only way to make the bound tight with training samples of fixed size is through minimizing h . However, it is also shown that VC dimension of SVMs can be very large (even infinite). Therefore, we argue that the VC dimension makes limited sense in practice despite of the theoretical significance it brings. Nevertheless, loose upper bound for the testing error does not mean poor generalization performance. In fact, SVMs have been demonstrated on various applications with excellent generalization capability.

Figure 2.2: An example of linear support vector regression and its ϵ -insensitive loss function.

Support vector regression

This part gives a brief review of ϵ -support vector regression (ϵ -SVR). ϵ -SVR uses ϵ -insensitive loss function defined as follows:

$$L_{\epsilon}(t, y) = \begin{cases} |t - y| - \epsilon, & \text{for } |t - y| \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (2.13)$$

where t is the target output and y is the estimated output. ϵ is a user specified parameter which forms an ϵ -tube. This loss function indicates the penalization for error larger than ϵ and it is found that any absolute error less than or equal to ϵ can be neglected.

Given the training data $(\mathbf{x}_i, t_i)_{i=1}^N$, $\mathbf{x} \in R^d$, $t \in R$, SVR algorithm aims to obtain the estimation of the target output, denoted by y , which is given by:

$$y = \sum_{j=1}^m w_j \phi_j(\mathbf{x}) + b = \mathbf{w} \cdot \phi(\mathbf{x}) + b, \quad (2.14)$$

where

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \dots, \phi_m(\mathbf{x})]^T,$$

and

$$\mathbf{w} = [w_1, \dots, w_m]^T.$$

$\phi(\mathbf{x})$ is the feature mapping function which maps the training data from the input space to the feature space. According to the definition of loss function, data points inside the ε -tube do not have impact on the solution of the optimization problem.

To reach the minimum empirical risk and the smallest $\|\mathbf{w}\|$ which minimizes the model complexity so as to obtain good generalization performance [89], the problem can be written in the form of a convex optimization problem:

$$\begin{aligned} &\text{minimize: } \frac{1}{2} \|\mathbf{w}\|^2, \\ &\text{subject to: } t_i - \mathbf{w} \cdot \phi(\mathbf{x}_i) - b \leq \varepsilon, \\ &\quad \mathbf{w} \cdot \phi(\mathbf{x}_i) + b - t_i \leq \varepsilon, \quad i = 1, \dots, N. \end{aligned} \tag{2.15}$$

By introducing the slack variables ξ_i, ξ_i^* , we have:

$$\begin{aligned}
 &\text{minimize: } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*), \\
 &\text{subject to: } t_i - \mathbf{w} \cdot \phi(\mathbf{x}_i) - b \leq \varepsilon + \xi_i, \\
 &\quad \mathbf{w} \cdot \phi(\mathbf{x}_i) + b - t_i \leq \varepsilon + \xi_i^*, \\
 &\quad \xi_i, \xi_i^* \geq 0 \quad i = 1, \dots, N.
 \end{aligned} \tag{2.16}$$

The primal Lagrange function for this optimization problem is:

$$\begin{aligned}
 L_p(\mathbf{w}, \xi, \xi^*, \alpha, \alpha^*, \gamma, \gamma^*) = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) - \sum_{i=1}^N \alpha_i [\mathbf{w} \cdot \phi(\mathbf{x}_i) + b - t_i + \varepsilon + \xi_i] \\
 & - \sum_{i=1}^N \alpha_i^* [t_i - \mathbf{w} \cdot \phi(\mathbf{x}_i) - b - \varepsilon + \xi_i^*] - \sum_{i=1}^N (\gamma_i \xi_i + \gamma_i^* \xi_i^*),
 \end{aligned} \tag{2.17}$$

where $\alpha_i, \alpha_i^*, \gamma_i, \gamma_i^* \geq 0$ are the Lagrange multipliers. According to the saddle point condition that the partial derivatives of L_p with respect to $(\mathbf{w}, b, \xi_i, \xi_i^*)$ are equal to zero, we further have:

$$\begin{aligned}
 &\sum_{i=1}^N (\alpha_i - \alpha_i^*) = 0 \\
 &\mathbf{w} = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \phi(\mathbf{x}_i) \\
 &\gamma_i = C - \alpha_i \\
 &\gamma_i^* = C - \alpha_i^*.
 \end{aligned} \tag{2.18}$$

By substituting (2.18) in (2.17), we can get the corresponding dual optimization problem:

$$\begin{aligned}
 \text{maximize: } L_d(\alpha_i, \alpha_i^*) &= \sum_{i=1}^N d_i(\alpha_i - \alpha_i^*) - \varepsilon \sum_{i=1}^N (\alpha_i + \alpha_i^*) \\
 &\quad - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) K_{\text{SVR}}(\mathbf{x}_i, \mathbf{x}_j), \\
 \text{subject to: } \sum_{i=1}^N (\alpha_i - \alpha_i^*) &= 0, \\
 0 \leq \alpha_i &\leq C, \quad i = 1, \dots, N, \\
 0 \leq \alpha_i^* &\leq C, \quad i = 1, \dots, N,
 \end{aligned} \tag{2.19}$$

where $K_{\text{SVR}}(\mathbf{x}_i, \mathbf{x}_j) = \phi^T(\mathbf{x}_i)\phi(\mathbf{x}_j)$.

With $\mathbf{w} = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \mathbf{x}_i$, the estimated output function can be expressed as:

$$f(x) = \mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{N_s} (\alpha_i - \alpha_i^*) K_{\text{SVR}}(\mathbf{x}, \mathbf{x}_i) + b, \tag{2.20}$$

where N_s is the number of support vectors which reveals the complexity of the network structure.

2.1.3 Least square support vector machine

Suykens and Vandewalle [83] proposed a least square version to SVM classifier (LS-SVM). The formulation of LS-SVM is based on the one of original SVM where l_1 norm penalty is replaced with $l-2$ norm and inequality constraint is replaced with the equality constraint. An straightforward illustration will be: in Figure 2.1, instead of attempting

to push data points beyond the boundaries, LS-SVM tries to push the data points onto the two boundaries. In LS-SVM, the classification problem is formulated as:

$$\begin{aligned} \text{minimize: } & \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \frac{1}{2} \sum_{i=1}^N \xi_i^2 \\ \text{subject to: } & t_i(\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) = 1 - \xi_i, \quad i = 1, \dots, N, \end{aligned} \quad (2.21)$$

where ξ_i is the training error.

The inequality constraints (2.8) are used in the conventional SVM while equality constraints (2.21) are used in LS-SVM. Different from SVM which needs to solve quadratic programming, LS-SVM only needs to solve a set of linear equations and thus the least square approach can be implemented. Compared to SVM, LS-SVM produces better generalization performance and has less computational cost in many applications.

To train such an LS-SVM is equivalent to solving the following primal Lagrange problem:

$$L_{P_{LS-SVM}} = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \frac{1}{2} \sum_{i=1}^N \xi_i^2 - \sum_{i=1}^N \alpha_i (t_i (\mathbf{w} \cdot \phi(\mathbf{x}_i) + b) - 1 + \xi_i). \quad (2.22)$$

Based on the Karush-Kuhn-Tucker theorem [29], the following solution can be obtained for LS-SVM:

$$\begin{bmatrix} \mathbf{0} & \mathbf{T}^T \\ \mathbf{T} & \frac{\mathbf{I}}{C} + \Omega_{LS-SVM} \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{T}^T \\ \mathbf{T} & \frac{\mathbf{I}}{C} + \mathbf{Z}\mathbf{Z}^T \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \vec{\mathbf{1}} \end{bmatrix}, \quad (2.23)$$

where

$$\mathbf{Z} = \begin{bmatrix} t_1 \phi(\mathbf{x}_1) \\ \vdots \\ t_N \phi(\mathbf{x}_N) \end{bmatrix} \quad (2.24)$$

$$\Omega_{LS-SVM} = \mathbf{Z}\mathbf{Z}^T.$$

The feature mapping $\phi(\mathbf{x})$ is a row vector¹, $\mathbf{T} = [t_1, t_2, \dots, t_N]^T$, $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_N]^T$ and $\vec{\mathbf{1}} = [1, 1, \dots, 1]^T$. In LS-SVM, as $\phi(\mathbf{x})$ is usually unknown, Mercer's condition [35] can be applied to matrix Ω_{LS-SVM} :

$$\Omega_{LS-SVM_{i,j}} = t_i t_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) = t_i t_j K(\mathbf{x}_i, \mathbf{x}_j). \quad (2.25)$$

The decision function of LS-SVM classifier is: $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^N \alpha_i t_i K(\mathbf{x}, \mathbf{x}_i) + b)$.

Loss of sparsity

One of advantages of SVM is brought by the sparse representation. In fact, only a small amount of Lagrange multipliers α are normally non-zero. That is to say, the final decision function is modeled by only a small group of data points. Hence, whenever a new data comes, it does not need to be involved in computation of the kernel function with every training data. In other words, the testing phase becomes sort of easier and faster.

Due to the usage of $l-2$ norm penalty term in the formulation of LS-SVM, the condition constraints are changed accordingly. The Karush-Kuhn-Tucker (KKT) condition

¹Feature mappings $\phi(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$ are defined as row vectors while the rest are column in default.

constraints are derived by making the derivative of the primal Lagrangian function equal to zero. And the Lagrange multipliers in LS-SVM are proportional to the training errors by a constant C . Although most of α s are no more zeros, it does not twist the fact that training procedure of LS-SVM is much easier than SVM, thanks to the resultant linear equations.

2.2 Optimization Method Based Extreme Learning Machine for Classification

2.2.1 Extreme learning machine using optimization method

It has been shown that from the optimization point of view SVMs and ELMs are actually consistent. So Huang et al. [52] have further studied ELMs for classification using standard optimization method.

ELM can be reformulated as:

$$\begin{aligned} \text{minimize: } & \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \xi_i, \\ \text{subject to: } & t_i \beta \cdot h(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1, \dots, N, \\ & \xi_i \geq 0, \quad i = 1, \dots, N, \end{aligned} \tag{2.26}$$

where $h(\mathbf{x})$ is the ELM mapping for input $\mathbf{x}$. Hence, similar optimization function and constraints for ELM can be drawn according to (2.8). The difference between ELM and SVM lies in the bias b for the hyperplane which appeared in (2.8) for SVM. Here b is not needed since the separating hyperplane in ELM feature space tends to pass through the origin.

By defining ELM kernel as:

$$\begin{aligned} K_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j) &= h(\mathbf{x}_i) \cdot h(\mathbf{x}_j) \\ &= [G(\mathbf{a}_1, b_1, \mathbf{x}_i), \dots, G(\mathbf{a}_L, b_L, \mathbf{x}_i)]^T \cdot [G(\mathbf{a}_1, b_1, \mathbf{x}_j), \dots, G(\mathbf{a}_L, b_L, \mathbf{x}_j)]^T, \end{aligned} \quad (2.27)$$

where $h()$ is still the random feature mapping because parameters $(\mathbf{a}, b)$ are randomly generated as in conventional ELM. We have our final dual optimization function:

$$\begin{aligned} \text{minimize: } L_D &= \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N t_i t_j K_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j) \alpha_i \alpha_j - \sum_{i=1}^N \alpha_i \\ \text{subject to: } &0 \leq \alpha_i \leq C, i = 1, \dots, N. \end{aligned} \quad (2.28)$$

As can be seen from above, ELM for classification is essentially a combination of ELM kernel and standard optimization method which is used to find the optimal solution for ELM. By comparing (2.11) and (2.28), it is found that ELM has milder optimization constraints than SVM.

2.2.2 Discussion

Although from the optimization point of view there exists consistency between ELM and SVC that both intend to minimize the empirical error and the norm of output weight ($\mathbf{w}$ for SVC, β for ELM) so as to achieve good generalization performance, the difference between the concept of separating hyperplane of SVMs and that of ELMs is obvious. In ELM feature space, the separating hyperplane tends to pass the origin with probability one. That is to say, there is no bias b in the optimization constraint: $t_i \beta \cdot h(\mathbf{x}_i) \geq 1 - \xi_i$. Consequently in ELM case the optimization condition $\sum_{i=1}^N \alpha_i t_i = 0$ is not needed.

While in SVC, both optimization conditions: $\sum_{i=1}^N \alpha_i t_i = 0$ and $0 \leq \alpha_i \leq C, \forall i$ need to be satisfied.

Missing of bias b

In a word, the optimization based ELM proposed by Huang et al. [52] is not just replacing the SVM kernel with ELM random kernel. The optimization constraints are changed as well, thanks to the missing of bias b in the decision function in the feature space. Consequently, ELM for classification has only one optimization condition: $0 \leq \alpha_i \leq C, \forall i$. But SVC needs to satisfy $\sum_{i=1}^N \alpha_i t_i = 0$ and $0 \leq \alpha_i \leq C, \forall i$. Obviously ELM has relatively loose optimization conditions. The solution of SVM is sub-optimal to that of ELM in the sense that the search region is more restricted [52].

ELM feature mapping and less human intervention

A more appropriate perception for the number of hidden layer nodes would be the dimension of the feature space. Hence the number of hidden nodes is actually just a parameter of the mapping function. No wonder good performance is achieved when large number of hidden nodes are selected. This is consistent with the fundamental of SVM that the data is linearly separable after mapped to a space with much higher dimension. Just because of the insensitivity of selection of number of hidden nodes, ELM has less parameters to tune, which is supposed to be the biggest advantage over SVM. Note that in SVM, there are two parameters to tune, the trade-off constant C and the kernel parameter.

Sparsity

Experiment results have shown that the number of support vectors in optimization based ELM [52] is sparse, thanks to the l_1 norm penalty term used in the loss function and

the quadratic programming tools to solve the constraint optimization problem. One of the important advantages of SVM maintains which efficiently speeds up the evaluation phase.

2.3 Unified Solution Provided by Extreme Learning Machine Framework

The computational complexity of the conventional SVM is closely related to the number of training samples, more precisely, at least quadratic since quadratic programming tool is used in finding the solution. Least square support vector machine (LS-SVM) [83] was proposed as an easier and faster implementation of SVM. By reformulating the optimization structure, inequality optimization constraints in original SVM are replaced by equality equations, which can be solved by least square methods instead of quadratic programming.

In the previous section, ELM using quadratic programming tool as in conventional SVM was reviewed. Later, Huang, et al. [55] proposed a kernelized ELM and found that kernelized ELM actually simplifies the implementation of least square SVM (LS-SVM). Furthermore, it was found in [55] that the framework of ELM actually provides a unified solution for a “generalized” SLFN. The support vector network, other regularized network, and even the traditional single hidden-layer feedforward (neural) network are all included and united in ELM framework with a single solution (2.31, 2.39).

The perspectives of ELM as a least-square based learning algorithm or solution of an optimization problem actually reflect each other and are presented below.

2.3.1 Algorithmic view

The main feature of ELM that distinguishes from conventional neural network learning algorithms is the random generation of hidden nodes. More precisely, the parameters of the hidden nodes are randomly assigned independent of the training samples and the hidden layer output (with L nodes) can be presented by a row vector $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \dots, h_L(\mathbf{x})]$ where $\mathbf{x}$ is the input sample. Given N training samples $(\mathbf{x}_i, t_i)$, the mathematical model of the SLFNs is

$$\mathbf{H}\beta = \mathbf{T}, \quad (2.29)$$

where $\mathbf{H}$ is the hidden layer output matrix and β is the output weight

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix}. \quad (2.30)$$

The least square solution with minimal norm is analytically determined using Moore-Penrose “generalized” inverse [77, 81].

$$\begin{aligned} \text{When } N < L: \quad \beta &= \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \\ \text{When } L < N: \quad \beta &= \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T} \end{aligned} \quad (2.31)$$

As can be seen from the two formulas above, a positive value $\frac{1}{C}$ is added to the diagonal of $\mathbf{H}\mathbf{H}^T$ or $\mathbf{H}^T \mathbf{H}$ in order for better generalization performance [41]. Users are offered

two formulas to select from according to the size of training data.

2.3.2 Optimization view

Solution of (2.29) can also be obtained using the optimization method. Similar to SVMs which aim to minimize the training errors and maximize the marginal distance between two classes, the goal of ELM is the same:

$$\text{minimize: } \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 \text{ and } \|\boldsymbol{\beta}\|. \quad (2.32)$$

Similar to LS-SVM, the optimization problem is mathematically written as

$$\begin{aligned} \text{minimize: } & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^N \|\boldsymbol{\xi}_i\|^2, \\ \text{subject to: } & \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = \mathbf{t}_i^T - \boldsymbol{\xi}_i^T, \quad i = 1, \dots, N, \end{aligned} \quad (2.33)$$

where $\boldsymbol{\xi}_i = [\xi_{i,1}, \dots, \xi_{i,m}]^T$ is the training error vector of the m output nodes with respect to the training sample $\mathbf{x}_i$. Here C is the regularization parameter to represent the trade-off between the minimization of training errors and the maximization of the marginal distance.

According to KKT theorem [29], same solution as (2.31) is obtained.

2.3.3 Wide type of feature mappings and kernels

The feature mapping in ELM $\mathbf{h}(\mathbf{x}) = [h(\mathbf{x})_1, \dots, h(\mathbf{x})_L]$, also called the hidden layer output vector, actually maps the data from the original data space to the hidden layer

space with dimensionality L . As the paramount feature of ELM, the parameters $(\mathbf{a}_i, b_i)$ in the hidden layer node function $h(\mathbf{x})_i = G(\mathbf{a}_i, b_i, \mathbf{x})$ are randomly generated according to any continuous probability distribution. In other words, the hidden layer need not be tuned, in contrast with the conventional neural network learning algorithms.

Theoretically speaking, the hidden layer node function $G(\mathbf{a}, b, \mathbf{x})$ includes almost all nonlinear piecewise continuous functions as long as ELM universal approximation capability theorems are satisfied [51, 55]. A wide type of feature mapping functions can be considered which include but not limit to

- Sigmoid function

$$G(\mathbf{a}, b, \mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b))} \quad (2.34)$$

- Gaussian function

$$G(\mathbf{a}, b, \mathbf{x}) = \exp(-b\|\mathbf{x} - \mathbf{a}\|^2) \quad (2.35)$$

- Hardlimit function

$$G(\mathbf{a}, b, \mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{a} \cdot \mathbf{x} - b \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (2.36)$$

- Multiquadrics function

$$G(\mathbf{a}, b, \mathbf{x}) = (\|\mathbf{x} - \mathbf{a}\|^2 + b^2)^{1/2} \quad (2.37)$$

On the other hand, it is not necessary that the feature mapping function $\mathbf{h}(\mathbf{x})$ is always known. If $\mathbf{h}(\mathbf{x})$ is unknown to the users, a kernel matrix for ELM as shown in (2.38) is defined [55] in accordance with Mercer's conditions [35]. The most popular kernel in

use is Gaussian kernel $K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma\|\mathbf{u} - \mathbf{v}\|^2)$, where γ is the kernel parameter.

$$\Omega_{ELM} = \mathbf{H}\mathbf{H}^T : \Omega_{ELM,i,j} = h(\mathbf{x}_i) \cdot h(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) \quad (2.38)$$

Hence, the output function of an SLFN is kernelized as

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \\ &= \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{ELM} \right)^{-1} \mathbf{T}. \end{aligned} \quad (2.39)$$

Therefore, with different feature mappings or kernels, the proposed weighted ELM provides a unified solution for the “generalized” SLFNs, including conventional neural networks, support vector networks, regularization networks, polynomial networks, and etc.

2.3.4 Comprehensive comparison between SVM, LS-SVM and ELM

Among the machine learning techniques, it is the similarity from optimization point of view that connects least square based ELM and optimization method based SVM/LS-SVM. According to the literature of ELM, ELM using quadratic programming optimization tool as in conventional SVM is validated as efficient as SVM and parameter tuning process of ELM is much easier due to the insensitive parameters. However, this ELM can be viewed as a transition in ELM literature. Later, the optimization problem of ELM was re-analyzed using least square method as in LS-SVM. The interesting thing about the resultant solution is that it actually unifies a number of regularized type of networks,

where SVM/LS-SVM can be viewed as special cases in ELM framework. Nevertheless, it is very important that a comprehensive comparison between the three optimization based techniques is conducted from every aspect [55].

Feature mapping

The feature mapping $\phi(\mathbf{x})$ in SVM and LS-SVM is usually unknown so that kernel can be applied. While the feature mapping $\mathbf{h}(\mathbf{x})$ in ELM can be known as well as unknown. When it is known as ELM random node, a wide type of functions can be selected as the feature mapping functions as long as the universal approximation/classification capability is satisfied, which is also the unique feature of ELM. On the other hand, just as in SVM/LS-SVM, a kernel version is available when $\mathbf{h}(\mathbf{x})$ is unknown.

Take the popular sigmoid hidden layer function (which is a non-kernel feature mapping) for example. Experiment results have shown that the performance is insensitive to the function parameter L - number of hidden nodes. In fact, good performance can be achieved when L is set as a large value. Here is one explanation of this phenomena. Since the hidden layer function takes the role as mapping the data to a feature space, the number of hidden nodes is actually the dimension of that feature space. According to the fundamentals of SVM, it is believed that the data is linearly separable after mapped to a space with much higher dimension. Ergo, not only the flexibility of various feature mappings, but also the convenience of parameter tuning that ELM provides.

Universal approximation/classification capability

Another unique feature of ELM is the guaranteed universal approximation/classification capability [55] provided the random node or kernel function being piecewise continuous. While such capability on SVM/LS-SVM was never guaranteed.

Loss function type

The loss function used in ELM and LS-SVM is least square of errors, which render the Lagrange multipliers α proportional to the training errors. On the other hand, due to L1 loss function used in SVM, the Lagrange multipliers are partly zero, with those nonzero define the separating boundary.

Regression

The unified solution provided by ELM actually is directly applicable not only on regression problems, but also on classification problems. While, SVM/LS-SVM are only proposed for classification problems. To solve regression problems, different variants, such as ε -support vector regression, were proposed.

Multiclass classification

Based on the “generalized” SLFN, ELM is able to tackle multiclass classification problems without further modification, where the number of output nodes are set the same as number of classes and position of node with maximum output value represents the label. However, neither SVM or LS-SVM is capable of multiclass classification tasks. Usually decomposition methods or encoding/decoding schemes are incorporated to reduce a multiclass classification problem into multiple binary sub-problems, which obviously has increased the computational complexity.

Output function

As mentioned before, there is no bias b in the output function of ELM.

Kernel based ELM:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{ELM} \right)^{-1} \mathbf{T}.$$

Non-kernel based ELM:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H} \mathbf{H}^T \right)^{-1} \mathbf{T},$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}.$$

Kernel based SVM/LS-SVM:

$$f(\mathbf{x}) = \text{sign} \left(\sum_{i=1}^N \alpha_i t_i K(\mathbf{x}, \mathbf{x}_i) + b \right).$$

Performance

According to [55], the simulations on benchmarking datasets including 12 regression, 12 binary and 12 multiclass classification problems have shown that, ELM have better or at

least comparable performance than SVM/LS-SVM. Moreover, as far as computational complexity is concerned, apparently ELM is a better choice.

Chapter 3

Optimization Method Based ε -ELM

It has been shown that from the optimization method point of view extreme learning machines (ELMs) are consistent to support vector machines (SVMs) in classification applications but with milder optimization constraints [52]. In this work we further study ELM for regression using optimization method and the connection between ELM and ε -SVR. The following part presents the proposed method which will be verified to make the implementation of ε -SVR much easier.

3.1 Proposed ε -ELM for Regression Based on Optimization Method

The previous work [52] has proposed ELM for classification which is actually the combination of ELM and SVM. Similarly, with the analysis above, in this paper we combine ELM and ε -SVR to form ε -ELM for regression. Different from ε -SVR, ε -ELM uses ELM random kernel and different from ELM, standard optimization method instead of least square is used to compute the solution.

So we aim to find the optimal solution for the SLFNs with number of hidden nodes L :

$$f(\mathbf{x}) = \sum_{i=1}^L \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) = \beta \cdot h(\mathbf{x}), \quad (3.1)$$

where $h(\mathbf{x})$ is hidden layer feature mapping with function $G(\mathbf{a}_i, b_i, \mathbf{x})$. Note that the bias b appeared in output function of SVR is not needed here. The reason is that after mapped to ELM feature space, the hyperplane tends to pass through the origin.

With the definition of ε -sensitive loss function (same as in ε -SVR), we have:

$$\begin{aligned} & \text{minimize: } \frac{1}{2} \|\beta\|^2 \\ & \text{subject to: } t_i - \beta \cdot h(\mathbf{x}_i) \leq \varepsilon, \\ & \quad \beta \cdot h(\mathbf{x}_i) - t_i \leq \varepsilon, \quad i = 1, \dots, N, \end{aligned} \quad (3.2)$$

which can be reformulated by adding the slack variables ξ and ξ^* :

$$\begin{aligned} & \text{minimize: } \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N (\xi + \xi^*) \\ & \text{subject to: } t_i - \beta \cdot h(\mathbf{x}_i) \leq \varepsilon + \xi, \\ & \quad \beta \cdot h(\mathbf{x}_i) - t_i \leq \varepsilon + \xi^*, \quad i = 1, \dots, N, \end{aligned} \quad (3.3)$$

where C illustrates the trade off between the minimization of the norm of output weight and that of the empirical errors. Accordingly, we have the primal Lagrange function for

this optimization problem:

$$L_p(\beta, \xi, \xi^*, \alpha, \alpha^*, \gamma, \gamma^*) = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) - \sum_{i=1}^N \alpha_i [\beta \cdot h(\mathbf{x}_i) - t_i + \varepsilon + \xi_i] - \sum_{i=1}^N \alpha_i^* [t_i - \beta \cdot h(\mathbf{x}_i) - \varepsilon + \xi_i^*] - \sum_{i=1}^N (\gamma_i \xi_i + \gamma_i^* \xi_i^*). \quad (3.4)$$

Similarly, according to the saddle point condition, the partial derivative of the primal Lagrange function with respect to $(\beta, \xi$ and $\xi^*)$ should be equal to zero; while in ε -SVR case, it is the partial derivative with respect to $(\mathbf{w}, b, \xi$ and $\xi^*)$ that is supposed to vanish. This difference lies in the fact that data in ELM feature space tends to pass the origin. Hence bias b is no more needed in the case of ELM.

$$\begin{aligned} \beta &= \sum_{i=1}^N (\alpha_i - \alpha_i^*) h(\mathbf{x}_i) \\ \gamma_i &= C - \alpha_i \\ \gamma_i^* &= C - \alpha_i^* \end{aligned} \quad (3.5)$$

With the equations above and ELM kernel K_{ELM} defined in (2.27), the corresponding dual Lagrange function is shown as follows:

$$\begin{aligned} \text{maximize: } L_d(\alpha_i, \alpha_i^*) &= \sum_{i=1}^N t_i (\alpha_i - \alpha_i^*) - \varepsilon \sum_{i=1}^N (\alpha_i + \alpha_i^*) \\ &\quad - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i - \alpha_i^*) (\alpha_j - \alpha_j^*) K_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j), \end{aligned} \quad (3.6)$$

subject to: $0 \leq \alpha_i \leq C, \quad i = 1, \dots, N,$

$0 \leq \alpha_i^* \leq C, \quad i = 1, \dots, N.$

Comparing (3.6) with (2.19), it is found that ε -ELM has less optimization constraints than ε -SVR. It can be traced back in the saddle point condition (3.5) where we do not need to consider the partial derivatives of L_p against b as we do in the case of ε -SVR since b does not exist in the estimation function of ε -ELM.

Finally, the estimated output function of ε -ELM is:

$$f(\mathbf{x}) = \beta \cdot h(\mathbf{x}) = \sum_{i=1}^{N_s} (\alpha_i - \alpha_i^*) K_{\text{ELM}}(\mathbf{x}, \mathbf{x}_i), \quad (3.7)$$

which is similar to (2.20) but without b . Note that both ELM kernel and SVR kernel are used to map the training data from input space to feature space. But parameters in ε -SVR kernel require iterative tuning (for example γ for Gaussian kernel function) while ε -ELM kernel is randomly generated without user intervene.

3.2 Discussion

Similarities can be found between ε -SVR and ε -ELM: (1) Seen from the decision functions $f(x) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{N_s} (\alpha_i - \alpha_i^*) K(\mathbf{x}, \mathbf{x}_i)$, SVR can be interpreted as one particular “generalized” single-hidden layer feedforward networks (SLFNs). This SLFNs has N_s hidden nodes. For the i th hidden node represented by the i th support vector, the output is obtained by SVM kernel $K(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)$ with the weight connecting this hidden node to the output node represented as $(\alpha_i - \alpha_i^*)$. The network structure of ELM in the original view and the view of optimization method is shown in Figure 3.1. (2) The goal of both SVR and ELM is to minimize both the empirical error and the output weight ($\mathbf{w}$ for SVR; β for ELM) thus to achieve the minimum training error as well as good generalization performance.

But there exists significant differences between ε -SVR and ε -ELM: (1) In ELM frame

work, the hidden nodes are randomly generated with the number of hidden nodes specified by users. The process of generating the hidden nodes is independent of the training data. And the decision function of ELM is the linear combination of the output of all the hidden nodes. But the decision function of SVR is the linear combination of the support vectors whose number is not determined by the user. And the “hidden nodes” (support vectors) in ε -SVR is related to the training samples. (2) It has been proved that [58, 57, 50, 47, 48] in ELM feature space, the training data can be separated with probability one by a hyperplane which happens to pass through the origin. Thus the decision function of ε -ELM does not include b which has appeared in the decision function of ε -SVR. Consequently ε -ELM need to satisfy relatively loose optimization condition since the final dual Lagrange function of ε -ELM has one less optimization constraint.

3.3 Performance Verification

In this section, comparison about the performance of ε -SVR and ε -ELM is conducted on 9 benchmarking regression problems¹ (Table 3.1). Problems *bodyfat* and *space_ga* are from StatLib². The remaining data sets are from UCI [11]. For all the problems both the attribute values and the target value are scaled to [0,1]. Simulations on each data set are repeated for 50 times with training and testing data randomly selected for each trial.

We are using the QP solver in Kernel Method Matlab Toolbox as the solutions to the optimization problems in both SVR and ELM. All the experiments are carried out in MATLAB 7.1 running in a PC with Pentium 4, 2.99GHZ CPU.

¹<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html>

²<http://lib.stat.cmu.edu/datasets>

Table 3.1: Specification of regression problems

Datasets	# Attributes	# Training data	# Testing data
Pyrim	27	40	34
Triazines	60	90	96
Bodyfat	14	130	122
BostonHousing	13	250	256
Auto_mpg	7	320	78
Concrete	8	500	530
Mg	6	700	685
Space_ga	6	1600	1507
Abalone	8	2000	2177

3.3.1 User specified parameters

For both ε -SVR and ε -ELM, ε is set as 0.05 for the sake of convenience. That means the absolute error user can tolerate is set as 0.05. In ε -SVR, we choose the popular Gaussian kernel $K_{SVR}(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma|\mathbf{x} - \mathbf{x}_i|^2)$. Together with the cost parameter C (trade off between $\|\mathbf{w}\|$ and empirical error), similar to [44], we have 225 different combinations: $C \in \{2^{-2}, 2^{-1}, \dots, 2^{11}, 2^{12}\}$ and $\gamma \in \{2^{-10}, 2^{-9}, \dots, 2^3, 2^4\}$.

In ε -ELM, the ELM kernel is: $K_{ELM}(\mathbf{x}, \mathbf{x}_i) = h(\mathbf{x}) \cdot h(\mathbf{x}_i) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \dots, G(\mathbf{a}_L, b_L, \mathbf{x})]^T \cdot [G(\mathbf{a}_1, b_1, \mathbf{x}_i), \dots, G(\mathbf{a}_L, b_L, \mathbf{x}_i)]^T$ where $G(\mathbf{a}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b)))$ is sigmoid type of function and L represents the number of hidden nodes. 150 different combinations of (C, L) are tried where $C \in \{2^{-2}, 2^{-1}, \dots, 2^{11}, 2^{12}\}$ is similar like above and $L \in \{20, 30, 50, 80, 100, 200, 500, 1000, 2000, 3000\}$ is the same as in [52].

3.3.2 Experimental results

For each combination of (C, γ) or (C, L) , 50 trials with randomly generated training and testing data are carried out to obtain the average performance which is displayed in Table 3.2.

From Table 3.2 it is found that ε -SVR and ε -ELM have similar performance in average

Datasets	ε -SVR						ε -ELM				
	(C, γ)	Training Time(s)	Training RMSE	Testing RMSE	Testing Dev	# SVs	(C, L)	Training Time(s)	Training RMSE	Testing RMSE	Testing Dev
Pyrim	$(2^2, 2^0)$	0.0118	0.0387	0.0902	0.0283	18	$(2^{-2}, 100)$	0.0125	0.0629	0.0886	0.0281
Triazines	$(2^2, 2^3)$	0.0597	0.1314	0.1385	0.0030	468	$(2^{-2}, 100)$	0.0597	0.1213	0.1475	0.0126
Bodyfat	$(2^6, 2^3)$	0.0364	0.0365	0.0377	0.0078	11	$(2^0, 100)$	0.0280	0.0358	0.0375	0.0084
BostonHousing	$(2^3, 2^0)$	0.3589	0.0506	0.0798	0.0109	97	$(2^3, 1000)$	0.4076	0.0579	0.0803	0.0081
Auto_mpg	$(2, 2^{-1})$	0.6305	0.0546	0.0685	0.0096	126	$(2^2, 1000)$	0.6370	0.0656	0.0699	0.0075
Concrete	$(2^8, 2^0)$	3.2950	0.0522	0.0818	0.0039	231	$(2^9, 3000)$	3.8826	0.0559	0.0844	0.0052
Mg	$(2^{-2}, 2^{-1})$	5.2898	0.1311	0.1464	0.0108	59	$(2^8, 3000)$	8.7061	0.1279	0.1398	0.0038
Space_ga	$(2^8, 2^{-1})$	32.2185	0.0362	0.0391	0.0012	273	$(2^{12}, 3000)$	27.2980	0.0382	0.0402	0.0012
Abalone	$(2, 2^{-1})$	82.7747	0.0727	0.0760	0.0014	762	$(2^5, 3000)$	83.4928	0.0735	0.0769	0.0018

Table 3.2: Comparison of the average performance between ε -SVR and ε -ELM.

testing accuracy (RMSE), standard deviation of the testing RMSE and average time used for training. For problems pyrim, triazines and bodyfat with number of training data no more than or around 100, 100 hidden nodes are sufficient for the SLFNs to achieve the optimal performance. For problems with number of training data larger than 100, large amount of hidden nodes are needed (like 1000 or 3000) in our cases.

Figure 3.2 has shown the relationship of the average testing RMSE and parameter selections in both ε -SVR (C, γ) and ε -ELM (C, L). It is shown that ε -SVR is sensitive with the parameter selection, especially the choice of γ . In ε -ELM the performance is found insensitive to the choice of parameters. And the good performance is normally achieved when the number of hidden nodes is large.

3.4 Summary

In this work ε -ELM was proposed for regression as a combination of ε -SVR and the original ELM, where standard optimization method helps to realize the solution of SLFNs with ELM random kernel. It was shown that ε -SVR and ε -ELM are consistent in the aim of minimizing both the training error and the output weight ($\mathbf{w}$ for ε -SVR and β for ε -ELM). Furthermore, since the hyperplane in ELM feature space tends to pass through the origin, in ε -ELM for regression, bias b is no longer needed in the output function which results in less optimization constraints. The experimental result has shown that ε -ELM is comparable to ε -SVR in the aspects of the testing rate, training time and the number of support vectors. But ε -ELM is not sensitive to number of hidden nodes, which makes it easy to implement without intensive parameter tuning. For data with small training size (less than or around 100), we can set the number of hidden nodes as 100; otherwise we can set the number of hidden nodes as a large value like 1000. So there is only one parameter we need to tune.

Together with work proposed in [52], there are optimization method based ELM for classification as well as for regression (ε -ELM). With respect to another optimization

method based technique, namely SVM, ELM based classifier and regressor are with comparable generalization performance. But parameter tuning is much easier because only one parameter needs to be tuned. However, compared to SVMs, which have been applied in various real world applications in area of computer vision and so on, ELMs are relatively new to researchers. Therefore, in the following work, optimization based ELM for classification is applied for the first time in a popular and interesting biometric, namely face recognition system.

(a) Primal ELM Network for Regression

(b) Dual ELM Network for Regression

Figure 3.1: (a) Primal ELM network for regression: 1) L hidden nodes; 2) The output of the i -th hidden node is $G(\mathbf{a}_i, b_i, \mathbf{x}_i)$; 3) the output weight vector linking the hidden layer to the output node is $\beta = \sum_{i=1}^N (\alpha_i - \alpha_i^*) K_{ELM}(\mathbf{x}, \mathbf{x}_i)$. (b) Dual ELM network for regression: 1) N_s hidden nodes where N_s is the number of support vectors; 2) The output of the s -th hidden node is $K_{ELM}(\mathbf{x}, \mathbf{x}_s)$; 3) the output weight connecting the s -th hidden node to the output node is $(\alpha_s - \alpha_s^*)$, where $K_{ELM}(\mathbf{x}, \mathbf{x}_s) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \dots, G(\mathbf{a}_L, b_L, \mathbf{x})]^T \cdot [G(\mathbf{a}_1, b_1, \mathbf{x}_s), \dots, G(\mathbf{a}_L, b_L, \mathbf{x}_s)]^T$.

(a) ϵ -SVR in boston (b) ϵ -ELM in boston (c) ϵ -SVR in concrete (d) ϵ -ELM in concrete (e) ϵ -SVR in space_ga (f) ϵ -ELM in space_ga

Figure 3.2: The relationship of testing RMSE to the parameter selection of ϵ -SVR (C, γ) and ϵ -ELM (C, L).

Chapter 4

Face Recognition Based on Extreme Learning Machines

Biometrics in a security system, usually to recognize the identity of a person from images of face, fingerprint or iris, is very important and widely studied in the area of computer vision and machine learning [68, 99]. Compared with the other two biometrics, face recognition is well known for easy implementation in everyday life. In various face recognition systems, common machine learning techniques to perform the task of recognition are nearest neighbor classifier (NN), support vector machines (SVMs) and neural networks. However, in the literature extreme learning machine (ELM) was seldom used in realization of face recognition although good generalization performance was shown on benchmarking datasets. This chapter presents one of my works to apply ELMs in face recognition together with the comparison with current popular classifiers like SVMs and NN.

Figure 4.1: The main procedures in a face recognition system.

4.1 Background

Generally speaking, a face recognition system is decomposed into two independent steps, see Figure 4.1. In the first step, a proper representation of the face images are required in an attempt to provide sufficient and efficient information on which the future classification is possible. The next step is the recognition step where training models are trained based on the chosen representation. There are mainly two categories of approaches to represent a face image [34, 37]: holistic-based and feature-based. The representative holistic-based methods are Eigenfaces [88] and Fisherfaces [100], where the whole face region is considered as the input. In the feature-based methods, such as active appearance model (AAM), usually local features of the eyes, nose, and mouth are extracted for collections of information such as the locations and local statistics.

In a face recognition system, dimensionality reduction methods are commonly used to relieve the sufferings from significant statistical redundancies existing in the face images [99]. The essence of such methods is to transform the image data from high dimensional space to low dimensional space so as to reduce redundancies and noises as well as to reveal the essential features of face images. Among the various dimensionality reduction algorithms [99, 37, 66], principal components analysis (PCA) [99] and linear discriminant analysis (LDA) [99] have been considered as the most classic approaches. According to PCA, the covariance matrix of input vectors can be decomposed so that some linear combination of the orthogonal basis actually represent the original input. Due to lacking of discriminative capability, PCA might not be a good choice for classification problems [84]. To improve the discriminative capability, information of class

label is incorporated with the aim to maximize between-class difference and minimize within-class difference. However, some potential problems [97] exist: linear projection in LDA is not able to reveal the nonlinear structure in the data; and problem of matrix singularity may happen. Discriminative locality alignment (DLA) [97] was proposed to overcome those problems. The solution provided by DLA is actually to focus on every training sample and preserve the relationship of its surrounding samples throughout the process. A brief review of the three dimensionality methods are given in the next section.

The final step of a face recognition system is of our special concern. Theoretically speaking, any machine learning technique with classification technique is a possible candidate. As mentioned before, nearest neighbor (NN) is the most common classifier in face recognition, such as Eigenfaces [88] and Fisherfaces [100]. An NN classifier simply selects the closest image in terms of geometric distance from the images pool of already registered identity. Another two popular machine learning techniques, Support vector machine (SVM) [75] and neural network [68] are also applicable. However, SVM is only designed for binary cases so that additional techniques are required for multiclass applications, with two popular algorithms being one-against-all SVM (SVM-OAA) [37] and one-against-one SVM (SVM-OAO) [4]. For instance, SVM-OAA consists of M binary SVMs for an application with M -categories. The task of every SVM is to distinguish between only one class of samples from the rest classes of samples. There are $M(M-1)/2$ binary SVMs in the structure of SVM-OAO to conduct pairwise classification among all the categories. According to the literature, SVM was successfully applied in combination with the subspace learning methods [96, 32, 98]. However, to the best of our knowledge, study of manifold learning (such as DLA) is quite limited till now, especially with supervised machine learning techniques like SVM. Furthermore, as a newly proposed machine learning tool, ELM for classification [52] has not been used in face recognition yet.

4.2 Brief on Dimensionality Reduction Methods

To study face recognition in a general case, holistic-based as well as feature-based representation of face images are used in this work. On the same time, the classic subspace learning PCA and LDA and manifold learning method DLA are all incorporated to reduce the dimensionality of the face representation. This section gives a brief introduction to PCA, LDA and DLA.

4.2.1 Principal component analysis

PCA [74, 99] has been widely used as one of the classic dimensionality reduction methods. Given an input vector $\mathbf{x}$ and its covariance matrix C , the eigen decomposition is shown as

$$C\Phi = \Phi\Lambda, \quad (4.1)$$

where Φ and Λ represent eigen vectors and eigen values of this equation respectively. In other words, $\mathbf{x}$ is approximated as the linear combinations of basis Φ_i obtained above with the coefficients $a_i, i = 1, \dots, n$:

$$\mathbf{x} = \sum_{i=1}^n a_i \Phi_i, \quad (4.2)$$

where n is the number of eigen vectors obtained from (4.1).

4.2.2 Linear discriminant analysis

In LDA, the following gives the definition of within-class scatter matrix S_w and between-class scatter matrix S_b , which actually is closely related to the label information

$$\begin{aligned} S_w &= \sum_{i=1}^M Pr(\omega_i) C_i \\ S_b &= \sum_{i=1}^M Pr(\omega_i) (\mathbf{m}_i - \mathbf{m}_0)(\mathbf{m}_i - \mathbf{m}_0)^T, \end{aligned} \quad (4.3)$$

where C_i is the covariance matrix and $Pr(\omega_i)$ is the prior class probability for input vectors belonging to class ω_i . $\mathbf{m}_i$ and $\mathbf{m}_0$ represent the average mean for input vectors from class ω_i and the global mean, respectively.

The ratio of the determinant of S_b and S_w are supposed to be maximized so as to distinguish samples from different classes. Therefore, the projection matrix W to transform original sample is computed as follows corresponding to the

$$S_b/S_w W = W \Lambda_W, \quad (4.4)$$

where Λ_W are eigen values of the generalized eigen problem. Hence, after projection, the data from the same class are closer and data from different classes are farther apart.

4.2.3 Discriminative locality alignment

As a linear dimensionality reduction method, DLA [97] was proposed to overcome the particular problems mentioned above in LDA.

Suppose that we are given a set of samples $X = [x_1, \dots, x_N] \in R^{m \times N}$, from M categories. For a given sample x_i , k_1 neighbor samples from the same class are randomly selected: $x_{i_1}, \dots, x_{i_{k_1}}$. Another k_2 nearest neighbors for x_i are randomly selected from samples in different classes: $x_{i_1}, \dots, x_{i_{k_2}}$. Then the local patch X_i for sample x_i is formed as follows:

$$X_i = \begin{bmatrix} x_i, x_{i_1}, \dots, x_{i_{k_1}}, x_{i_1}, \dots, x_{i_{k_2}} \end{bmatrix}$$

.

For each patch, after transformation, the local patch becomes

$$Y_i = \begin{bmatrix} y_i, y_{i_1}, \dots, y_{i_{k_1}}, y_{i_1}, \dots, y_{i_{k_2}} \end{bmatrix}$$

.

To make sure for every sample the transformed version to be close to samples from the same class and far away from the samples in different classes, mathematically it is shown as

$$\arg \min_{y_i} \left(\sum_{j=1}^{k_1} \|y_i - y_{i_j}\|^2 - C \sum_{p=1}^{k_2} \|y_i - y_{i_p}\|^2 \right), \quad (4.5)$$

where $C \in [0, 1]$ is adjusted by the user to decide the trade-off in importance between minimization of within-class distance and maximization of between-class distance. With definition

$$w_i = \begin{bmatrix} \overbrace{1, \dots, 1}^{k_1}, \overbrace{-C, \dots, -C}^{k_2} \end{bmatrix}^T$$

,

(4.5) then becomes:

$$\arg \min_{Y_i} \text{tr} (Y_i L_i Y_i^T), \quad (4.6)$$

$$\text{where } L_i = \begin{bmatrix} \sum_{j=1}^{k_1+k_2} (w_i)_j & -w_i^T \\ -w_i & \text{diag}(w_i) \end{bmatrix}.$$

The details of DLA can be found in [97]. Since the local patch of every sample is taken into consideration instead of the individual sample itself, the first problem in LDA as mentioned above is solved because not only the discriminative information but also the non-linearity of the sample distribution are preserved.

For each sample, the optimization for its local patch X_i ($i = 1, \dots, N$) is described as (4.6). To unify the optimizations of all the samples as a whole one, a new definition called *selection matrix* is introduced. For the i^{th} patch $Y_i = [y_i, y_{i^1}, \dots, y_{i^{k_1}}, y_{i_1}, \dots, y_{i_{k_2}}]$, $F_i = \{i, i^1, \dots, i^{k_1}, i_1, \dots, i_{k_2}\}$ is the corresponding index set, and the selection matrix S_i is:

$$(S_i)_{pq} = \begin{cases} 1 & \text{if } p = F_i\{q\} \\ 0 & \text{else} \end{cases}$$

,

where $S_i \in R^{N \times (k_1+k_2+1)}$, $p = 1, \dots, N$ and $q = 1, \dots, k_1 + k_2 + 1$. Thus, Y_i can be expressed as $Y S_i$, where $Y = [\mathbf{y}_1, \dots, \mathbf{y}_N]$ is the collection of the local patches for all the

N samples. Substituting it into (4.6), we have:

$$\arg \min_Y \sum_{i=1}^N \text{tr}(Y S_i L_i S_i^T Y^T) = \arg \min_Y \text{tr}(Y L Y^T), \quad (4.7)$$

where $L = \sum_{i=1}^N S_i L_i S_i^T \in R^{N \times N}$ is the alignment matrix.

By imposing $U^T U = I_d$, where U is the transformation matrix and I_d is a $d \times d$ identity matrix, (4.7) can be rewritten as:

$$\arg \min_U \text{tr}(U^T X L X^T U). \quad (4.8)$$

Finally the solution of (4.8) and the transformation matrix can be obtained by operating the standard eigen-decomposition of $X L X^T u = \lambda u$, which is free of matrix singularity problem compared with the eigen equation of LDA (because we do not need to compute the matrix inverse like in LDA).

4.3 Proposed Multi-class Classifiers Based on ELM in Face Recognition System

ELM for classification has been verified to outperform SVM as a binary classifier [52] in terms of generalization performance as well as convenience in implementation. Therefore, multi-class classifiers based on optimization based ELM are proposed in this work, with the strategy of OAA and OAO. The proposed classifiers are tested in a face recognition system. Without loss of generality, the proposed classifiers are compared with NN, SVM-OAA and SVM-OAO in different systems where the face representation ap-

proaches are varied including both holistic-based method and feature-based method.

Seen from (2.26), optimization method based ELM is designed as the binary classifier. In multiclass applications, various techniques exist to allow usage of binary classifiers. Among those methods, one-against-all (OAA) [37] and one-against-one (OAO) [4] are widely used.

Seen from equation (2.28) the decision function of ELM for classification is:

$$f(\mathbf{x}) = \text{sign} \left(\sum_{s=1}^{N_s} \alpha_s t_s K_{\text{ELM}}(\mathbf{x}, \mathbf{x}_s) \right) \quad (4.9)$$

Given training data from M categories, OAA is made up of M binary classifiers with each one trained to distinguish one class and all the other ones. As new data with unknown label comes, there is a competition among all the basic classifiers. The rule is that one with the largest output of decision function wins the competition and the prize is the capability in determining the label of the sample [89]. According to the literature, OAA appears to be the most popular technique in constructing multi-class SVM classifier [96, 32, 98].

In OAO, every binary classifier is assigned the task to distinguish between one pair of classes, resulting in $(M-1)*M/2$ classifiers in total. Within every classifier, the sign of the output of decision function indicates which class in that particular pair has been chosen. When a testing data arrives, every basic classifier provides the preference of a class over another. The class label chosen by maximum numbers of classifiers wins the competition in the case of OAO.

4.4 Performance Evaluation

In this section the performance of ELM-OAA and ELM-OAO in a face recognition system with both holistic learning method and feature-based learning method (AAM) on four benchmarking face databases is validated. The comparison has been made against three other popular classifiers: nearest neighbor (NN) and SVM-OAA/SVM-OAO.

4.4.1 Face databases

In holistic learning three databases are tested: YALE [8], ORL¹, UMIST [33]. The YALE database contains images from 15 individuals with 11 images for each individual. The images vary in illumination, facial expression, and decorations e.g. when a person with or without glasses. 40 persons contribute to the ORL database with 10 images per person. Different illuminations, and expressions like smile, eye closing etc can be found in the image series. 564 face images from 20 persons are included in the UMIST database. The participants, with different races are requested to take photos from profile to frontal view.

Different from the previous three databases, BioID² database has large variation on illumination, background, and even face sizes. With the annotation on facial images³ and AAM [20], feature representation about shape, appearance, and texture of each image is able to be obtained. In the case of database BioID, feature vector with dimensionality 481 is obtained for each image using tools above.

For YALE database, the images are in the size of 40×40 pixels; for UMIST and ORL, the size of face images is 32×32 with 256 grey levels representing each pixel. And before training, the grey level of each image is normalized into the range of $[0, 1]$.

Figure 4.2 displays the sample images from Yale, ORL and UMIST, which we use in

¹<http://www.uk.research.att.com/facedatabase.html>

²<http://www.bioid.com/support/downloads/software/bioid-face-database.html>

³<http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/>

Figure 4.2: Face samples: Row 1 for YALE, Row 2 for ORL, Row 3 for UMIST.

Figure 4.3: Face samples from BioID database.

holistic learning . Sample images from BioID are shown in Figure 4.3.

4.4.2 Experimental results

For YALE, ORL, UMIST, 3, 5 and 7 images are randomly selected from each individual for training to avoid the bias on any particular person. The remainders within every person are used for testing. On the other hand, 2, 3, 4 and 5 images are randomly selected from each person in BioID for training.

Speaking of the classifiers in the experiments, no parameter needs to be tuned in the NN classifier. As mentioned in [52], for SVM with RBF kernel: $K_{SVR}(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma|\mathbf{x} - \mathbf{x}_i|^2)$, 225 different pairs of (C, γ) are tried with 15 different values of C (0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 1000, 10000) and 15 different values of γ (0.0001, 0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10, 20, 100, 1000, 10000). While for ELM based classifiers with sigmoid random nodes: $G(\mathbf{a}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b)))$, 15 different values of C (0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 1000, 10000) and 10 different values of L (20, 30, 50, 80, 100, 200, 500, 1000, 2000, 3000) have been tried, resulting in 150 different pairs in total.

Speaking of DLA, four parameters are tuned: the projected dimensionality (P -dim),

Table 4.1: The optimal parameter (P -dim, k_1 , k_2 , C_{DLA}) of DLA in different systems.

	NN	SVM-OAA	SVM-OAO	ELM-OAA	ELM-OAO
UMIST (7)	(33,4,5,0.1)	(25,5,4,0.1)	(21,5,1,0.1)	(26,5,2,0.1)	(26,5,2,0.1)
UMIST (5)	(10,3,4,0.1)	(24,3,3,0.1)	(31,2,1,0.1)	(28,3,3,0.1)	(28,3,3,0.1)
UMIST (3)	(18,2,1,0.1)	(21,2,1,0.1)	(18,2,1,0.1)	(18,2,1,0.1)	(18,2,1,0.1)
ORL (7)	(35,2,4,0.3)	(40,3,4,0.1)	(45,3,8,0.1)	(45,3,3,0.1)	(45,3,3,0.1)
ORL (5)	(30,3,2,0.8)	(40,4,4,0.1)	(40,3,8,0.1)	(39,4,2,0.1)	(39,4,2,0.5)
ORL (3)	(30,2,4,0.1)	(30,2,4,0.1)	(36,2,5,0.1)	(32,2,5,0.1)	(32,2,5,0.1)
YALE (7)	(15,3,5,0.1)	(14,3,4,0.1)	(15,4,2,0.3)	(15,3,4,0.1)	(15,3,4,0.1)
YALE (5)	(30,3,4,0.1)	(14,3,3,0.1)	(14,3,3,0.1)	(14,3,6,0.1)	(14,3,3,0.1)
YALE (3)	(18,2,1,0.1)	(14,2,1,0.3)	(14,2,1,0.3)	(14,2,1,0.5)	(14,2,1,0.1)
BioID (5)	(21,4,5,0.1)	(25,4,7,0.1)	(25,3,7,0.1)	(25,4,6,0.1)	(25,4,4,0.3)
BioID (4)	(21,3,2,0.1)	(25,3,3,0.1)	(25,3,3,0.1)	(30,3,3,0.1)	(25,3,3,0.1)
BioID (3)	(21,2,2,0.1)	(25,2,3,0.1)	(25,2,1,0.1)	(27,2,2,0.1)	(27,2,3,0.1)
BioID (2)	(21,1,1,0.1)	(25,1,3,0.1)	(25,1,3,0.1)	(27,1,3,0.1)	(25,1,3,0.1)

k_1 which represents the number of samples selected from the same class for a local patch, k_2 which represents the number of samples selected from different classes for that local patch, and C_{DLA} which determines the trade-off of within-class and between-class importance [97]. The final choice of the parameters is listed in Table 4.1.

The recognition result is recorded as an average result of 20 trials. All the experiments are conducted in MATLAB 7.1 running in a PC with Pentium 4, 2.99GHZ CPU.

In order to show how the dimensionality reduction methods influence the data distribution and which one suitable for further recognition, two-dimensional representation of the images of 5 selected individuals from YALE are illustrated in Figure 4.4, before and after PCA, LDA and DLA projection respectively. There are 5 training samples from every individual, each represented by one point of particular shape. Different number other than 5 of the images for every person might be selected, which should result in similar figures as Figure 4.4. The discriminative capability of LDA projection is obvious, since images from one person are well separated from the others. However, images from the same person are too close to distinguish. As an improvement on LDA, it can be found from DLM projection that the superposition of images from the same person is greatly reduced. Note that, the number listed in the bracket after each representation method indicates number of images from every individual are randomly selected for

Figure 4.4: Two-dimensional representation of the images of selected individuals from YALE.

training. For instance, LDA(5) in Figure 4.4(b) indicated the set of experiments where 5 images are selected from every individual for training with LDA method.

Table 4.2 to Table 4.5 shows the recognition result of five classifiers in terms of classification rate. Note that to give a brief impression on the size of the support vector network, number of support vectors for SVM-based classifiers are provided as well.

The comparison of the recognition result of the three OAA-based classifiers are given in Figure 4.5. The case on OAO based methods can be found quite similar. The training time of SVM and ELM with both OAA and OAO are compared and displayed in Figure 4.6 and Figure 4.7. How selection of parameters can affect the recognition result is shown in Figure 4.8 on the case of SVM-OAA and SVM-OAO with an example of 3 images per person from YALE database. The figure only includes result of PCA projection and cases on LDA or DLA are supposed to be similar.

YALE (P -dim)	NN		OAA Or OAO	SVM						ELM				
	Testing Rate(%)	Testing Dev(%)		Testing Rate(%)	Testing Dev(%)	Training Time(s)	C	γ	# SVs	Testing Rate(%)	Testing Dev(%)	Training Time(s)	C	L
PCA (7) (105)	60.50	5.41	OAA	81.83	4.98	0.75	10^4	10	418	82.08	4.25	2.21	50	1000
			OAO	76.00	4.57	2.30	10^4	10	1163	76.50	3.97	2.15	2	1000
PCA (5) (75)	56.33	3.77	OAA	76.61	4.64	0.67	10^4	20	335	76.50	4.13	1.41	1000	1000
			OAO	69.89	3.96	2.05	10^4	10	872	69.50	4.01	1.63	20	1000
PCA (3) (45)	48.46	3.99	OAA	66.83	4.29	0.50	10^3	5	251	66.79	4.56	0.48	5	1000
			OAO	58.92	4.45	1.64	10^3	5	594	58.71	5.05	1.02	100	1000
LDA (7) (14)	83.83	6.21	OAA	85.33	4.61	0.52	10^{-3}	0.1	326	83.75	4.95	1.24	10	1000
			OAO	85.00	5.97	2.10	1	0.2	1045	84.67	5.50	2.36	2	1000
LDA (5) (14)	78.89	3.69	OAA	80.89	3.67	0.48	10^{-3}	0.1	358	79.78	4.53	1.07	2	1000
			OAO	79.61	3.33	1.72	1	0.1	653	79.33	3.94	2.04	2	1000
LDA (3) (14)	64.33	4.77	OAA	68.00	4.30	0.50	1	0.2	239	66.04	4.47	0.41	2	1000
			OAO	65.08	5.41	1.53	2	0.2	522	64.67	5.59	0.68	2	1000
DLA (7)	87.25	3.68	OAA	89.00	3.03	1.13	10^4	10	156	89.04	2.70	1.68	5	1000
			OAO	89.17	2.73	3.25	10^4	0.1	616	88.67	3.04	2.02	2	1000
DLA (5)	82.00	3.94	OAA	83.61	3.12	1.28	10^4	0.1	323	83.63	2.91	1.74	1	1000
			OAO	82.67	3.91	5.56	10^4	0.1	530	83.22	3.38	1.44	1	1000
DLA (3)	70.37	3.79	OAA	72.54	3.17	1.04	10^4	0.1	409	72.50	4.12	0.78	2	1000
			OAO	71.21	3.86	4.60	10^4	0.1	573	70.71	4.16	0.85	20	1000

Table 4.2: Comparison between the NN, SVM-OAA, SVM-OAO, ELM-OAA and ELM-OAO on database YALE.

UMIST (<i>P</i> -dim)	NN		OAA Or OAO	SVM						ELM				
	Testing Rate(%)	Testing Dev(%)		Testing Rate(%)	Testing Dev(%)	Training Time(s)	<i>C</i>	γ	# SVs	Testing Rate(%)	Testing Dev(%)	Training Time(s)	<i>C</i>	<i>L</i>
PCA (7) (140)	88.91	2.54	OAA	93.15	2.18	1.49	50	0.8	588	92.55	2.09	2.09	10	1000
			OAO	93.32	2.53	7.12	50	2	1980	93.49	2.25	3.19	100	1000
PCA (5) (100)	80.84	3.51	OAA	87.66	3.05	0.97	50	0.8	463	87.33	2.90	1.34	50	1000
			OAO	86.87	3.05	7.23	10^4	10	1570	87.27	2.41	2.44	100	1000
PCA (3) (60)	67.53	3.36	OAA	75.26	3.90	0.38	50	1	343	75.21	3.81	0.67	10^3	1000
			OAO	73.49	4.54	5.44	10^4	10	1088	73.70	3.45	1.52	50	1000
LDA (7) (19)	92.05	2.30	OAA	92.01	2.32	1.24	0.5	0.1	451	91.28	2.41	1.29	5	1000
			OAO	92.13	2.30	3.27	5	0.1	637	92.35	2.82	1.15	50	1000
LDA (5) (19)	86.67	2.51	OAA	87.35	2.54	0.70	0.2	0.1	493	85.92	2.86	1.14	1	1000
			OAO	86.87	2.48	4.73	1	0.2	1534	86.76	1.99	1.06	100	1000
LDA (3) (19)	75.37	4.60	OAA	75.91	4.58	0.62	0.5	0.1	752	74.49	4.62	0.53	10	1000
			OAO	75.52	4.43	5.22	50	0.4	742	75.74	3.87	1.03	10^3	1000
DLA (7)	96.09	1.50	OAA	96.22	1.35	3.18	10^{-3}	0.1	551	94.66	1.84	4.45	2	1000
			OAO	96.01	1.43	6.18	10^4	1	414	96.17	1.65	1.23	10	1000
DLA (5)	91.76	2.22	OAA	92.24	2.39	1.56	10^{-3}	0.1	605	89.96	2.48	1.12	1	1000
			OAO	91.93	2.33	6.18	10^4	10	453	91.88	2.22	1.08	10	1000
DLA (3)	80.85	3.88	OAA	81.52	3.78	0.51	10^{-3}	0.1	518	78.55	3.96	2.02	2	1000
			OAO	80.85	3.90	6.02	10^3	0.1	948	80.77	3.93	0.93	10	1000

Table 4.3: Comparison between the NN, SVM-OAA, SVM-OAO, ELM-OAA and ELM-OAO on database UMIST.

ORL (<i>P</i> -dim)	NN		OAA Or OAO	SVM						ELM				
	Testing Rate(%)	Testing Dev(%)		Testing Rate(%)	Testing Dev(%)	Training Time(s)	<i>C</i>	γ	# SVs	Testing Rate(%)	Testing Dev(%)	Training Time(s)	<i>C</i>	<i>L</i>
PCA (7) (280)	91.67	2.37	OAA	97.46	1.39	23.2	10	0.4	1484	97.11	1.63	6.38	10^3	1000
			OAO	96.67	1.79	65.8	10	0.4	8509	96.28	2.42	20.1	5	1000
PCA (5) (200)	86.25	1.57	OAA	94.65	1.86	3.14	50	1	1158	94.97	1.47	3.84	20	1000
			OAO	92.90	1.82	52.5	50	0.8	6459	92.40	1.78	14.7	5	1000
PCA (3) (120)	77.29	3.31	OAA	88.09	2.49	2.72	10	0.4	1084	88.29	2.35	2.04	5	1000
			OAO	83.59	3.26	28.8	50	0.8	4396	83.57	3.09	10.4	10	1000
LDA (7) (39)	95.50	1.63	OAA	97.04	1.22	153	10	10^4	11200	96.46	1.38	7.68	5	1000
			OAO	95.87	1.82	21.3	50	0.8	8362	95.62	1.60	8.05	5	1000
LDA (5) (39)	92.77	1.97	OAA	94.57	1.96	24.7	10	10^4	8000	93.93	2.23	9.28	1	1000
			OAO	93.15	2.13	18.8	10	0.2	3364	92.75	1.78	5.98	10	1000
LDA (3) (39)	84.59	3.74	OAA	88.09	2.95	1.20	10	0.4	967	87.50	3.12	2.74	20	1000
			OAO	84.09	3.40	20.2	50	0.8	3914	84.02	3.80	4.21	50	1000
DLA (7)	97.62	1.49	OAA	98.58	1.33	0.55	10^4	0.1	3652	98.44	1.17	4.54	5	1000
			OAO	98.50	1.29	86.6	10^4	10	3637	98.54	1.57	5.91	50	1000
DLA (5)	96.20	1.75	OAA	97.20	1.49	1.18	10^4	0.1	2552	96.73	1.21	1.52	10	1000
			OAO	96.80	1.62	12.5	10^4	0.1	4055	96.65	1.71	26.4	10	1000
DLA (3)	90.14	2.58	OAA	91.38	2.93	2.01	10^4	0.1	3057	90.96	2.93	1.55	10	1000
			OAO	88.70	3.37	10.9	10^4	1	2154	89.80	2.73	4.73	20	1000

Table 4.4: Comparison between the NN, SVM-OAA, SVM-OAO, ELM-OAA and ELM-OAO on database ORL.

BioID (<i>P</i> -dim)	NN		OAA Or OAO	SVM						ELM				
	Testing Rate(%)	Testing Dev(%)		Testing Rate(%)	Testing Dev(%)	Training Time(s)	<i>C</i>	γ	# SVs	Testing Rate(%)	Testing Dev(%)	Training Time(s)	<i>C</i>	<i>L</i>
PCA (5) (110)	92.64	1.36	OAA	97.92	0.78	0.71	10 ³	20	419	98.35	0.77	2.09	10	1000
			OAO	95.73	1.20	4.09	50	5	162	95.62	1.24	1.51	0.2	1000
PCA (4) (88)	90.81	1.88	OAA	96.64	1.46	0.53	10 ⁴	20	384	96.58	1.66	1.54	50	1000
			OAO	93.79	1.93	3.61	50	5	141	93.71	1.99	1.22	10 ³	1000
PCA (3) (66)	88.01	2.55	OAA	95.61	1.75	0.46	10 ³	20	341	94.91	1.96	1.05	5	1000
			OAO	91.35	2.62	3.21	10 ³	20	116	91.43	2.56	0.97	0.2	1000
PCA (2) (44)	81.81	2.73	OAA	90.74	2.49	0.37	10 ⁴	20	284	90.17	1.66	0.64	0.2	1000
			OAO	84.43	3.27	2.90	10 ³	20	85	84.54	3.28	0.66	10 ³	1000
LDA (5) (21)	98.68	0.63	OAA	98.81	0.54	0.63	2	0.4	486	98.78	0.57	0.49	10 ⁴	1000
			OAO	98.69	0.5	3.38	20	2	93	98.67	0.55	0.87	0.2	1000
LDA (4) (21)	98.48	0.77	OAA	98.52	0.86	0.52	5	1	400	98.44	0.92	0.40	5	1000
			OAO	98.38	0.89	3.60	0.5	0.4	148	98.34	0.88	0.64	10	1000
LDA (3) (21)	97.04	1.57	OAA	97.58	1.18	0.47	10	2	370	97.55	1.10	0.30	0.5	1000
			OAO	97.34	1.11	3.34	0.5	0.4	132	97.22	1.05	0.59	50	1000
LDA (2) (21)	88.34	2.60	OAA	90.91	1.84	0.37	50	1	342	90.63	2.04	0.20	0.5	1000
			OAO	90.63	1.61	2.83	50	5	83	90.50	1.76	0.54	10 ⁴	1000
DLA (5)	98.91	0.37	OAA	99.17	0.31	0.50	10 ⁴	20	242	99.17	0.30	0.77	100	1000
			OAO	98.83	0.34	3.30	10 ⁴	0.4	1755	98.90	0.45	2.16	100	1000
DLA (4)	98.21	1.12	OAA	98.72	0.88	0.45	50	2	252	98.92	0.85	0.66	5	1000
			OAO	98.24	1.10	3.23	10 ⁴	20	800	98.24	1.13	1.98	2	1000
DLA (3)	96.93	1.19	OAA	97.65	1.25	0.46	10 ³	20	266	97.78	1.20	0.53	50	1000
			OAO	96.95	1.30	3.04	10 ⁴	0.4	1072	97.06	1.16	1.80	20	1000
DLA (2)	92.36	2.89	OAA	94.41	2.06	0.38	100	5	236	94.77	2.21	0.41	100	1000
			OAO	92.11	2.52	2.83	10 ⁴	2	567	92.45	2.49	1.61	2	1000

Table 4.5: Comparison between the NN, SVM-OAA, SVM-OAO, ELM-OAA and ELM-OAO on database BioID.

Figure 4.5: Comparison of the generalization performance of NN, SVM-OAA and ELM-OAA on YALE.

Figure 4.6: Comparison of the training time of SVM-OAA and ELM-OAA on YALE.

Figure 4.7: Comparison of the training time of SVM-OAO and ELM-OAO on YALE.

(a) PCA+SVM-OAA

(b) PCA+SVM-OAO

(c) PCA+ELM-OAA

(d) PCA+ELM-OAO

Figure 4.8: The effect of tunable parameters ((C, γ) for SVM and (C, L) for ELM) on the generalization performance in holistic learning with 3 training samples from YALE.

Figure 4.9: The recognition rate for each person from database YALE using ELM-OAA.

Figure 4.10: Row 1: images of Identity 3 from YALE; Row 2: images of Identity 11 from YALE.

4.4.3 Discussions

It is not difficult to conclude from Table 4.2 to Table 4.5, and Figure 4.5 that both SVM and ELM classifier outperform NN classifier, especially when dimensionality reduction method without discriminative capability (such as PCA) is used or when limited number of images are available. However, among the three dimensionality reduction methods, face recognition system with DLA usually achieves best performance.

It is also found that performance of optimization based ELM is comparable to that of SVM in terms of classification accuracy and deviation. Speaking of classification deviation, it is necessary to mention YALE, which results in largest deviation compared with the other datasets due to various illumination and facial expressions in the face images.

As far as OAA and OAO are concerned, it is shown from all the datasets that, OAA based classifiers has better performance than the corresponding OAO based classifiers. When it comes to the aspect of training time of ELM, which has much to do with number of hidden nodes, is found quite close to SVM when it is set as 1000. Although an default setting on number of hidden nodes is given, promising performance can be obtained when it is attentively tuned.

From the parameter sensitivity test given by Figure 4.8, ELM with Sigmoid random node is found insensitive to user specified parameters. Moreover, good performance is obtained normally when large L is given. While the sensitivity of parameters in SVM is very so high that the choice of parameters should be carefully tuned, especially on the case of OAO. More precisely, two parameters need to be tuned simultaneously. Therefore, the conclusion is that ELM is a better choice when computational convenience is of special concern since only one parameter needs to be tuned.

An insightful study on the basis of every person is conducted on YALE as shown in Figure 4.9, where the 6 plot lines indicate the combination of different number of training samples per person and different dimensionality reduction methods used. Similar figure can be obtained for the SVM case. It is interesting to find that some identities (Identity

11) tend to be recognized more easily no matter of what configurations of the recognition system are. On the other hand, some identities (Identity 3) are naturally difficult to be recognized. However, the performance of such vulnerable identities can be improved significantly as long as configuration of the system is properly made. The possible reason of this tendency on each individual can be found from the displayed samples of Identity 3 and 11 (Figure 4.10). Compared with easily recognized Identity 11, the images of Identity 3 are found with relatively large variations in facial expressions and illuminations.

4.5 Extension of Work: Face recognition with Kernelized ELM

In the previous work, optimization based ELM was demonstrated as a promising classifier to recognize identities from face images. Compared with another popularly used technique SVM, ELM using optimization method provides comparable classification performance yet with much easier parameter tuning process.

However, both SVM and optimization method based ELM are naturally suitable for problems of only binary type. That is why techniques such as OAA [37] and OAO [4] are needed to decompose the multiclass problem into binary sub-problems. However, such techniques have posed a potential problem on aspects of algorithm complexity and computational complexity, especially when there are quite a number of identities the face images belong to.

Another problem about quadratic programming tool used in the standard optimization method is that, the computational complexity is at least quadratic of the number of training samples, which indeed causes slow recognition when there are huge number of training data. Therefore, least square support vector machine (LS-SVM) [83] was proposed as an easier and faster implementation of SVM, where SVM loss function was

replaced by least loss function, inequality optimization constraints in original SVM are replaced by equality equations, and least square method in lieu of quadratic programming tool is utilized. Hence, LS-SVM is actually computationally more efficient than SVM.

4.5.1 Kernelized ELM

As mentioned in Chapter 2, one of the latest work of ELM [55] has found that ELM actually provides a unified solution which generalizes to neural networks, support vector networks, and regularized networks. But this time least square method as in LS-SVM and conventional ELM in lieu of the standard optimization method as in SVM and optimization method based ELM is used to realize the implementation. The future mapping of ELM in the work of [55] can be conventional ELM random nodes as well as kernel functions. To some extent, kernelized ELM actually simplifies the implementation of LS-SVM. Details of the kernelized ELM in the unified ELM framework can be found in Section 2.3.

By utilizing the kernelized ELM, the two aforementioned problems are solved. Because the ELM framework can be applied in multiclass problems directly. And the computational cost is relieved by using least square method instead of the quadratic programming tool. Therefore, as an extensive work, the kernelized ELM [55] is proposed as the classifier in face recognition. The performance is compared with LS-SVM over the benchmarking face data set YALE.

Note that, LS-SVM is also a binary classifier. Four coding schemes were proposed for LS-SVM to transform the original problem into binary subproblems. These coding schemes are namely minimum output coding (MOC), one against all (OAA), one against one (OAO) and error correcting output coding (ECOC). That is to say, multiple binary classifiers are still needed in the implementation. In contrast to SVM, LS-SVM and even optimization method based ELM, single ELM classifier is enough to carry out

multi-category classification tasks.

4.5.2 Performance evaluation

The performances of LS-SVM and ELM are evaluated on the benchmarking face dataset YALE [8]. YALE consists of 15 identities with an average of 11 images per person. A large variation on illumination, facial expressions, and decorations are posed on the face images. Images are in the size of 40×40 pixels of grey value. Therefore, the resulting face vectors are of dimension 1600, which are normalized into the range of $[0, 1]$. 3, 5 and 7 images are randomly selected from each person for training. Figure 4.11 exhibits some sample images from YALE. The eigen vectors of PCA projections are all kept for the sake of convenience thus to result in the dimension of 45, 75 or 105 corresponding to 3, 5 or 7 images per person, respectively. The dimension of the image vector is reduced to 14 after LDA transformation.

The simulations are conducted in MATLAB 7.0.1 environment running on a Core 2 Quad, 2.66G CPU with 2G RAM.

Figure 4.11: Display of sample images from database YALE.

Table 4.6: Comparison of performance of LS-SVM and ELM on database YALE.

YALE	LS-SVM (Gaussian Kernel)				ELM (Gaussian Kernel)			
	Test Rate (%)	Test Dev (%)	Train Time (s)	(C, γ)	Test Rate (%)	Test Dev (%)	Train Time (s)	(C, γ)
PCA(3)	57.54	4.73	0.0130	$(2^4, 2^{10})$	69.25	3.72	0.0008	$(2^4, 2^{11})$
PCA(5)	68.72	4.32	0.0149	$(2^{10}, 2^{20})$	80.80	3.32	0.0008	$(2^5, 2^{10})$
PCA(7)	74.58	4.08	0.0203	$(2^{10}, 2^{16})$	85.58	3.35	0.0039	$(2^9, 2^{13})$
LDA(3)	65.25	5.52	0.0130	$(2^{-4}, 2^6)$	67.83	4.81	0.0015	$(2^5, 2^4)$
LDA(5)	79.61	3.50	0.0164	$(2^{-4}, 2^4)$	81.28	3.65	0.0008	$(2^{-4}, 2^{-1})$
LDA(7)	84.83	5.97	0.0218	$(2^{10}, 2^{12})$	85.92	4.31	0.0023	$(2^2, 2^{-7})$

Parameter Settings

Both LS-SVM and ELM adopt the popular Gaussian kernel function $K(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma|\mathbf{x} - \mathbf{x}_i|^2)$. A grid search on $\{2^{-24}, 2^{-23}, \dots, 2^{24}, 2^{25}\}$ have been conducted for regularization parameter C and kernel parameter γ when seeking for the optimal performance result.

Recognition Accuracy

An average result of testing accuracy based on 20 trials are given in Table 4.6, where the proposed kernelized ELM are compared with LS-SVM. Note that PCA (5) in the table actually means 5 images are randomly selected from each person for training. As expected, the better result of recognition accuracy actually reflects the ELM theory where solutions of SVM/LS-SVM are the sub-optimal solutions of ELM. It is also observable that system with unsupervised method such as PCA relies more on the capability of the classifier. Moreover, larger standard deviation are detected in the result of LS-SVM caused by 20 random train-test splittings, which implies that kernelized ELM tends to be more stable in the performance.

Moreover, the comparison can be made with SVM, LS-SVM with Gaussian kernel and optimization based ELM with Sigmoid random nodes shown in Table 4.2 in the previous stage of work. The advantage of kernelized ELM over all three aforementioned algorithms is obvious in terms of recognition rate.

Computational Complexity and Training Time

As mentioned above, LS-SVM has to rely on 4 different coding schemes in order to be applied in multiclass problems, all of which are tested in the experiments. Among them, OAO has shown much better recognition performance than the other three. In the coding scheme of OAO, the output label (scale) is extended to a label vector of dimension $(M - 1) * M/2$, which requires $(M - 1) * M/2$ binary LS-SVM classifiers in training.

The computational complexity of such coding scheme is actually comparable to OAO technique used in constructing binary SVMs. Take YALE for example, a total of 105 binary classifiers are required in the architecture. Hence, the computational complexity of such coding scheme used in LS-SVM is quite high. on the other hand, a single ELM classifier is enough for multiclass classification tasks. Simply, the number of the output nodes of the SLFN is set the same as the number of classes, which is 15 for database YALE. The convenience of ELM also results in less training time compared to LS-SVM shown in Table 4.6.

Sensitivity of User Specified Parameters

Usually the generalization performance of SVM/LS-SVM is sensitive in parameter selection. As well known that insensitive parameters are wanted as in the ELM framework with random nodes since the parameter tuning process can be much easier. The sensitivity of user specified parameters in both LS-SVM and ELM is studied as shown in Figure 4.12. It is observable that the generalization performance of both LS-SVM and ELM is sensitive to user specified parameters (C, γ) , where C is the cost parameter and γ is the kernel parameter. This finding is consistent with that in [55] with kernel version tested on benchmarking datasets of regression and classification problems.

4.6 Summary

In this work, ELM-OAA and ELM-OAO are proposed as the classifiers in a face recognition system. Without loss of generality, the performance is compared with SVM-OAA and SVM-OAO in a variety of systems with holistic learning or feature-based learning, using either subspace learning or manifold learning. With comparable recognition performance and training time, optimization method based ELM exceeds SVM as far as the convenience in parameter selection is concerned. In ELM, Only one parameter (C) needs to be tuned after another one (L) is assigned with a large value.

(a) PCA(5)+LS-SVM

(b) LDA(5)+LS-SVM

(c) PCA(5)+ELM

(d) LDA(5)+ELM

Figure 4.12: Testing accuracy with respect to various parameter selections of (C, γ) for LS-SVM and ELM with Gaussian kernels.

After optimization method based ELM, kernelized ELM in the unified framework was applied in face recognition in the extensive work. In some way, kernelized ELM can be considered as the simplified version of LS-SVM, where bias b in the hypothesis is eliminated. It was argued that unified ELM framework is more computationally efficient than SVM, LS-SVM and even optimization method based ELM. The experiment result on database YALE has shown that kernelized ELM is able to perform better than SVM, LS-SVM and optimization based ELM with Sigmoid random nodes in terms of recognition capability and computational complexity. Reasonably speaking, the kernelized ELM is with promising potential to be applied in other biometrics, such as fingerprint and iris recognition.

The application in face recognition can be viewed as a verification of the universal classification capability of unified ELM framework proved in [55]. However, the unified ELM framework is not only proved with classification capability, but also with universal approximation capability. In the following work, a popular real world application is of our special interest, namely relevance ranking, where machine learning techniques are used to approximate the complex unknown relationship between document and relevance degree.

Chapter 5

Learning to Rank with Extreme Learning Machine

5.1 Introduction

In recent years, relevance ranking has been a popular topic in fields such as document retrieval, collaborative filtering, key term extraction, sentiment analysis, etc. Due to the discriminative capability, machine learning techniques are widely adopted in relevance ranking, namely, "learning-to-rank". As in most of the machine learning algorithms, the framework of a learning-to-rank system usually consists of a learning system and a ranking system. In the stage of learning, a model is built based on the training data, which is capable of ranking the testing data with unknown relevance degree in the following ranking stage.

In literature, learning-to-rank algorithms can be categorized into three types: pointwise [22, 65, 24, 82], pairwise [38, 31, 14, 101, 17, 16, 91] and listwise [18, 92, 93, 95, 85]. Ranking is perceived as either a regression problem or classification problem in pointwise approaches, where the relationship between the feature (contain information of both query and the document) and target label (relevance judgment) is modeled. How-

ever, pairwise ranking is claimed to be more consistent with the nature of ranking, where the pairwise comparison between the documents instead of the pointwise approximation of document-label relationship is considered.

Representative pointwise approaches include subset ranking using regression [23], where relevance judgment is treated as the real value so that the hypothesis is approximated by a regression algorithm; binary/multiclass classification model is used to predict the ranking function [70, 65] by treating the relevance judgment as the class label and adopting proper loss function, such as hinge loss for support vector machine (SVM) [89] and least square loss for least square SVM (LS-SVM) [83].

In lieu of sample-label relationship approximation, the ranking function is searched in pairwise approaches to reveal the pairwise preference between two samples. A number of machine learning techniques are applied in modeling. For instance, neural network with gradient descent is used to minimize the cross-entropy loss in RankNet [14]; the same modeling technique to optimize fidelity loss in FRank [87]; boosting algorithms are also validated with exponential loss [31]; in RankSVM [61], the ranking function is searched to minimize the hinge loss function; and finally regularized least squares are used to optimize the least square loss function in [72]. However, extreme learning machine [57, 51, 47, 49, 52, 55], as a new powerful machine learning technique, is quite new to the field of relevance ranking.

To the best of our knowledge, there is no work of applying ELM in relevance ranking so far. Therefore, in this paper, ELM is studied as a learning-to-rank algorithm from the perspective of both pointwise and pairwise.

5.2 Pointwise ELM

In pointwise approaches, ranking problem is usually formulated as a traditional regression, classification or ordinal regression model. Features of the training samples are ex-

tracted from both the query and document. And the relevance judgment of each sample can be interpreted as the output value in regression models or class label in classification models.

Suppose there are n queries, associated with query q_i being m_i number of documents. The training samples are represented as $(\mathbf{x}_j^i, y_j^i)$, $i = 1, \dots, n$; $j = 1, \dots, m(i)$, where $\mathbf{x}_j^i$ is the feature extracted from the i th query and j th document with y_j^i indicating the relevance degree between them. To model the relationship between query-document pair $\mathbf{x}_j^i$ and their relevance degree, ELM is naturally capable of such task.

The mathematical model of the SLFNs can be represented as

$$\mathbf{H}\beta = \mathbf{T}, \quad (5.1)$$

where $\mathbf{H}$ is the hidden layer output matrix, β is the output weight, and $\mathbf{T} = [y_1^1, \dots, y_{m(1)}^1, \dots, y_{m(n)}^n]^T$ is the target vector

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1^1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_{m(1)}^1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_{m(n)}^n) \end{bmatrix}. \quad (5.2)$$

To minimize the least square loss function of the training error and regulation term $\|\beta\|$,

$$\text{Minimize: } \|\mathbf{H}\beta - \mathbf{T}\|^2 \text{ and } \|\beta\|, \quad (5.3)$$

the solution of ELM is as follows

$$\begin{aligned} \text{When } N < L: \beta &= \hat{\mathbf{H}}^\dagger \mathbf{T} = \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \\ \text{When } L < N: \beta &= \hat{\mathbf{H}}^\dagger \mathbf{T} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}, \end{aligned} \quad (5.4)$$

where L is number of hidden nodes in the SLFN and N is the number of total training samples and C is the regularization term specified by users. The two formulas in (5.4) can be flexibly chosen given different circumstances. When there are relatively fewer training samples, the upper one is computational efficient since it only needs to calculate the matrix inversion of the training sample size. On the other hand, the lower one is recommended when given quite large dataset size. Therefore, the lower formula is adopted in the implementation

Algorithm 1 Pointwise Ranking ELM Algorithm

Given: Training sample pairs $(\mathbf{x}_j^i, y_j^i)$, number of hidden nodes L , regularization term C , and testing sample $\mathbf{x}$,

Linear Random Node:

- 1: Generate L hidden nodes with the parameters $(\mathbf{a}, b)$ of each node randomly assigned;
- 2: Calculate the hidden layer output matrix $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1^1), \dots]^T$ as shown in (5.2);
- 3: Calculate the output weight β as shown in (5.4) and the output function of testing sample as $f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \cdot \beta$.

Linear Kernel:

- 4: Calculate the output function of testing sample as shown in (5.8) and (5.9).
-

5.3 Linear Random Node and Linear Kernel

According to [55], ELM actually provides a unified solution for the generalized SLFN, which extends from traditional neural network to regularized network and support vector network. Therefore, the hidden layer feature mapping $\mathbf{h}(\mathbf{x}_j^i)$ for training sample $\mathbf{x}_j^i$ has

large variety than any other machine learning technique.

5.3.1 Random linear node

Similar to other SLFN based algorithms, the hidden layer feature mapping has the form of a (row) vector $\mathbf{h}(\mathbf{x}_j^i) = [h_1(\mathbf{x}_j^i), \dots, h_L(\mathbf{x}_j^i)]$, where training data is mapped from original input space into the hidden layer space with dimensionality L - number of hidden nodes.

The hidden layer function $h_k(\mathbf{x}_j^i) = G(\mathbf{a}_k, b_k, \mathbf{x}_j^i)$, $k = 1, \dots, L$ is termed ELM random node in theory of ELM. The randomness lies in that parameters $(\mathbf{a}_k, b_k)$ of $(k\text{th})$ hidden node are randomly assigned according to any continuous probability distribution without knowledge of the training data. This unique feature of ELM that differs from other machine learning techniques renders a simple and fast algorithm, which has attracted tremendous attention recently.

With randomly generated parameters, the random node function $G(\mathbf{a}_k, b_k, \mathbf{x}_j^i)$ has very flexible form. The universal approximation capability is proved [51, 55] with such random nodes using almost any type of piecewise continuous functions. Two popular nonlinear functions are listed

- Sigmoid function

$$G(\mathbf{a}, b, \mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b))} \quad (5.5)$$

- Gaussian function

$$G(\mathbf{a}, b, \mathbf{x}) = \exp(-b\|\mathbf{x} - \mathbf{a}\|^2) \quad (5.6)$$

From heuristic point of view, nonlinear hidden node function is generally very helpful for linearly non-separable data or data with small scale. When it comes to ranking prob-

lems, where the number of training data usually is up to tens of thousands, nonlinear hidden node function is computationally costly, yet may not guarantee better performance than linear function. Therefore, linear random node is proposed in the paper, which is new in ELM theory.

- Linear function

$$G(\mathbf{a}, b, \mathbf{x}) = \mathbf{a} \cdot \mathbf{x} + b \quad (5.7)$$

where $\mathbf{a}$ and b are randomly generated as in the traditional ELM random nodes.

Remark 1 As proved in [51, 55], ELM with random nodes is actually universal approximator provided the activation function is any type of piecewise continuous function, such as Sigmoid function, RBF function or piecewise function. Obviously the proposed linear random node complies with such function constraint to render a universal approximator. In other words, ELM with linear random nodes is capable of approximating any continuous functions.

5.3.2 Linear kernel

On the other hand, the feature mapping $\mathbf{h}(\mathbf{x}_j^i)$ may not be explicitly given. And as in support vector network, solution of ELM is kernerlized in (5.9) with kernel defined as

$$\Omega_{ELM} = \mathbf{H}\mathbf{H}^T : \Omega_{ELM i,j} = h(\mathbf{x}_i) \cdot h(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j). \quad (5.8)$$

$$\begin{aligned}
\mathbf{f}(\mathbf{x}) &= \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \\
&= \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{ELM} \right)^{-1} \mathbf{T}.
\end{aligned} \tag{5.9}$$

A widely used kernel is Gaussian kernel $K(\mathbf{u}, \mathbf{v}) = \exp(-\gamma\|\mathbf{u} - \mathbf{v}\|^2)$, where γ is the kernel parameter. However, as far as the computational cost as well as performance efficiency are concerned, the linear kernel $K(\mathbf{u}, \mathbf{v}) = \mathbf{u} \cdot \mathbf{v}$ is adopted in this work.

5.4 Pairwise ELM

In pointwise approaches, it is the relationship between the query-document pair $\mathbf{x}_j^i$ (i th query and j th document) and its corresponding relevance degree y_j^i that is approximated by the hypothesis $f(\mathbf{x}_j^i)$. While the principle in pairwise approaches is that, if any document d_u is more relevant to query q_i comparing with document d_v , then relevance degree of y_u^i should be larger than y_v^i . In other words, with respect to each query, it is the preference of one document over another in terms of relevance degree that is of our interest. Afterwards, a regular binary classifier is used to approximate such preference.

5.4.1 Proposed pairwise ELM

The aim of pairwise ELM is obviously to search for a hypothesis $f()$ such that $f(\mathbf{x}_u^i) > f(\mathbf{x}_v^i)$ if $y_u^i > y_v^i$ (Figure 5.1). Within the framework and traditional least square loss

Figure 5.1: Illustration of input space and output space for pointwise learning and pairwise learning.

function provided by ELM, the problem can be mathematically written as

$$\text{Minimize: } \frac{1}{2} \|\beta\|^2 + C \frac{1}{2} \sum_{i=1}^n \sum \|\xi_{u,v}^i\|^2, \quad (5.10)$$

$$\text{Subject to: } (\mathbf{h}(\mathbf{x}_u^i) - \mathbf{h}(\mathbf{x}_v^i))\beta = (y_u^i - y_v^i) - \xi_{u,v}^i, \quad i = 1, \dots, n.$$

As observed from (5.10), pairwise comparison can only be conducted among documents associated with the same query. With training data given as $(\mathbf{x}_j^i, y_j^i)$, a prior step is required to partition the original training data depending on the query. Inspired by the work of [72], graph theory helps to store such information in a $N \times N$ symmetric matrix $\mathbf{W}$, where $N = \sum_{i=1}^n m(i)$ is the overall number of documents. $\mathbf{W}$ is one type of adjacency graph, where $\mathbf{W}_{i,j} = 1$ if document d_i and d_j are connected to the same query and $\mathbf{W}_{i,j} = 0$ otherwise. All the diagonal entries are set to 1 in $\mathbf{W}$. The corresponding Laplacian matrix $\mathbf{L}$ is then defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D} = \text{diag}\{\sum_{j=1}^N \mathbf{W}_{1,j}, \dots, \sum_{j=1}^N \mathbf{W}_{N,j}\}$. A simple example is demonstrated as follows.

Suppose we are given four samples $\{d_1, d_2, d_3, d_4\}$ and d_1 and d_2 are connected with query 1 while d_3 and d_4 are connected with query 2. We have Suppose we are given four samples $\{d_1, d_2, d_3, d_4\}$ and d_1 and d_2 are connected with query 1 while d_3 and d_4 are connected with query 2. We have

$$\mathbf{W} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}; \mathbf{L} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}. \quad (5.11)$$

By comparing $\mathbf{L} \cdot [d_1 \ d_2 \ d_3 \ d_4]^T$ and $\mathbf{L} \cdot [y_1 \ y_2 \ y_3 \ y_4]^T$, an equation array (5.12) is obtained with the first two reflecting query 1 and the last two reflecting query 2. It is not too difficult to imagine how it works for pairwise comparison when there are more than two queries and 4 documents

$$\begin{cases} d_1 - d_2 \rightarrow y_1 - y_2 \\ -d_1 + d_2 \rightarrow -y_1 + y_2 \\ d_3 - d_4 \rightarrow y_3 - y_4 \\ -d_3 + d_4 \rightarrow -y_3 + y_4 \end{cases}. \quad (5.12)$$

Therefore, with the incorporation of graph $\mathbf{W}$, (5.10) can be rewritten in a way that is more similar to traditional regression/classification form

$$\text{Min: } \frac{1}{2} \|\beta\|^2 + C \frac{1}{2} \mathbf{W} \sum_{i,j=1}^N \|(y_i - y_j) - (f(\mathbf{x}_i) - f(\mathbf{x}_j))\|^2. \quad (5.13)$$

According to Theorem 1 in [72], to minimize $\mathbf{W} \sum_{i,j=1}^N \|(y_i - y_j) - (f(\mathbf{x}_i) - f(\mathbf{x}_j))\|^2$ is equivalent to minimize $(y_i - f(\mathbf{x}_i))^T \mathbf{L} (y_j - f(\mathbf{x}_j))$. The unconstrained optimization

problem becomes

$$\text{Min: } \frac{1}{2} \beta^T \beta + C \frac{1}{2} (\mathbf{T} - \mathbf{H}(\mathbf{x})\beta)^T \mathbf{L} (\mathbf{T} - \mathbf{H}(\mathbf{x})\beta). \quad (5.14)$$

The solution of (5.14) can be obtained by setting the derivative to zero

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{L} \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{L} \mathbf{T}. \quad (5.15)$$

Similar to [55], the kernel version for ranking problems is not difficult to be derived as

$$f(\mathbf{x})_{\text{kernel}} = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \mathbf{L} \Omega_{ELM} \right)^{-1} \mathbf{L} \mathbf{T}. \quad (5.16)$$

By combining with graph theory, pairwise ranking is reduced to a regular regression/classification problem. Moreover, as consistent with ELM theory, both random node version (5.15) and kernel version (5.16) are available. As mentioned earlier, only linear random node and linear kernel are considered in the experiments.

Algorithm 2 Pairwise Ranking ELM Algorithm

Given: Training sample pairs $(\mathbf{x}_j^i, y_j^i)$, number of hidden nodes L , regularization term C , and testing sample $\mathbf{x}$,

Linear Random Node:

- 1: Generate L hidden nodes with the parameters $(\mathbf{a}, b)$ of each node randomly assigned;
- 2: Calculate the hidden layer output matrix $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1^1), \dots]^T$ as shown in (5.2);
- 3: Construct matrix $\mathbf{W}$ to store the information of query-document relationship and compute the corresponding Laplacian matrix $\mathbf{L}$;
- 4: Calculate the output weight β as shown in (5.15) and the output function of testing sample as $f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \cdot \beta$.

Linear Kernel:

- 5: Construct matrix $\mathbf{W}$ to store the information of query-document relationship and compute the corresponding Laplacian matrix $\mathbf{L}$;
 - 6: Calculate the output function of testing sample as shown in (5.8) and (5.16).
-

5.4.2 Discussions with other pairwise approaches

A similar work on pairwise ranking can be found in [72], where a regularization framework is built based on the least square loss function together with the graph theory, resulting in similar solution (5.15,5.16) as in pairwise ELM. However, regularized least square actually can be viewed as one special implementation in the ELM framework. The feature mapping function $G(\mathbf{a}, b, \mathbf{x})$ can be known to users, termed as ELM random node, which is the unique feature of traditional ELM. The feature mapping function may not be known, which results in a kernel version otherwise. In fact, solution provided in [72] can be considered as a special case of pairwise RankELM (RELM). When the parameters $(\mathbf{a}, b)$ in ELM linear random node $G(\mathbf{a}, b, \mathbf{x}) = \mathbf{a}\mathbf{x} + b$ are set as $\mathbf{a} = \mathbf{1}$ and $b = 0$ instead of being randomly generated, the resultant solution is the same as in [72].

A competitive regularized network based pointwise approach, namely ridge regression, which in essence is a regularized regressor with least square loss function. Hence the model solution is quite similar to ELM with non-kernel node (5.4) but $\mathbf{h}()$ in ELM can be a wide type of ELM random nodes as well as the kernel. Again, this algorithm can be considered as one special case of the ELM framework. Therefore, the performance of pointwise ELM with both random nodes and kernel are compared with this algorithm in the experiment.

By mainly concerning the operation of matrix multiplication and inversion and treating number of hidden nodes as a constant (since it is fixed as 1000 in the experiment), the complexity of pointwise ELM with linear random node/linear kernel and pairwise ELM with linear random node/linear kernel are respectively $\mathcal{O}(N)$, $\mathcal{O}(N^3)$, $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ where N is the total number of training samples. Note that pointwise ELM with linear random node has shown great advantage over other state-of-art algorithms, say $\mathcal{O}(N \log N)$ for RankSVM given dataset of such large scale in document ranking.

5.5 Performance Evaluation

The proposed pointwise RankELM and pairwise RankELM with both linear random nodes and linear kernel are evaluated on the publicly available dataset collection LETOR [76], for the task of query-document pair ranking. The performances are further compared with state-of-art linear pointwise approaches (linear regression, regularized regression with L2 loss function) as well as linear pairwise approaches (RankSVM with linear kernel), whose results are made public as well based on uniformly pre-processed datasets. The evaluation metrics are selected as mean average precision (MAP) [6] and normalized discounted cumulative gain (NDCG) [60], which are commonly used in fields of information retrieval.

5.5.1 Data specification

Three datasets from LETOR are tested in the experiment, namely OHSUMED, TD2003 and TD2004 dataset. There are 1,6140 query-document pairs from OHSUMED with 106 queries, 4,9058 from TD2003 with 50 queries and 7,4146 from TD2004 with 72 queries. For OHSUMED, the 3 relevance labels are 2 (relevant), 1 (possibly relevant) and 0 (irrelevant). The relevance label for TD2003 and TD2004 is either 1 (relevant) or 0 (irrelevant). The features are normalized version and 5-fold cross validation are used. The data is partitioned into 5 parts, where 3 are used for training, 1 for parameter selection (according to MAP result) and 1 for performance validation. The final result in comparison is averaged over the 5 folders.

5.5.2 Parameter settings

Since the feature mapping in ELM can be known as well as unknown, the linear random node and linear kernel are both validated. Two parameters need to be tuned in linear random node: number of hidden nodes L and regularization term C . According to the litera-

Figure 5.2: Relationship of the MAP with choices of the parameters tested on Folder 1 from OHSUMED.

ture of ELM, the performance is normally insensitive to choice of L . Figure 5.2 displays the result given different choices of parameters for pointwise ELM. It is observable that the performance of pointwise ELM is not sensitive with choice of L , as consistent with the ELM theory. And similar observation can be found for pairwise ELM with linear random node. Therefore, L is fixed as 1,000 in the linear random node through the whole experiment. A wide range of trials are made with $C \in \{2^{-30}, 2^{-28}, \dots, 2^{18}, 2^{20}\}$. In linear kernel implementation, only the regularization term C are required specification, of which a grid search as in linear random node is conducted.

5.5.3 Experiment results

Each folder in a dataset contains the partitions for training, validation and testing. Table 5.1 displays the MAP value of learning models validated on testing partition, folder by folder. And an average MAP value over all the 5 folders is provided as well. It is obvious that learning algorithms with regularization term has significantly improved the performance compared with linear regression without regularization term. On most of the cases, ranking algorithms based on ELM where the feature mapping can be linear random node or linear kernel outperforms regularized linear regression.

In the framework of pairwise learning, RankSVM is selected as a reference to compare

Table 5.1: MAP value obtained from various pointwise learning methods and pairwise learning methods of every folder from dataset OHSUMED, TD2003 and TD2004.

OHSUMED	Pointwise Methods				Pairwise Methods		
	regression	Regularized L2	RELM Node	RELM Kernel	RankSVM	RELM Node	RELM Kernel
F1	0.2979	0.3374	0.3421	0.3459	0.3038	0.3421	0.3450
F2	0.4302	0.4351	0.4510	0.4466	0.4468	0.4483	0.4514
F3	0.4398	0.4703	0.4734	0.4751	0.4648	0.4707	0.4696
F4	0.4978	0.5061	0.5137	0.5127	0.4990	0.5118	0.5115
F5	0.4442	0.4703	0.4574	0.4724	0.4528	0.4639	0.4642
Avg	0.4220	0.4439	0.4475	0.4505	0.4334	0.4474	0.4483
TD2003	regression	Regularized L2	RELM Node	RELM Kernel	RankSVM	RELM Node	RELM Kernel
F1	0.1262	0.1745	0.1677	0.1511	0.1637	0.1555	0.1614
F2	0.3009	0.2031	0.3010	0.2982	0.2576	0.2965	0.3138
F3	0.2665	0.3703	0.2673	0.4101	0.4079	0.3242	0.4093
F4	0.2658	0.2734	0.2466	0.2332	0.2356	0.2707	0.2411
F5	0.2450	0.1955	0.2437	0.2536	0.2490	0.2395	0.2610
Avg	0.2409	0.2434	0.2453	0.2692	0.2628	0.2573	0.2773
TD2004	regression	Regularized L2	RELM Node	RELM Kernel	RankSVM	RELM Node	RELM Kernel
F1	0.1725	0.1908	0.2139	—	0.2107	0.2306	—
F2	0.2117	0.1817	0.2118	—	0.2091	0.2186	—
F3	0.2196	0.2035	0.2146	—	0.2063	0.1948	—
F4	0.1910	0.1886	0.2077	—	0.2185	0.2263	—
F5	0.2442	0.2311	0.2562	—	0.2740	0.2612	—
Avg	0.2078	0.1992	0.2208	—	0.2237	0.2263	—

with. Traditional SVM with linear kernel is adopted as a binary classifier, with computational complexity as a quadratic function of the number of training samples which is normally quite large. It can be found that the performance of pairwise RELM outperforms RSVM on most of the cases. In addition, the option of linear random node actually helps to significantly reduce the computational complexity, which is scaled with the number of hidden nodes L (1000 in the experiment). Due to large dataset size, RELM with only random node version is conducted on dataset TD2004.

Since the metric NDCG is suitable for multiple label problems and precision@position is suitable for binary problems, the NDCG result of OHSUMED and precision@position results of TD2003 and TD2004 are provided in Table 5.2 and Table 5.3.

5.6 Summary

In this work, ELM was extended to the field of relevance ranking for the first time. According to literature of learn-to-rank algorithms, ranking problems can be formulated

Table 5.2: NDCG value obtained from various pointwise learning methods and pairwise learning methods of every folder from dataset OHSUMED.

OHSUMED	Pointwise Methods				Pairwise Methods		
	regression	Regularized L2	RELM Node	RELM Kernel	RankSVM	RELM Node	RELM Kernel
NDCG@1	0.4456	0.5361	0.5550	0.5232	0.4958	0.5211	0.5642
NDCG@2	0.4532	0.4784	0.4932	0.5024	0.4331	0.4977	0.5019
NDCG@3	0.4426	0.4740	0.4874	0.4975	0.4207	0.4895	0.4923
NDCG@4	0.4368	0.4601	0.4739	0.4896	0.4240	0.4742	0.4866
NDCG@5	0.4278	0.4568	0.4696	0.4782	0.4164	0.4691	0.4755
NDCG@6	0.4216	0.4536	0.4623	0.4685	0.4159	0.4589	0.4611
NDCG@7	0.4217	0.4449	0.4556	0.4645	0.4133	0.4557	0.4567
NDCG@8	0.4187	0.4444	0.4565	0.4616	0.4072	0.4496	0.4573
NDCG@9	0.4136	0.4427	0.4529	0.4627	0.4124	0.4466	0.4555
NDCG@10	0.4110	0.4436	0.4501	0.4607	0.4140	0.4422	0.4574

Table 5.3: Precision value obtained from various pointwise learning methods and pairwise learning methods of every folder from dataset TD2003 and TD2004.

TD2003	Pointwise Methods				Pairwise Methods		
	regression	Regularized L2	RELM Node	RELM Kernel	RankSVM	RELM Node	RELM Kernel
P@1	0.3200	0.3400	0.3600	0.3600	0.3200	0.3400	0.3600
P@2	0.3000	0.3400	0.3400	0.3500	0.3100	0.3200	0.3400
P@3	0.2600	0.3000	0.2867	0.2867	0.2933	0.2533	0.2800
P@4	0.2450	0.2550	0.2750	0.2600	0.2850	0.2400	0.2700
P@5	0.2160	0.2360	0.2440	0.2400	0.2760	0.2280	0.2600
P@6	0.2133	0.2067	0.2267	0.2267	0.2500	0.2233	0.2333
P@7	0.2057	0.2086	0.2029	0.2057	0.2229	0.2086	0.2200
P@8	0.1950	0.1975	0.2050	0.1950	0.2075	0.1975	0.2025
P@9	0.1844	0.1822	0.1911	0.1889	0.2000	0.1845	0.1911
P@10	0.1780	0.1760	0.1820	0.1780	0.1880	0.1760	0.1820
TD2004	regression	Regularized L2	RELM Node	RELM Kernel	RankSVM	RELM Node	RELM Kernel
P@1	0.3600	0.2933	0.4000	—	0.4133	0.4533	—
P@2	0.3400	0.3200	0.3800	—	0.3467	0.4000	—
P@3	0.3333	0.2978	0.3511	—	0.3467	0.3600	—
P@4	0.3200	0.2833	0.3333	—	0.3333	0.3433	—
P@5	0.3120	0.2720	0.3200	—	0.3013	0.3253	—
P@6	0.2867	0.2644	0.2933	—	0.2867	0.2933	—
P@7	0.2705	0.2629	0.2781	—	0.2800	0.2857	—
P@8	0.2683	0.2500	0.2733	—	0.2667	0.2700	—
P@9	0.2593	0.2415	0.2607	—	0.2548	0.2578	—
P@10	0.2493	0.2347	0.2507	—	0.2520	0.2480	—

in pointwise or pairwise point of view, where in this paper pointwise RankELM and pairwise RankELM are proposed respectively. Ranking algorithms based on ELM actually has a large variety of choices when it comes to the feature mapping functions, namely any type of piecewise continuous function or in the form of a kernel. In the experiment, as far as computation cost is concerned, the linear random node and linear kernel are used. The experimental results show that ranking algorithms based on ELM outperforms state-of-art linear ranking algorithms on most of cases. To effectively and efficiently extend ELM with nonlinear feature mappings on large scale data remains an interesting topic.

To deal with large scale dataset, there are many techniques in the literature. Online sequential learning is one of them, which is also capable of learning sequentially arriving data. Online sequential ELM (OS-ELM) was proposed in ELM theory [67], which is based on the original ELM. It is obvious that the unified ELM framework are better in generalization performance as well as variety in feature mapping functions. Moreover, OS-ELM focused only on left pseudo-inversion during implementation. Therefore, in the following work, an online sequential mode is studied based on the latest ELM framework, exploring both left and right pseudo-inversion in implementation.

Another characteristic of document ranking data is that number of irrelevant documents usually exceeds number of relevant documents, which is a typical imbalanced data distribution. Any normal machine learning technique, with assumption of balanced data distribution, would naturally result in a bias towards the majority class. Therefore, an improvement on the ELM framework is proposed in the work after next one to carried out imbalanced data learning where the bias is aimed to be eliminated.

Chapter 6

Online Sequential Learning Based on Enhanced Extreme Learning Machine Using Left or Right Pseudo-Inverse

6.1 Introduction

Training algorithms for feedforward networks, including least-square based extreme learning machine (ELM) [57, 51, 47, 48] for single-hidden layer feedforward networks (SLFNs) and backpropagation (BP) based on gradient descent [80] for multi-layer feedforward neural networks, have attracted the attention of many researchers for the past years. Main focus is given to the batch learning mode of the aforementioned algorithms. However, in real world applications, the training data may not come at once or the size of training data may be too large. In such circumstances, online sequential learning instead of batch learning is preferred.

Sequential learning algorithms based on BP for SLFNs with additive nodes have been proposed in literature [71, 5]. Resource allocation network (RAN), one of the training algorithms for feedforward networks with RBF nodes, has been extended to sequen-

tial learning mode as well [53, 54]. These sequential learning algorithms may not be efficient enough due to the disadvantages in convergence rate, training speed and parameter tuning complexity. Moreover, the data can be learned only on a one by one basis. Online sequential extreme learning machine (OS-ELM) was proposed by Liang, et al. [67] where the training data can be learned not only on a one-by-one basis but also on a chunk-by-chunk basis. OS-ELM is based on the original ELM where the SLFN can be viewed as a linear system with the solution being the left pseudo-inverse of the hidden layer output matrix $\mathbf{H}$ in the following form: $\mathbf{H}^\dagger = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$. Inheriting the advantage of simplicity from original ELM, which randomly generates the hidden layer nodes, OS-ELM outperforms the state-of-art sequential learning algorithms both in generalization capability and in computational efficiency. Therefore, the performance comparison with the state-of-art method in this paper is conducted over OS-ELM.

As mentioned in ridge regression theory [41], the solution tends to be more stable and better generalization performance can be achieved by adding a positive value to the diagonal of $\mathbf{H}^T \mathbf{H}$ or $\mathbf{H} \mathbf{H}^T$. The resultant enhanced ELM with *sigmoid* additive nodes has been studied in the work of Toh [86] and Deng, et al. [25]. In the recent work of Huang, et al. [55], the idea of a unified solution based on ELM (referred to as *eELM*) for SLFNs and other regularization networks with wide type of feature mappings or kernels was first proposed and fulfilled.

OS-ELM is derived on the basis of batch mode original ELM using left pseudo-inverse. However, latest development in ELM [55] has shown much better advantage in generalization capability than the original ELM. On the other hand, to the best of our knowledge, very few work has been done about the sequential learning using right pseudo-inverse. In this paper, we propose an online sequential learning algorithm based on eELM using the right pseudo-inverse (referred to as *OS-eELM-right*). Moreover, the online sequential learning for SLFNs and other regularization networks is built regarding both left pseudo-inverse (referred to as *OS-eELM-left*) and right pseudo-inverse (*OS-eELM-right*). The performance of the proposed framework is compared with original OS-ELM through six benchmarking data sets for regression and multi-class classifica-

tion applications.

6.2 Brief Review of eELM

According to the ridge regression theory [41], one can add a positive value to the diagonal of $\mathbf{H}\mathbf{H}^T$ or $\mathbf{H}^T\mathbf{H}$, the resultant solution is more stable and tends to have better generalization performance. In [55] a unified solution framework for SLFNs, SVM [89] and other regularization networks were proposed where solution based on right pseudo-inverse for small scale of dataset and solution based on left pseudo-inverse for large scale dataset are given by

$$\begin{aligned} \text{Right: } \beta &= \mathbf{H}^\dagger \mathbf{T} = \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T} \\ \text{Left: } \beta &= \mathbf{H}^\dagger \mathbf{T} = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}. \end{aligned} \tag{6.1}$$

where C is determined by users. Observed from simulations on wide types of datasets, the hidden node number L is normally set as a large value to obtain good generalization performance, while the regularization term C in (6.1) is the only parameter user needs to specify according to various datasets.

6.3 The Proposed Online Sequential Learning Algorithm

In this section, the online sequential learning algorithm based on eELM is proposed, comprising of OS-eELM-left which calculates the left pseudo-inverse when small scale of initial training data is observed, and OS-eELM-right which calculates the right pseudo-inverse when large scale of initial training data is presented.

6.3.1 OS-eELM-left

The only difference between OS-eELM-left and OS-ELM is that OS-eELM is derived on the basis of the latest development of ELM [55], which we call eELM in this paper. Hence, it is not difficult to find out that only a slight change need to be made to extend OS-ELM to OS-eELM-left. The solution is to obtain the initial output weight $\beta^{(0)}$ as $\beta^{(0)} = (\frac{\mathbf{I}}{C} + \mathbf{H}_0^T \mathbf{H}_0)^{-1} \mathbf{H}_0^T \mathbf{T}_0 = \mathbf{P}_0 \mathbf{H}_0^T \mathbf{T}_0$. The rest of the equations in the training procedure remains exactly the same as OS-ELM. Therefore, more focus is given to the derivation of OS-eELM-right which makes a great difference to OS-ELM by using right pseudo-inverse.

6.3.2 The proposed OS-eELM-right

Given the initial training set $\mathfrak{X}_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$, the minimum norm least square solution to minimize $\|\mathbf{H}_0 \beta - \mathbf{T}_0\|$ is given by $\beta^{(0)} = \hat{\mathbf{H}}_0^\dagger \mathbf{T}_0 = \mathbf{H}_0^T (\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T)^{-1} \mathbf{T}_0$. Given another chunk of data $\mathfrak{X}_1 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=N_0+1}^{N_0+N_1}$ of size N_1 , from the batch learning point of view, the problem is to minimize

$$\begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \beta^{(1)} - \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}. \quad (6.2)$$

According to (6.1), the output weight β becomes

$$\beta^{(1)} = \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T \right)^{-1} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}. \quad (6.3)$$

Let $\mathbf{A} = \frac{\mathbf{I}}{C} + \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^T$, we have

$$\mathbf{A}^{-1} = \begin{bmatrix} \frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T & \mathbf{H}_0 \mathbf{H}_1^T \\ \mathbf{H}_1 \mathbf{H}_0^T & \frac{\mathbf{I}}{C} + \mathbf{H}_1 \mathbf{H}_1^T \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}.$$

This 2×2 block matrix is invertible if and only if $(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T)$ and its schur complement are invertible [13]. From the initial step, it is easy to show that $(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T)$ is invertible. Similar to [28], it can be proved that schur complement of $(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T)$, denoted by $\mathbf{S}$, is invertible with probability one. And according to [13], we have

$$\begin{aligned} \mathbf{A}_{11} &= \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T \right)^{-1} + \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T \right)^{-1} \\ &\quad (\mathbf{H}_0 \mathbf{H}_1^T) \mathbf{S}^{-1} (\mathbf{H}_1 \mathbf{H}_0^T) \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T \right)^{-1} \\ \mathbf{A}_{12} &= - \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T \right)^{-1} (\mathbf{H}_0 \mathbf{H}_1^T) \mathbf{S}^{-1} \\ \mathbf{A}_{21} &= - \mathbf{S}^{-1} (\mathbf{H}_1 \mathbf{H}_0^T) \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T \right)^{-1} \\ \mathbf{A}_{22} &= \mathbf{S}^{-1} \end{aligned} \tag{6.4}$$

where

$$\begin{aligned} \mathbf{S} &= \left(\frac{\mathbf{I}}{C} + \mathbf{H}_1 \mathbf{H}_1^T \right) - (\mathbf{H}_1 \mathbf{H}_0^T) \left(\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T \right)^{-1} (\mathbf{H}_0 \mathbf{H}_1^T) \\ &= \frac{\mathbf{I}}{C} + \mathbf{H}_1 (\mathbf{I} - \mathbf{H}_0^\dagger \mathbf{H}_0) \mathbf{H}_1^T. \end{aligned} \tag{6.5}$$

Denote $\mathbf{P}_0 = \mathbf{I} - \mathbf{H}_0^\dagger \mathbf{H}_0$, then $\mathbf{S}_1$, which represents the state of $\mathbf{S}$ after the first chunk data arrives, can be expressed as: $\mathbf{S}_1 = \frac{\mathbf{I}}{C} + \mathbf{H}_1 (\mathbf{I} - \mathbf{H}_0^\dagger \mathbf{H}_0) \mathbf{H}_1^T = \frac{\mathbf{I}}{C} + \mathbf{H}_1 \mathbf{P}_0 \mathbf{H}_1^T$. Thus we

have

$$\begin{aligned}
\beta^{(1)} &= \begin{bmatrix} \mathbf{H}_0^T & \mathbf{H}_1^T \end{bmatrix} \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix} \\
&= \mathbf{H}_0^T \mathbf{A}_{11} \mathbf{T}_0 + \mathbf{H}_1^T \mathbf{A}_{21} \mathbf{T}_0 + \mathbf{H}_0^T \mathbf{A}_{12} \mathbf{T}_1 + \mathbf{H}_1^T \mathbf{A}_{22} \mathbf{T}_1 \\
&= \beta^{(0)} + \mathbf{H}_0^\dagger \mathbf{H}_0 \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 \beta^{(0)} - \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 \beta^{(0)} \\
&\quad - \mathbf{H}_0^\dagger \mathbf{H}_0 \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{T}_1 + \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{T}_1 \\
&= \beta^{(0)} - (\mathbf{I} - \mathbf{H}_0^\dagger \mathbf{H}_0) \mathbf{H}_1^T \mathbf{S}_1^{-1} (\mathbf{H}_1 \beta^{(0)} - \mathbf{T}_1) \\
&= \beta^{(0)} - \mathbf{P}_0 \mathbf{H}_1^T \mathbf{S}_1^{-1} (\mathbf{H}_1 \beta^{(0)} - \mathbf{T}_1).
\end{aligned} \tag{6.6}$$

In completion of the learning of the first chunk of data, $\mathbf{P}$ is updated as follows

$$\begin{aligned}
\mathbf{P}_1 &= \mathbf{I} - \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{H}_0 \\ \mathbf{H}_1 \end{bmatrix} \\
&= \mathbf{I} - \mathbf{H}_0^\dagger \mathbf{H}_0 - \mathbf{H}_0^\dagger \mathbf{H}_0 \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 \mathbf{H}_0^\dagger \mathbf{H}_0 + \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 \\
&\quad \mathbf{H}_0^\dagger \mathbf{H}_0 + \mathbf{H}_0^\dagger \mathbf{H}_0 \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 - \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 \\
&= (\mathbf{I} - \mathbf{H}_0^\dagger \mathbf{H}_0) + (\mathbf{I} - \mathbf{H}_0^\dagger \mathbf{H}_0) \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 \mathbf{H}_0^\dagger \mathbf{H}_0 \\
&\quad - (\mathbf{I} - \mathbf{H}_0^\dagger \mathbf{H}_0) \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 \\
&= \mathbf{P}_0 - \mathbf{P}_0 \mathbf{H}_1^T \mathbf{S}_1^{-1} \mathbf{H}_1 \mathbf{P}_0.
\end{aligned} \tag{6.7}$$

As seen from equation (6.6), the output weight $\beta^{(1)}$ is expressed as a function of $\beta^{(0)}$, $\mathbf{S}_1$, $\mathbf{P}_0$ and newly arriving chunk of data $\mathbf{H}_1$; $\mathbf{S}_1$ can actually be expressed as the function of $\mathbf{P}_0$ and newly arriving chunk $\mathbf{H}_1$; $\mathbf{P}_0$ is updated using itself and newly arriving data only in each iteration. Therefore, whenever new data arrives, a recursive algorithm for updating the output weight β can be derived. Given $(k+1)th$ chunk of data, the hidden

layer output matrix of which is $\mathbf{H}_{k+1}$, the recursive algorithm works as follows

$$\begin{aligned}\mathbf{S}_{k+1} &= \frac{\mathbf{I}}{C} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T \\ \beta^{(k+1)} &= \beta^{(k)} - \mathbf{P}_k \mathbf{H}_{k+1}^T \mathbf{S}_{k+1}^{-1} \left(\mathbf{H}_{k+1} \beta^{(k)} - \mathbf{T}_{k+1} \right)\end{aligned}\quad (6.8)$$

Finally, the proposed OS-eELM-right can be summarized as follows.

The proposed algorithm can learn data chunk by chunk or one by one, thus is able to handle the situation where data is arriving sequentially. In general, there are two stages during training, namely the initialization stage and incremental learning stage.

- (1) **Initialization:** Given the chunk of initial training data $\mathfrak{X}_0 = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N_0}$ and $\mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m$,

- a) Randomly generate L hidden nodes (additive hidden nodes or RBF hidden nodes). Good performance can be achieved normally when L is assigned a large value.
- b) Calculate the initial hidden layer output matrix $\mathbf{H}_0$.
- c) Compute the initial output weight $\beta^{(0)} = \mathbf{H}_0^T (\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T)^{-1} \mathbf{T}_0$.
- d) Calculate $\mathbf{P}_0 = \mathbf{I} - \hat{\mathbf{H}}_0^\dagger \mathbf{H}_0 = \mathbf{I} - \mathbf{H}_0^T (\frac{\mathbf{I}}{C} + \mathbf{H}_0 \mathbf{H}_0^T)^{-1} \mathbf{H}_0$.
- e) Set $k = 0$.

- (2) **Incremental Learning:** Given $(k+1)th$ chunk of data

$\mathfrak{X}_{k+1} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=(\sum_{j=0}^k N_j)+1}^{\sum_{j=0}^{k+1} N_j}$, where N_{k+1} presents the number of training samples in the $(k+1)th$ chunk,

- a) Calculate the hidden layer output matrix $\mathbf{H}_{k+1}$ for the $(k+1)th$ chunk of data.
- b) Update the output weight:

$$\begin{aligned}\mathbf{S}_{k+1} &= \frac{\mathbf{I}}{C} + \mathbf{H}_{k+1} \mathbf{P}_k \mathbf{H}_{k+1}^T \\ \beta^{(k+1)} &= \beta^{(k)} - \mathbf{P}_k \mathbf{H}_{k+1}^T \mathbf{S}_{k+1}^{-1} \left(\mathbf{H}_{k+1} \beta^{(k)} - \mathbf{T}_{k+1} \right)\end{aligned}\quad (6.9)$$

c) Update $\mathbf{P}_{k+1}$:

$$\mathbf{P}_{k+1} = \mathbf{P}_k - \mathbf{P}_k \mathbf{H}_{k+1}^T \mathbf{S}_{k+1}^{-1} \mathbf{H}_{k+1} \mathbf{P}_k.$$

d) Set $k = k + 1$. Go to (2).

6.4 Discussions

The choice of OS-eELM-left or OS-eELM-right is dependent on the size of initial training data N_0 . It is wise to choose OS-eELM-right when a small scale of initial training data is provided, otherwise OS-eELM-left is a better choice. Since good performance is achieved normally when the number of hidden nodes is set a large value, which is 1,000 in [55] corresponding to up to 40,000 observations, the threshold for the choice of left or right is set the same as 1,000. OS-eELM-left is recommended when more than 1,000 observations are given during the initial training phase; otherwise OS-eELM-right is supposed to save the computation and time.

Both of OS-eELM-left and OS-eELM-right can learn data on a one-by-one basis or chunk-by-chunk basis where the chunk size can be varied. When only one observation $(\mathbf{x}_{k+1}, t_{k+1})$ is provided during the incremental learning phase, the updating equation for $\beta^{(k+1)}$ and $\mathbf{P}_{k+1}$ can be further simplified

$$\begin{aligned} \beta^{(k+1)} &= \beta^{(k)} - \frac{\mathbf{P}_k \mathbf{h}_{k+1}^T (\mathbf{h}_{k+1} \beta^{(k)} - t_{k+1})}{\frac{1}{C} + \mathbf{h}_{k+1} \mathbf{P}_k \mathbf{h}_{k+1}^T} \\ \mathbf{P}_{k+1} &= \mathbf{P}_k - \frac{\mathbf{P}_k \mathbf{h}_{k+1}^T \mathbf{h}_{k+1} \mathbf{P}_k}{\frac{1}{C} + \mathbf{h}_{k+1} \mathbf{P}_k \mathbf{h}_{k+1}^T}. \end{aligned} \quad (6.10)$$

Similar to OS-ELM, batch mode eELM-left can be considered as the special case of OS-eELM-left while batch mode eELM-right is the special case of OS-eELM-right, when $N_0 = N$.

6.5 Performance Evaluation

Since OS-ELM has been verified to have better performance than other well-known sequential learning algorithms [67], in this paper, the proposed OS-eELM-right is compared with OS-ELM and OS-eELM-left. For sequential learnings, the number of initial training samples is set the same as in [67]. All simulations are conducted on a laptop with Pentium Dual CPU 1.86GHz and 2GB memory.

6.5.1 Data specification

As shown in Table 6.1, six benchmarking data sets [12] have been studied in the simulations, including three regression problems (auto-MPG and abalone), and three multi-class classification problems (image segmentation and satellite image). Same as in [67], the input attributes and output attributes of regression problems are normalized into the range of $[0,1]$; the input attributes of classification problems are normalized into the range of $[-1,1]$.

Table 6.1: Specification of benchmarking datasets.

Dataset	#Attributes	#Classes	#Train Data	#Test 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

6.5.2 Parameter settings

Sigmoidal additive hidden nodes are selected for all the algorithms. Other hidden nodes, such as RBF nodes, can be studied in the future work. Similar to [55, 52], good performance is achieved usually when the number of hidden nodes L is large. The per-

formances of our proposed algorithm and OS-eELM-left are insensitive to L as well (Figure 6.1). Therefore, it is convenient that L is fixed as 1000 for both algorithms. Another parameter C which determines the positive value added to the diagonal needs to be specified by users. A wide range of $C \{2^{-24}, 2^{-23}, \dots, 2^{24}, 2^{25}\}$ is validated for the optimal testing rate.

Figure 6.1: Testing RMSE of abalone with respect to C and L

6.5.3 Comparison of average testing accuracy

The average testing RMSE for regression problems and average testing rate for classification problems are obtained over 50 trials.

It can be observed from Table 6.2 and Table 6.3 that for online sequential learning, the accuracy on testing data is improved when a positive value is added to the diagonal of $\mathbf{H}^T \mathbf{H}$ for left pseudo-inverse or $\mathbf{H} \mathbf{H}^T$ for right pseudo-inverse. It is also shown that the sequential algorithms using right or left pseudo-inverse obtain similar testing accuracy.

6.5.4 Comparison of training time

Different from the original ELM, the number of hidden nodes is fixed as a large value (1000) in the enhanced ELM. That is the reason why enhanced ELM tends to be slower

Table 6.2: Performance comparison on regression problems

Regression		Auto-MPG					Abalone					California Housing				
	ELM-Based	Time (s)	RMSE		Parameter		Time (s)	RMSE		Parameter		Time (s)	RMSE		Parameter	
	Algorithms		Training	Testing	C	L		Training	Testing	C	L		Training	Testing	C	L
Sequential 20-by-20	OS-ELM	0.0103	0.0686	0.0759	-	25	0.0437	0.0753	0.0779	-	25	0.3204	0.1304	0.1331	-	50
	OS-eELM-left	3.9827	0.0607	0.0724	2^8	1000	23.9346	0.0719	0.0774	2^{12}	1000	75.1894	0.1246	0.1282	2^{10}	1000
	OS-eELM-right	2.1338	0.0610	0.0709	2^8	1000	2.1338	0.0610	0.0709	2^8	1000	78.7415	0.1247	0.1282	2^{10}	1000
Sequential [10,30]	OS-ELM	0.0115	0.0680	0.0781	-	25	0.0711	0.0752	0.0783	-	25	0.4842	0.1303	0.1322	-	50
	OS-eELM-left	3.8794	0.0612	0.0706	2^8	1000	26.3018	0.0724	0.0771	2^{10}	1000	78.7824	0.1200	0.1294	2^{14}	1000
	OS-eELM-right	2.6289	0.0578	0.0701	2^{10}	1000	15.8310	0.0752	0.0769	2^4	1000	71.4949	0.1203	0.1300	2^{14}	1000
Sequential 1-by-1	OS-ELM	0.0365	0.0690	0.0739	-	25	0.2315	0.0755	0.0780	-	25	3.4161	0.1300	0.1338	-	50
	OS-eELM-left	17.3220	0.0643	0.0714	2^6	1000	131.1391	0.0736	0.0766	2^8	1000	537.7711	0.1185	0.1285	2^{16}	1000
	OS-eELM-right	14.1596	0.0644	0.0708	2^6	1000	106.4070	0.0724	0.0769	2^{10}	1000	497.1973	0.1203	0.1300	2^{14}	1000

Table 6.3: Performance comparison on multiclass classification problems

Classification		Image Segment					Satellite Image					DNA				
	ELM-Based	Time (s)	Rates(%)		Parameter		Time (s)	Rates(%)		Parameter		Time (s)	Rates(%)		Parameter	
	Algorithms		Training	Testing	C	L		Training	Testing	C	L		Training	Testing	C	L
Sequential 20-by-20	OS-ELM	0.7806	97.02	94.88	-	180	19.9366	91.96	88.95	-	400	1.9425	92.67	87.94	-	200
	OS-eELM-left	17.7426	98.19	95.91	2^8	1000	55.5763	94.62	89.95	2^6	1000	20.5104	97.26	93.75	2^{-8}	1000
	OS-eELM-right	11.9593	98.11	95.99	2^8	1000	52.0285	93.92	90.04	2^4	1000	20.1678	97.25	93.73	2^{-8}	1000
Sequential [10,30]	OS-ELM	1.1691	97.05	94.89	-	180	28.7451	91.97	88.98	-	400	1.8754	92.87	88.06	-	200
	OS-eELM-left	15.6828	98.54	95.94	2^{10}	1000	55.2733	94.64	89.91	2^6	1000	20.9615	97.30	93.66	2^{-8}	1000
	OS-eELM-right	9.4103	98.56	96.00	2^{10}	1000	51.5571	95.03	89.67	2^8	1000	21.3806	97.32	93.77	2^{-8}	1000
Sequential 1-by-1	OS-ELM	10.3722	97.05	94.75	-	180	377.1901	92.00	89.00	-	400	24.2925	92.57	87.91	-	200
	OS-eELM-left	93.6714	97.58	95.88	2^6	1000	284.5552	93.93	89.99	2^4	1000	120.4346	97.28	93.74	2^{-8}	1000
	OS-eELM-right	88.2663	98.17	95.92	2^8	1000	280.7665	93.94	89.98	2^4	1000	132.8770	97.24	93.65	2^{-8}	1000

than the original ELM in both batch and sequential learning mode.

It is not difficult to find out from (6.9) that the computational complexity of the incremental learning phases for both OS-eELM-left and OS-eELM-right are similar. Therefore, Main focus on analysis of computational complexity is given to the initial phases for both formulas. For right pseudo-inverse of the form $\mathbf{H}^\dagger = \mathbf{H}^T (\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)^{-1}$, matrix inversion is carried out on a $N \times N$ matrix. While it is $L \times L$ in the case of left pseudo-inverse. Therefore, in terms of the training speed, right pseudo-inverse is preferred when the size of training data is small; otherwise left pseudo-inverse is more appropriate. Hence, it can be observed from Table 6.2 and Table 6.3 that OS-eELM-right runs relatively faster than OS-eELM-left since the size of initial training data is chosen much smaller than the number of hidden nodes.

6.6 Summary

In this work, an online sequential learning algorithm for SLFNS and other regularization networks based on the enhanced ELM is proposed, which is capable of learning data on a one-by-one basis or chunk-by-chunk basis. Simulations on six benchmarking datasets have shown that, by adding a positive value to the diagonal of $\mathbf{H}\mathbf{H}^T$ and $\mathbf{H}^T\mathbf{H}$, the generalization performance of our proposed methods outperform the original OS-ELM. In addition, in the simulations, OS-eELM-right is more suitable for sequential learning than OS-eELM-left concerning the issue of training speed since there are less than 1,000 observations during the initial training phase.

In addition to large scale or sequentially arriving, the real world data is often of imbalanced distribution, where some classes significantly outnumber other classes. This characteristic of a data often exists in applications such as document ranking (studied in Chapter 5), fraud detection and network intrusion and so on. In the next work, the unified ELM framework is modified to adapt to imbalanced data learning.

Chapter 7

Weighted Extreme Learning Machine for Imbalance Learning

7.1 Introduction

Raw data with imbalanced class distribution can be found almost everywhere, from biomedical application to fraud detection or network intrusion, etc. When classifying data with complex class distribution, the regular learning algorithm have a natural tendency to favor the majority class by assuming balanced class distribution or equal misclassification cost. Typically, to deal with imbalanced data one can attempt to establish balanced data distribution through various sampling methods or algorithmic approaches [36, 1]. To re-balance the data distribution, there are mainly two sampling techniques: undersampling approach which removes a fraction of the majority samples and the oversampling approach which duplicates the minority samples. The challenge the oversampling method is facing is to distinguish the informative samples and the redundant samples. And the assumption that the neighborhood of a minority sample share the same label is not always satisfied with different types of data. As the other type of re-balance technique, the algorithmic approach is considered consistent with the sampling

approach and is of particular interest in this paper.

A popular algorithmic approach is to assign a different misclassification cost for each particular example [36]. But cases when the misclassification cost matrix is difficult or unnecessary to generate may arise in the real world applications. The most straightforward solution is to automatically generate the misclassification cost matrix in accordance with the class distribution, which usually is in the form of a weight scheme inversely proportional to the number of samples in the class.

Weighted regularized ELM was proposed in the work of Toh [86] and Deng, et al [25]. However, neither of them was targeting the imbalanced data problem. Further, “generalized” feature mapping and kernel node was not considered as in the latest work of Huang, et al [55].

In this paper, a unified solution of weighted ELM for “generalized” SLFNs is proposed, to tackle binary/multiclass classification tasks, which is robust to both balanced and imbalanced data distribution. The proposed method shares some important features in common with unweighted ELM. They are simple in theory and fast in implementation; the hidden nodes can be wide type of feature mapping or in the form of the kernel; and the classification capability to classify any disjoint region is proved to guarantee the generalization performance. In an attempt to alleviate the bias in performance caused by imbalanced class distribution, an extra weight is assigned to each sample to strengthen the impact of minority class while weaken the relative impact of majority class. In the rest of the paper, we refer to the majority class with negative label and the minority class with positive label.

7.2 Evaluation Metrics

Usually overall accuracy is used to measure the effectiveness of a classifier. Unfortunately, in presence of imbalanced data, this metric may fail to provide adequate informa-

tion about the performance of the classifier. Furthermore, the method is very sensitive to the class distribution and might be misleading in some way [36]. For instance, given a binary classification problem consisting of 1 percent positive class and 99 percent negative class. Any dumb classifier would easily achieve 99 percent accuracy by classifying all the samples as negative. Although 99 percentage of overall accuracy seems quite impressive, the fact that the accuracy for the minority class is actually 0 should not be ignored.

To give more insight into the accuracy obtained within each class in lieu of the accuracy of all the samples, an evaluation metric named G-mean is adopted in this paper. After computing the accuracy within each class, the final result to measure the functionality of a classifier is the geometric mean of those accuracies. Take the binary classifier above for example, the G-mean value would be as low as 0 since the accuracy for minority class is 0. In particular, for binary classification problem, G-mean is the square root of (positive class accuracy $\times$ negative class accuracy), where TP, TN, FP, FN stand for true positive, true negative, false positive and false negative, respectively (7.1)

$$\text{G-mean} = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}}. \quad (7.1)$$

Another interesting tool, receiver operating characteristics (ROC) graph [27], provides a visual illustration of the performance of classifiers on binary datasets, where a classifier corresponds to a point. X-coordinate of the point represents false positives rate (FP_rate) and y-coordinate represents true positives rate (TP_rate), so that classification results for both positive class and negative class are perceivable with a single point. That is to say, the performance exhibited by ROC graph is independent of the class distribution and cost information.

$$\begin{aligned} TP_rate &= \frac{TP}{\#P} \\ FP_rate &= \frac{FP}{\#N} = 1 - \frac{TN}{\#N} = 1 - TN_rate, \end{aligned} \quad (7.2)$$

where #P is the number of samples in positive class; #N for negative class. For instance,

Figure 7.1: An example of ROC graph.

the classifier denoted by point $A(0, 1)$ is a perfect classifier, which is able to correctly classify all the points in both positive class and negative class. Classifier represented by point $B(0.4, 0.7)$ achieves 70 percent accuracy for positive class and 60 percent accuracy for negative class. Points on the diagonal, such as $D(0.3, 0.3)$ represent the random classifiers, which provide random guesses about the sample label. However, points below the diagonal, on the other hand, are in no way bad classifiers. In fact, they are consistent to the symmetric points within the upper triangle and can be obtained by reversing the label sign of each sample. Hence, the upper triangle rather than the lower triangle is of our interest. In a word, points close to point A in Figure 7.1 usually represent classifiers with good performance on both positive and negative classes.

7.3 Weighted ELM

Unweighted extreme learning machine (ELM) with non-kernel or kernel hidden nodes have been demonstrated on various datasets, including face recognition [102, 103], protein series [63], etc. The analysis of performance is mainly related to the classification capability of the classifier without considering complex data distribution. This section starts with the analysis of the effect of imbalanced data distribution on the classification performance.

7.3.1 Effect of imbalance on classification performance of ELM

Without loss of generality, consider a binary classification problem with a large amount of samples labeled as negative class and very few samples labeled as positive class. In Figure 7.2 samples from majority class are denoted as the red plus sign and those from minority class are denoted as the blue circle. Intuitively speaking, with the advantage in quantity, the majority class tends to push the separating boundary towards the minority side to gain better classification result for itself.

Mathematically, the intuition can be explained referring to (7.4). There are two terms to minimize: the cumulative training errors for all the samples and the norm of output weight β , related with a trade-off constant C . To minimize β is actually to maximize the marginal distance. 1) When C is small (see Figure 7.2(a)), meaning user is more concerned with maximization of the marginal distance, the separating boundary is supposed to be very close to the minority class in order for maximal marginal class. As a result, minority class might come across a bad generalization performance since their cumulative errors are ignored compared with large amount of majority class samples. 2) When C is relatively large (Figure 7.2(d)), the boundary is curved as a result of more concern on minimization of training errors. The accuracy of minority class is compromised again if the imbalance degree is high since the boundary is slight to the right side instead of in the middle. As a matter of fact, the main cause of the performance compromise is that identical focus is given to two classes provided that the imbalance in quantity already exists. Hence, the straightforward solution is to associate higher C to minority class. The second row of Figure (7.2) illustrates the movement of separating boundary towards the middle, which is supposed to be the ideal boundary, after trade-off C for minority class is set six times compared to majority class. Therefore, it is necessary to propose ELM with weighted trade-off to cope with imbalanced data distribution.

Figure 7.2: Effect of imbalanced class distribution on synthetically generated dataset, with 3 settings of trade-off constant C , where the red cross represents the majority class, the blue circle represents the minority class.

7.3.2 Proposed weighted ELM as binary classifier

Given a set of training data $[\mathbf{x}_i, t_i], i = 1, \dots, N$ belonging to two classes, where t_i is either +1 or -1 to indicate the positive class or the negative class, we define an $N \times N$ diagonal matrix $\mathbf{W}$ associated with every training sample $\mathbf{x}_i$. Usually if $\mathbf{x}_i$ comes from a minority class (assumed to be positive class), the associated weight W_{ii} is relatively larger than others. How to define the specific weight for each sample will be discussed later. To maximize the marginal distance and to minimize the weighted cumulative error with respect to each sample, we have an optimization problem mathematically written as:

$$\text{Minimize: } \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|^2 \text{ and } \|\boldsymbol{\beta}\|, \quad (7.3)$$

where $\mathbf{T} = [t_1, \dots, t_N]$. More precisely,

$$\text{Minimize: } L_{PELM} = \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \mathbf{W} \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2, \quad (7.4)$$

$$\text{Subject to: } \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = t_i^T - \xi_i^T, \quad i = 1, \dots, N.$$

To recall, $\mathbf{h}(\mathbf{x}_i)$ is the feature mapping vector in the hidden layer with respect to $\mathbf{x}_i$, and β represents the output weight vector connecting the hidden layer and output layer. Since it is a binary classifier, there is only one node in the output layer. ξ_i , the training error of sample $\mathbf{x}_i$, is caused by the difference of the desired output t_i and the actual output $\mathbf{h}(\mathbf{x}_i)\beta$.

According to KKT theorem, the equivalent dual optimization problem with respect to (7.4) is

$$L_{DELM} = \frac{1}{2}\|\beta\|^2 + C\mathbf{W}\frac{1}{2}\sum_{i=1}^N \xi_i^2 - \sum_{i=1}^N \alpha_i(\mathbf{h}(\mathbf{x}_i)\beta - t_i + \xi_i), \quad (7.5)$$

where the Lagrange multiplier α_i is the constant factor of sample $\mathbf{x}_i$ in the linear combination to form the final decision function. Further, by making the partial derivatives with respect to variables (β, ξ_i, α_i) all equal to zero, the KKT optimality conditions are obtained

$$\begin{aligned} \frac{\partial L_{DELM}}{\partial \beta} = 0 &\rightarrow \beta = \sum_{i=1}^N \alpha_i \mathbf{h}(\mathbf{x}_i)^T = \mathbf{H}^T \alpha \\ \frac{\partial L_{DELM}}{\partial \xi_i} = 0 &\rightarrow \alpha_i = C\mathbf{W}\xi_i, \quad i = 1, \dots, N \\ \frac{\partial L_{DELM}}{\partial \alpha_i} = 0 &\rightarrow \mathbf{h}(\mathbf{x}_i)\beta - t_i + \xi_i = 0, \quad i = 1, \dots, N. \end{aligned} \quad (7.6)$$

Similar to (2.31), two versions of solutions of β can be derived from (7.6) regarding left pseudo-inverse or right pseudo-inverse. When presented data with small size, right pseudo-inverse is recommended because it involves the inverse of an $N \times N$ matrix. Otherwise, left pseudo-inverse is more suitable since it is much easier to compute matrix inversion of size $L \times L$ when L is much smaller than N .

$$\begin{aligned} \text{When } N \text{ is small: } \beta &= \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}\mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{W}\mathbf{T} \\ \text{When } N \text{ is large: } \beta &= \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{W}\mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{W}\mathbf{T} \end{aligned} \quad (7.7)$$

Given a new sample $\mathbf{x}$, the output function of the ELM classifier is obtained from $f(\mathbf{x}) =$

$\text{sign } \mathbf{h}(\mathbf{x})\beta$

$$\begin{aligned} f(\mathbf{x})_{N \times N} &= \text{sign } \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}\mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{W}\mathbf{T} \\ f(\mathbf{x})_{L \times L} &= \text{sign } \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{W}\mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{W}\mathbf{T}. \end{aligned} \quad (7.8)$$

Inspired from work of [55] and the definition of a kernel (2.38), output function in terms of kernel is naturally derived from the $N \times N$ version

$$\begin{aligned} f(\mathbf{x})_{\text{kernel}} &= \text{sign } \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}\mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{W}\mathbf{T} \\ &= \text{sign } \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}\Omega_{ELM} \right)^{-1} \mathbf{W}\mathbf{T}. \end{aligned} \quad (7.9)$$

7.3.3 Proposed weighted ELM as multiclass classifier

One of the pragmatic advantages of ELM is that a single classifier is capable of multiclass classification task. As far as the network is concerned, the implementation makes use of an SLFN with multiple output nodes. For a given testing sample, the output node with largest decision function value indicates the class label. Assume a set of multiclass training samples $[\mathbf{x}_i, t_i], i = 1, \dots, N$, where $t_i \in [1, m]$. According to the ELM algorithm, each class label is expanded into a label vector of length m . Take $\mathbf{x}_i$ from class two as example, the corresponding label vector is $\mathbf{t}_i = \underbrace{[-1, 1, -1, \dots, -1]}_m$.

Subsequently, to minimize the cumulative error vector ξ_i for every sample and to maximize the marginal distance $2/\|\beta\|$, we have

$$\begin{aligned} \text{Minimize: } L_{PELM} &= \frac{1}{2} \|\beta\|^2 + C\mathbf{W} \frac{1}{2} \sum_{i=1}^N \|\xi_i\|^2, \\ \text{Subject to: } \mathbf{h}(\mathbf{x}_i)\beta &= \mathbf{t}_i^T - \xi_i^T, \quad i = 1, \dots, N, \end{aligned} \quad (7.10)$$

where the label matrix for the whole dataset is differently defined as in binary case $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_N]$ and so is the error vector ξ_i with respect to training sample $\mathbf{x}_i$, $\xi_i = [\xi_1, \dots, \xi_m]$. The solution to the optimization problem above can be found the same as (7.7). Therefore, the output function vector $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \dots, f_m(\mathbf{x})]$ for a testing sample $\mathbf{x}$ is shown in (7.8) as the non-kernel format and (7.9) as the kernel format. To find out the predicted label of $\mathbf{x}$, users just need to refer to a simple equation as below

$$\text{label}(\mathbf{x}) = \arg \max f_i(\mathbf{x}), i \in \{1, \dots, m\}. \quad (7.11)$$

Note that another popular machine learning technique support vector machine (SVM) [89, 83], originally a binary classifier, fails to be applied to the multiclass problems directly without modification. Usually the multiclass problem is decomposed into binary subproblems, implemented by multiple SVMs. This explains the statement about the advantage of ELM classifiers in the beginning of this section.

7.4 Discussions

The previous section provided a solution for imbalanced data learning by adding a weight matrix in extreme learning machine (ELM) to strengthen the impact of minority class while weaken the impact of majority class. The effect of adding the weight to classification performance improvement is illustrated by a binary classification problem. Issues about how to interpret the effect of the weight and determine the value are discussed as well. Finally, a comparison is conducted between the existing weight ELMs and the proposed method.

7.4.1 Essence of the difference between unweighted ELM and weighted ELM

From either the algorithmic or optimization point of view, the learning process of the traditional unweighted ELM is actually to search for a mathematical form to approximate the underlying relationship between the feature of the sample and its label. The solution has to maximize the marginal distance between the two classes while minimize the training errors.

In unweighted ELM, the Lagrange multiplier $\alpha_i = C\xi_i, \forall i$ of $\mathbf{x}_i$, which indicates the weight of fraction in decision function, is proportional to the training error ξ_i of that sample. So that those samples provides more information (with larger training error) are emphasized in the training model.

However, a good approximator must give priority to not only those samples with larger errors, but also those samples which is able to characterize the data distribution. Thus the weighted ELM is proposed based on unweighted ELM to provide additional emphasis to the samples which imply the imbalanced class distribution. As seen from the Lagrange multiplier $\alpha_i = CW\xi_i, \forall i$ of $\mathbf{x}_i$, samples from the minority class is assigned with larger weight W_{ii} such that the information of imbalanced class distribution is well perceived.

7.4.2 Movement of boundary with/without weight

As far as optimization is concerned, the essence of an ELM classifier is to find a boundary to separate data from any disjoint two or multiple parts with maximal marginal distance between any two parts, after the data is mapped into the hidden layer space. In imbalanced data environment, the separating boundary is supposed to be pushed toward the side of minority class, which in fact favors the performance of majority class. The difference of unweighted ELM and weighted ELM can be interpreted from the movement of the separating boundary.

Figure 7.3: The decision boundary produced by unweighted ELM and weighted ELM on dataset *banana* with imbalance ratio 1:43 on Row 1 or 3:43 on Row 2.

Take a binary problem *banana* with two imbalance ratios for example. The separating boundaries produced by ELM using sigmoid feature mapping with/without weight are depicted in Figure 7.3. The first row represents training data consisting of 5 positive samples (red cross) and 215 negative samples (blue circle), with imbalance ratio 1:43. Training data on the second row consists of 15 positive samples and 215 negative samples, with less harsh imbalance ratio 3:43. In addition, to highlight the performance improvement on minority class, misclassified samples from minority class are circled by a black dotted ellipse.

The accuracies for majority class are both 100 percent, both of the accuracies for minority class are 80 percent when unweighted ELM is applied (see Figure 7.3(a) and 7.3(e)). Instead, when applying the weighted ELM, as a result of balancing the impact between minority class and majority class, it is obvious to observe that the boundary is pushed towards the majority class. An unfortunate fact is that the accuracy in minority class is increased in the cost of the decrease in majority class. As can be seen from Figure 7.3(b)-7.3(c) and Figure 7.3(e)-7.3(f), a certain amount of samples from majority class are misclassified in compromise for better accuracy in minority class, 100 percent for

banana (1:43) and 86.67 percent for *banana* (3:43). However, the misclassification rate should not exceed 20 percent, which gives an overall better result in terms of geometric mean.

Interestingly, the boundary movement phenomena helps relate the algorithmic approach and sampling approach in imbalanced data learning. In the proposed weighted ELM, after weighting scheme is applied, the separating boundary is pushed from the minority class towards the majority class. As a matter of fact, some amount of samples from majority class are misclassified as a by-product, which on the other hand is interpreted as that some redundant majority samples are removed. Further, another interpretation is that the minority samples around the boundary is duplicated in the neighborhood, which is one type of oversampling technique.

7.4.3 How to determine the weight

The weight matrix $\mathbf{W} = \text{diag}\{W_{ii}\}, i = 1, \dots, N$ plays an important role in the proposed algorithm. In essence, it determines to what degree of re-balance users are seeking for, and how much further the boundary is pushed towards the majority class.

Users can define W_{ii} for every sample $\mathbf{x}_i$, so that the proposed algorithm actually belongs to the family of cost sensitive learning. For the sake of convenience, we choose a weighting scheme automatically generated from the class information, which is in fact a special case of the cost sensitive learning.

$$\text{Weighting Scheme } \mathbf{W1} : W_{ii} = 1/\#(t_i) \quad (7.12)$$

where $\#(t_i)$ is the number of samples belonging to class t_i , $i = 1, \dots, m$.

After applying the weighting scheme $\mathbf{W1}$, we can roughly imagine the quantity of minority class and majority class is re-balanced into the ratio of $\underbrace{1 : 1 : \dots : 1}_m$.

Another proposed weighting scheme (7.13) is considered to minish the balancing step into the ratio of 0.618:1 (here the author adopts the value of golden standard that represents the perfection in nature) between minority classes and the majority classes. In binary classification, the minority class is assumed to be the positive class and the majority class is assumed to be the negative class. In multiclass classification, the minorities classes refer to those classes whose number is below the average samples per class and the majority classes are those with sample number above the average. As a result, compared to weighting scheme **W1** the boundary is pushed slightly backwards the minority class so that the misclassification cases in compromise on the majority side is sought of alleviated.

$$\text{Weighting Scheme } \mathbf{W2} \quad \begin{cases} W_{ii} = 0.618/\#(t_i), & \text{if } t_i > \text{AVG}(t_i) \\ W_{ii} = 1/\#(t_i), & \text{if } t_i \leq \text{AVG}(t_i) \end{cases} \quad (7.13)$$

7.4.4 Difference with other weighted ELMs and weighted least square algorithms

In literature, two weighted ELMs were proposed by Toh [86] and Deng, et al [25], though none is targeting the imbalanced data problem. Moreover, neither of them has mentioned the feasibility of “generalized” feature mappings or kernels.

Deng, et al [25] proposed a weighted regularized ELM (WR-ELM) which is similar to the proposed method (7.7) in the presence of large training data size. But difference in training procedure, weighting scheme and feature mapping between WR-ELM and the proposed weighted ELM is obvious. 1) Compared with the traditional three-step training procedure of ELM using only one classifier, two classifiers are used in the training phase. The network model is initially obtained by training the data with an *unweighted* ELM classifier. Afterwards, another weighted ELM (7.14) is applied to the training data to update the model on the basis of the error information obtained from the *unweighted* ELM. 2) The weighting scheme in WR-ELM is defined with respect to the

training error (7.15). More precisely, the weight of each sample is inversely proportional to the training error, so that samples with high training error are suppressed to prevent the effect of outliers. 3) Only sigmoid additive neural node is considered in their work. And WR-ELM mainly focuses on the case when the training data outnumber the network size (when $L < N$)

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T \mathbf{W}^2 \mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{W}^2 \mathbf{T}, \quad (7.14)$$

where $\mathbf{W} = \text{diag}\{v_1, v_2, \dots, v_N\}$ and

$$v_i = \begin{cases} 1, & |\xi_i/\hat{s}| \leq c_1 \\ \frac{c_2 - |\xi_i/\hat{s}|}{c_2 - c_1}, & c_1 \leq |\xi_i/\hat{s}| \leq c_2 \\ 10^{-4}, & |\xi_i/\hat{s}| > c_2 \end{cases} \quad (7.15)$$

The error information after training data with the first *unweighted* ELM includes, training error ξ_i for sample $\mathbf{x}_i$, robust estimate $\hat{s}$ of the standard deviation of ξ_i . c_1 and c_2 are assigned empirical values as 2.5 and 3, respectively.

The total error rate (TER) ELM [86] only defined the binary weighted ELM similar to (7.7), with weighting scheme same as $\mathbf{W}1$. To deal with multiclass tasks, one-against-all (OAA) method was utilized which is obviously complicated than a single classifier. Take an m -class problem for example, m binary TER-ELMs are required where the i th TER-ELM assumes the i th class as positive class and the rest $m - 1$ classes as the negative class. Suppose there are m_i^+ number of samples in the positive class, and m_i^- in the negative class. The output of samples from positive class is assigned $y_i^+ = (a + b)$, and $y_i^- = (a - b)$ for the negative class, where the threshold a and bias b is specified by users. With the preprocessed training data, the solution of the i th binary classifier is

$$\begin{aligned} \text{When } N \text{ is small: } \beta_i &= \mathbf{H}_i^T \left(\frac{\mathbf{I}}{C} + \mathbf{W}_i \mathbf{H}_i \mathbf{H}_i^T \right)^{-1} \mathbf{W}_i \mathbf{y}_i \\ \text{When } N \text{ is large: } \beta_i &= \left(\frac{\mathbf{I}}{C} + \mathbf{H}_i^T \mathbf{W}_i \mathbf{H}_i \right)^{-1} \mathbf{H}_i^T \mathbf{W}_i \mathbf{y}_i, \end{aligned} \quad (7.16)$$

where $\mathbf{W}_i = \mathbf{W}_i^+ + \mathbf{W}_i^-$ and $\mathbf{W}_i^+ = \text{diag}(0, \dots, 0, 1/m_i^+, \dots, 1/m_i^+)$, $\mathbf{W}_i^- = \text{diag}(0, \dots, 0, 1/m_i^-, \dots, 1/m_i^-)$.

The hidden layer output matrix $\mathbf{H}_i$ and the output vector $\mathbf{y}_i$ are in the order of negative class after positive class, that is to say, all the non- j th classes after j th class. An interesting point is that the strategy OAA itself creates an synthetic imbalance between positive and negative class, which may render the whole data space in a complex class distribution. Furthermore, although TER-ELM was concluded with the same solution as (7.7), kernels and “generalized” feature mapping was not mentioned.

In [42], a two-class kernel classifier was proposed for imbalanced learning problems, where the weighted regularized orthogonal least square algorithm (ROWLS) was used to train the classifier. Although the proposed weighted ELM (WELM) shares the same regularized least square cost function as work in [42], there are three main aspects that differentiate these two works. (1) ROWLS are kernel based classifiers. But in WELM, the hidden node function $h()$ might be known or unknown. That is to say, to map the original data into the feature space, a large variety of hidden node functions $h()$ or kernel functions (if $h()$ is unknown) are free to select; (2) ROWLS only considered binary classification problems. While WELM is able to tackle multi-class problems directly; (3) The least square solution in WELM is actually an analytical solution, which requires limited and fixed steps. While ROWLS tuned and obtained the solution in an iterative manner, which is time consuming compared with analytical solutions. However, an advantage of ROWLS is that, during iterative training only a subset of data samples are selected to form the regressors (kernel functions), which is particularly beneficial for large scale dataset.

7.5 Performance Evaluation

7.5.1 Data specification

The weighted extreme learning machine (ELM) was proposed based on unweighted ELM in the previous section to cope with data with a large variety of imbalanced class distribution. To verify the theoretical analysis between unweighted ELM and weighted ELM (with two weighting schemes defined for both binary and multiclass classification), 48 binary datasets and 4 multiclass datasets are tested in the experiments.

To quantitatively measure the imbalance degree of a dataset, the imbalance ratio (IR) is defined

$$\begin{aligned} \text{Binary: } \quad \text{IR} &= \frac{\#(+1)}{\#(-1)}, \\ \text{Multiclass: } \text{IR} &= \frac{\text{Min } \#(t_i)}{\text{Max } \#(t_i)}, \quad i = 1, \dots, m. \end{aligned} \tag{7.17}$$

Most of the binary datasets with 5-folder cross validations are downloaded online ¹ as displayed in Table 7.5.1 [3]. As shown in Table 7.5.1, the imbalance ratio of the dataset can be as low as 0.0078, and the highest imbalance ratio happens to be 0.5380. 4 binary datasets and 5 multiclass datasets from UCI [30] are displayed in Table 7.5.1 where the training and testing part can not be shuffled. The imbalance ratio of datasets from UCI varies from 0.0002 to 1.

The attributes of the datasets are normalized into $[-1, 1]$.

¹<http://sci2s.ugr.es/keel/study.php?cod=24>

Table 7.1: Specification of binary classification problems.

Data sets	# of Attributes	# of TrainData	# of TestData	Imbalance Ratio
ecoli1	7	268	68	0.2947
ecoli2	7	268	68	0.1806
ecoli3	7	268	68	0.1167
ecoli4	7	268	68	0.0635
yeast1	8	1187	297	0.4064
yeast3	8	1187	297	0.1230
yeast4	8	1187	297	0.0349
yeast5	8	1187	297	0.0304
yeast6	8	1187	297	0.0243
yeast-0-5-6-7-9_vs_4	8	422	106	0.1047
yeast-1-2-8-9_vs_7	8	757	188	0.0327
yeast-1-4-5-8_vs_7	8	554	139	0.0453
yeast-1_vs_7	7	367	92	0.0700
yeast-2_vs_4	8	411	103	0.1078
yeast-2_vs_8	8	385	97	0.0434
glass0	9	173	43	0.4786
glass1	9	171	43	0.5405
glass2	9	171	43	0.0823
glass4	9	171	43	0.0621
glass5	9	171	43	0.0427
glass6	9	171	43	0.1554
glass-0-1-2-3_vs_4-5-6	9	171	43	0.3053
glass-0-1-6_vs_2	9	153	39	0.0929
glass-0-1-6_vs_5	9	147	37	0.0500
iris0	4	120	30	0.5000
vowel0	13	790	198	0.1002
new-thyroid1	5	172	43	0.1944
new-thyroid2	5	172	43	0.1944
shuttle-C0_vs_C4	9	1463	366	0.0726
shuttle-C2_vs_C4	9	103	26	0.0404
segment0	19	1846	462	0.1661
vehicle0	18	676	170	0.3075
vehicle1	18	676	170	0.3439
vehicle2	18	676	170	0.3466
vehicle3	18	676	170	0.3333
page-blocks0	10	4377	1095	0.1137
page-blocks1	10	377	95	0.0620
abalone9_18	8	584	147	0.0600
abalone19	8	3339	835	0.0078
wisconsin	9	546	137	0.5380
haberman	3	244	62	0.3556
pima	8	614	154	0.5350

Table 7.2: Specification of binary and multiclass problems from UCI.

Data sets	# of Attributes	# of Classes	# of TrainData	# of TestData	Imbalance Ratio
Adult	123	2	4781	27780	0.3306
Banana	2	2	400	4900	0.8605
Colon(Gene Sel)	60	2	30	32	0.6667
Leukemia(Gene Sel)	60	2	38	34	0.4074
DNA	180	3	2000	1186	0.4415
Satimage	36	6	4435	2000	0.3871
Shuttle	9	7	43500	14500	0.0002
USPS	256	10	7291	2007	0.4733
Vowel	10	11	528	462	1

7.5.2 Parameter settings

In ELM theory, the feature mapping can be almost any nonlinear continuous piecewise function [51], or even in the form of a kernel [55]. In the experiments, we test the algorithms on Sigmoid additive node and Gaussian kernel, which are popular choices by researchers. There are two parameters to tune for ELMs with sigmoid additive node $G(\mathbf{a}, b, \mathbf{x}) = 1/(1 + \exp(-(\mathbf{a} \cdot \mathbf{x} + b)))$: the trade-off constant C and the number of hidden nodes L . A grid search of C on $\{2^{-18}, 2^{-16}, \dots, 2^{48}, 2^{50}\}$ and L on $\{10, 20, \dots, 990, 1000\}$ is conducted in seek of the optimal result. ELMs with Gaussian Kernel, which is expressed as $K(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma|\mathbf{x} - \mathbf{x}_i|^2)$, consists of two user specified parameters, the trade-off constant C and the kernel parameter γ . C is searched in a wide range $\{2^{-18}, 2^{-16}, \dots, 2^{48}, 2^{50}\}$ while γ is searched in the range of $\{2^{-18}, 2^{-16}, \dots, 2^{18}, 2^{20}\}$.

The effect of user specified parameters on classification performance is shown in Figure 7.4. It is obvious that the performance in terms of G-mean is not sensitive to the number of hidden nodes L in Sigmoid node. When it comes to the Gaussian kernel, it is found that the performance is sensitive to both parameters C and γ , which is consistent with the findings in [52, 55].

It can be easily spotted from Figure 7.4(c) that when C is small, usually below 2^0 in the experiments, very poor generalization performance is expected. In most of the cases G-mean value drops to as low as 0 when small C is assigned with a network of very few hidden nodes. The reason of the significant performance loss can be traced back to the optimization formula (7.4). When C is assigned a small value, the model of users' interest is mainly focusing on maximization of the marginal distance, which implies that the cumulative errors of minority class are negligible comparing with the large amount of majority class. An extreme case might happen that all the minority class is classified as the majority class, thus to cause the poor performance in terms of G-mean.

It can be also observed that the performance is decreasing when L is increasing given quite a large C , say 2^{50} , which can be viewed as an overfitting case. With reasonable

Figure 7.4: Display of G-mean distribution of yeast1 with different choices of C and L for Sigmoid node or C and γ for Gaussian kernel.

value of C , the generalization performance of the proposed method using Sigmoid node is insensitive to number of hidden nodes. Therefore, as the tradition of ELM, for the sake of convenience one can fix L as a large value and tune only C in Sigmoid node without affecting the performance.

7.5.3 Experiment results

Sigmoid node VS. Gaussian kernel

By comparing Table 7.3 and Table 7.4 or observed from Figure 7.5, the results of ELM with Sigmoid node or Gaussian kernel are quite similar. In a few cases, one is slightly better than another while in some other cases slightly worse. And it is encouraged to

explore more efficient nodes or kernels in the future work.

Unweighted ELM VS. weighted ELM

From Figure 7.5, it can be seen that, for most datasets, after applying weighting scheme in ELM, G-mean value increases significantly in the cost of relatively slight decreasing in the overall accuracy. That is to say, the separating boundary is moved towards the majority class so that additional minority samples are correctly classified, but slightly more majority samples are misclassified. This trade-off phenomenon is visible especially when imbalance degree is high. And in the case of *banana* (Figure 7.5(c)) where the classes are quite balanced, the results of unweighted ELM and weighted ELM are more or less. Therefore, the proposed weighted ELM is applicable to not only the imbalanced datasets, but also the well balanced datasets. In fact, the proposed weighted ELM can be viewed as the general solution that is robust to data with different imbalance degrees.

Weighted scheme 1 VS. weighted scheme 2

Figure 7.6 presents the ROC graph for Unweighed ELM, weighted ELM with weighting scheme **W1** and **W2** on three datasets with imbalance ratio in an ascending order. Each point represents one classifier and provides the accuracy information of both positive class and negative class.

In extremely imbalanced case (7.6(a)), as predicted the true positive rate is greatly increased after applying the weighting scheme in ELM. And the true negative rate is slightly increased as well which happens in some of the cases and the author believes it is closely related to the data complexity. The results of weighted ELM **W1** and **W2** are close. However, compared to weighted ELM **W1**, weighted ELM **W2** has slightly better true negative rate and slightly worse true positive rate, which is in the middle of unweighted ELM and weighted ELM **W1** (maybe more close to the latter).

Figure 7.6(b) is a typical case of the boundary movement theory analyzed in the last section. The true positive rate is increased in the compromise of decrease in true negative rate for both weighted ELMs. The weighting scheme **W1** pushes the boundary a little much towards the majority side so that additional minority samples are correctly classified but more majority samples are misclassified.

Figure 7.6(c) shows a case with well balanced class distribution, where the results of three classifiers are found quite close. That is to say, the proposed weighted ELM is able to handle well balanced data, whichever weighting scheme might be used.

In a word, observed also from Table 7.3 - 7.4, the weighting scheme **W2** is supposed to alleviate the compromise in majority class caused by boundary movement, while attempt to maintain the good performance in minority class than unweighted ELM. That is to say, it is the trade-off of unweighted ELM and weighted ELM **W1**. The choice of weighting schemes are up to the users' requirements.

Statistical test result

To further compare the results between ELM with Sigmoid hidden nodes or Gaussian Kernel, with weighting scheme 1 or weighting scheme 2, the pairwise t-test is used. Usually a small p-value indicates the fact that, one algorithm is significant better than the other one. The common threshold is set as 0.05 or 0.01. Therefore, it is obvious that the performance of original ELM in terms of G-mean is significantly improved after weighting scheme is applied.

Data complexity VS. imbalance ratio

In the literature of imbalanced learning [36, 94], an interesting point was mentioned again and again : the data complexity plays an more important role in the performance than the imbalance ratio. That is to say, the effect of imbalance ratio on easily separable

data distribution is minor. On the contrary, complex data distribution may render the classification performance easily affected by the imbalance ratio.

Although the topic of data complexity is out of the scope of this paper, this point of view is verified with the proposed weighted ELM on datasets with wide range of imbalance ratio. Datasets *glass-0-1-6_vs_5*, *page-blocks1*, *shuttle-C0_vs_C4*, *glass6*, *segment0*, *vehicle0*, *vehicle2*, *glass0*, *iris0*, *pima*, *wisconsin* and *banana* are typical datasets with minor difference with/without weighting. But the significant improvement in terms of G-mean for the rest datasets are observed after the proposed weighting scheme is applied.

Multiclass imbalanced learning

Finally we would like to note that, so far although main focus in this paper is given to the binary classification, multiclass imbalanced learning is of equal importance. Therefore, the weighted ELM is naturally extended to multiclass problems with two weighting schemes as in binary problems. A couple of multiclass datasets from UCI are tested in the experiments, see Table 7.5. Two weighting schemes **W1** (7.12) and **W2** (7.13) are proposed for multiclass problems as well.

The same observations of binary classifiers apply to multiclass classifiers. In particular, there is one dataset named *Shuttle* with imbalance ratio 0.0002, which witnesses the great improvement in the G-mean value. The proposed weighted ELM is verified to be efficient on the multiclass classification problems as well.

Performance result in terms of accuracy

Performance result in terms of accuracy of all the binary problems and multiclass problems is shown in Table 7.7, Table 7.8 and Table 7.5 for interested readers.

Table 7.3: Performance result of binary problems with imbalance ratio $\in (0, 0.2)$.

Data G-mean	Sigmoid Node						Gaussian Kernel					
	Unweighted ELM		Weighted ELM W1		Weighted ELM W2		Unweighted ELM		Weighted ELM W1		Weighted ELM W2	
	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)
abalone19	(2^{42} , 990)	47.52	(2^6 , 150)	77.19	(2^{36} , 40)	64.27	(2^{40} , 2^0)	51.36	(2^{24} , 2^{16})	74.47	(2^{24} , 2^{12})	68.35
yeast6	(2^{44} , 350)	70.77	(2^{34} , 20)	87.77	(2^{14} , 900)	88.29	(2^0 , 2^{-8})	75.66	(2^4 , 2^{-6})	89.81	(2^2 , 2^{-6})	90.36
yeast5	(2^{36} , 900)	81.04	(2^{30} , 100)	95.39	(2^2 , 200)	95.15	(2^4 , 2^{-6})	87.34	(2^4 , 2^{-8})	98.18	(2^{-18} , 2^{-8})	98.14
yeast-1-2-8-9_vs_7	(2^{42} , 880)	59.23	(2^{42} , 20)	75.83	(2^{50} , 10)	73.92	(2^{48} , 2^{10})	60.97	(2^4 , 2^{-2})	71.41	(2^{20} , 2^{10})	68.98
yeast4	(2^{42} , 960)	65.52	(2^6 , 20)	87.29	(2^6 , 100)	82.18	(2^{-6} , 2^{-16})	70.47	(2^2 , 2^{-4})	84.98	(2^6 , 2^{-2})	83.12
shuttle-C2_vs_C4	(2^{40} , 20)	93.54	(2^{31} , 10)	100	(2^{24} , 10)	100	(2^{-18} , 2^{-12})	94.14	(2^{-18} , 2^0)	100	(2^{24} , 2^4)	99.60
glass5	(2^{20} , 90)	90.81	(2^{10} , 110)	95.99	(2^{10} , 40)	96.60	(2^{18} , 2^6)	93.16	(2^{10} , 2^4)	96.51	(2^{14} , 2^6)	97.26
yeast-2_vs_8	(2^0 , 290)	72.83	(2, 960)	75.56	(2^8 , 60)	76.01	(2^{24} , 2^{-4})	77.24	(2^{14} , 2^{-4})	77.89	(2^{18} , 2^{-2})	78.00
yeast-1-4-5-8_vs_7	(2^{44} , 970)	61.07	(2^{10} , 120)	67.10	(2^{42} , 800)	64.26	(2^{24} , 2^{-4})	59.89	(2^0 , 2^{-8})	69.32	(2^{40} , 2^2)	65.68
glass-0-1-6_vs_5	(2^{18} , 660)	92.41	(2^6 , 960)	98.55	(2^6 , 960)	98.70	(2^{22} , 2^4)	98.55	(2^8 , 2^2)	98.85	(2^8 , 2^2)	98.85
abalone9_18	(2^{40} , 150)	75.29	(2^{16} , 70)	87.99	(2^{32} , 20)	88.72	(2^{18} , 2^0)	72.71	(2^{28} , 2^{14})	89.76	(2^{14} , 2^4)	86.83
page-blocks1	(2^{10} , 440)	97.78	(2^{30} , 30)	98.98	(2^{44} , 50)	98.98	(2^2 , 2^{-2})	97.89	(2^{20} , 2^{10})	98.07	(2^{28} , 2^{12})	98.64
glass4	(2^{34} , 30)	85.72	(2^{12} , 120)	91.34	(2^{10} , 140)	91.46	(2^{-4} , 2^{-2})	85.93	(2^8 , 2^2)	91.17	(2^{-2} , 2^4)	91.17
ecoli4	(2^{22} , 60)	91.96	(2^6 , 180)	97.83	(2^6 , 200)	95.90	(2^{46} , 2^{16})	94.32	(2^6 , 2^0)	98.24	(2^0 , 2^{-2})	97.42
yeast-1_vs_7	(2^{40} , 960)	65.58	(2^{16} , 550)	77.26	(2^{14} , 350)	76.91	(2^{14} , 2^{-4})	64.48	(2^{20} , 2^4)	77.72	(2^{14} , 2^2)	74.61
shuttle-C0_vs_C4	(2^{14} , 10)	100	(2^{38} , 10)	100	(2^{28} , 10)	100	(2^{14} , 2^0)	100	(2^{26} , 2^0)	100	(2^{26} , 2^0)	100
glass2	(2^{28} , 110)	79.49	(2^{22} , 140)	80.33	(2^{24} , 130)	83.34	(2^{16} , 2^{-2})	66.28	(2^{14} , 2^2)	82.59	(2^{14} , 2^4)	80.26
glass-0-1-6_vs_2	(2^{34} , 150)	67.78	(2^{14} , 380)	83.77	(2^{22} , 140)	83.06	(2^{50} , 2^{18})	63.20	(2^{32} , 2^8)	83.59	(2^{50} , 2^{16})	81.34
vowel0	(2^{28} , 110)	100	(2^{50} , 120)	100	(2^{40} , 100)	100	(2^{-18} , 2^{-10})	100	(2^{-18} , 2^{-10})	100	(2^{-18} , 2^{-10})	100
yeast-0-5-6-7-9_vs_4	(2^{32} , 390)	64.49	(2^{-2} , 150)	81.05	(2^{14} , 290)	80.56	(2^{-6} , 2^{-16})	68.68	(2^4 , 2^{-6})	82.21	(2^2 , 2^{-6})	80.32
yeast-2_vs_4	(2^{36} , 280)	86.25	(2^{26} , 940)	91.56	(2^{30} , 130)	90.02	(2^6 , 2^{-4})	85.31	(2^{32} , 2^4)	91.88	(2^{14} , 2^{-2})	90.25
page-blocks0	(2^{34} , 830)	89.92	(2^{22} , 750)	93.21	(2^{24} , 820)	93.40	(2^4 , 2^{-4})	89.62	(2^{16} , 2^0)	93.61	(2^{16} , 2^0)	92.98
ecoli3	(2^{44} , 70)	77.38	(2^{46} , 10)	90.17	(2^6 , 30)	90.00	(2^2 , 2^{-2})	76.78	(2^{10} , 2^4)	89.60	(2^2 , 2^{-2})	90.14
yeast3	(2^{40} , 100)	80.75	(2^{16} , 700)	93.25	(2^{16} , 440)	91.08	(2^{44} , 2^8)	84.13	(2^{48} , 2^{16})	93.11	(2^{16} , 2^2)	90.94
glass6	(2^{46} , 450)	94.96	(2^{44} , 30)	95.72	(2^{26} , 190)	95.90	(2^8 , 2^2)	92.60	(2^{30} , 2^8)	94.04	(2^{30} , 2^8)	94.04
segment0	(2^8 , 720)	99.24	(2^{18} , 870)	99.75	(2^{22} , 350)	99.70	(2^4 , 2^{-4})	99.57	(2^{12} , 2^0)	99.77	(2^{12} , 2^0)	99.82
ecoli2	(2^{36} , 60)	91.17	(2^{28} , 40)	93.91	(2^{30} , 20)	94.51	(2^{-18} , 2^{-6})	94.31	(2^8 , 2^{-2})	94.09	(2^8 , 2^{-2})	94.43
new-thyroid1	(2^{18} , 180)	98.24	(2^{18} , 30)	99.44	(2^{20} , 20)	99.72	(2^0 , 2^{-4})	99.16	(2^{14} , 2^0)	99.72	(2^6 , 2^0)	99.72
new-thyroid2	(2^{18} , 40)	95.55	(2^{16} , 290)	99.72	(2^{12} , 480)	100	(2^2 , 2^{-4})	99.44	(2^{12} , 2^{-4})	99.72	(2^{14} , 2^0)	99.72

Table 7.4: Performance result of binary problems with imbalance ratio $\in (0.2, 1)$.

Datasets G-mean	Sigmoid Node						Gaussian Kernel					
	Unweighted ELM		Weighted ELM W1		Weighted ELM W2		Unweighted ELM		Weighted ELM W1		Weighted ELM W2	
	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)
ecoli1	(2^{16} , 140)	87.77	(2^4 , 320)	90.69	(2^{22} , 40)	90.26	(2^0 , 2^{-4})	88.75	(2^{10} , 2^4)	91.04	(2^{50} , 2^{18})	90.25
glass-0-1-2-3_vs_4-5-6	(2^8 , 140)	90.67	(2^{10} , 290)	94.68	(2^{12} , 100)	93.14	(2^{10} , 2^{-2})	93.26	(2^{-18} , 2^{-4})	95.41	(2^{-18} , 2^{-4})	95.41
vehicle0	(2^{10} , 460)	98.57	(2^{16} , 850)	99.32	(2^{16} , 850)	99.04	(2^8 , 2^2)	99.36	(2^{16} , 2^2)	99.30	(2^{20} , 2^4)	99.36
adult	(2^{10} , 580)	73.86	(2^4 , 840)	81.67	(2^6 , 1000)	80.42	(2^5 , 2^5)	73.54	(2^{10} , 2^5)	81.62	(2^{20} , 2^{10})	80.37
vehicle3	(2^8 , 450)	78.15	(2^{14} , 710)	85.13	(2^{14} , 340)	84.34	(2^{16} , 2^4)	72.17	(2^{24} , 2^8)	92.45	(2^{20} , 2^6)	84.90
vehicle1	(2^8 , 570)	79.29	(2^{14} , 450)	85.30	(2^{12} , 450)	83.44	(2^{18} , 2^8)	80.60	(2^{24} , 2^8)	86.74	(2^{26} , 2^8)	85.72
vehicle2	(2^{12} , 600)	98.43	(2^{16} , 800)	99.12	(2^{16} , 550)	98.78	(2^8 , 2^0)	99.22	(2^{20} , 2^2)	99.45	(2^{22} , 2^4)	99.38
haberman	(2^{44} , 910)	49.16	(2^{34} , 10)	65.11	(2^{18} , 90)	59.26	(2^{42} , 2^0)	57.23	(2^{14} , 2^4)	66.26	(2^{32} , 2^6)	60.91
yeast1	(2^{44} , 300)	63.26	(2^{26} , 120)	72.57	(2^{32} , 250)	70.32	(2^0 , 2^{-6})	65.45	(2^{10} , 2^{-4})	73.17	(2^{10} , 2^{-4})	70.33
Leukemia(Gene Sel)	(2^{10} , 100)	100	(2^{10} , 100)	100	(2^{10} , 100)	100	(2^{-18} , 2^{-2})	100	(2^{-18} , 2^{-2})	100	(2^{-18} , 2^{-2})	100
glass0	(2^{14} , 950)	79.61	(2^{22} , 800)	81.17	(2^{22} , 710)	82.62	(2^0 , 2^{-6})	85.35	(2^0 , 2^{-6})	85.65	(2^4 , 2^{-6})	85.59
iris0	(2^{-2} , 10)	100	(2^2 , 10)	100	(2^2 , 10)	100	(2^{-18} , 2^{-10})	100	(2^{-18} , 2^{-10})	100	(2^{-18} , 2^{-10})	100
pima	(2^{32} , 30)	70.10	(2^{14} , 20)	74.74	(2^{12} , 40)	71.51	(2^0 , 2^2)	71.16	(2^8 , 2^2)	75.58	(2^{10} , 2^0)	72.52
wisconsin	(2^{34} , 50)	96.32	(2^{34} , 60)	97.07	(2^{10} , 270)	96.97	(2^{-2} , 2^2)	97.18	(2^8 , 2^4)	97.70	(2^6 , 2^2)	97.18
glass1	(2^{16} , 440)	78.36	(2^{22} , 900)	78.31	(2^{22} , 240)	79.32	(2^{-18} , 2^{-6})	77.48	(2^2 , 2^{-4})	80.35	(2^{-18} , 2^{-6})	77.48
Colon(Gene Sel)	(2^{-12} , 940)	85.28	(2^{-8} , 220)	83.12	(2^{-6} , 320)	85.28	(2^{-18} , 2^8)	85.28	(2^6 , 2^6)	83.12	(2^{-18} , 2^8)	85.28
banana	(2^{24} , 50)	88.98	(2^{34} , 50)	89.13	(2^{26} , 90)	89.04	(2^2 , 2^0)	89.33	(2^{12} , 2^0)	89.26	(2^8 , 2^0)	88.87

Table 7.5: Performance result of multiclass problems.

<div> <div>Datasets</div> <div>G-mean</div> </div>	Sigmoid Node						Gaussian Kernel					
	Unweighted ELM		Weighted ELM W1		Weighted ELM W2		Unweighted ELM		Weighted ELM W1		Weighted ELM W2	
	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)
DNA	(2 ⁻⁶ , 980)	94.14	(2 ² , 940)	94.10	(2 ² , 1000)	93.93	(2 ⁶ , 2 ⁶)	96.10	(2 ¹⁶ , 2 ⁶)	96.10	(2 ¹⁶ , 2 ⁶)	96.10
Satimage	(2 ⁸ , 780)	87.64	(2 ³⁶ , 980)	89.56	(2 ¹⁴ , 840)	89.03	(2 ⁵ , 2 ⁰)	89.34	(2 ¹⁰ , 2 ⁰)	90.62	(2 ¹² , 2 ⁻¹)	89.24
Shuttle	(2 ³⁰ , 200)	60.38	(2 ²⁴ , 950)	84.89	(2 ²⁶ , 400)	82.02	(2 ¹⁵ , 2 ⁻⁶)	65.33	(2 ²⁰ , 2 ⁻¹⁴)	91.98	(2 ²⁰ , 2 ⁻¹⁴)	92.00
USPS	(2 ²⁸ , 950)	96.39	(2 ²⁸ , 950)	96.36	(2 ⁴² , 750)	96.28	(2 ⁵ , 2 ⁸)	98.64	(2 ¹⁵ , 2 ⁷)	98.33	(2 ¹⁵ , 2 ⁷)	98.33
<div> <div>Datasets</div> <div>Accuracy</div> </div>	Sigmoid Node						Gaussian Kernel					
	Unweighted ELM		Weighted ELM W1		Weighted ELM W2		Unweighted ELM		Weighted ELM W1		Weighted ELM W2	
	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)
DNA	(2 ⁻⁶ , 640)	94.35	(2 ⁴ , 960)	93.09	(2 ⁴ , 1000)	94.10	(2 ⁶ , 2 ⁶)	96.29	(2 ²⁰ , 2 ¹⁵)	96.04	(2 ¹⁶ , 2 ⁶)	96.29
Satimage	(2 ⁸ , 780)	90.45	(2 ³⁶ , 980)	90.00	(2 ¹⁴ , 900)	90.40	(2 ⁴ , 2 ⁻²)	92.35	(2 ¹⁰ , 2 ⁰)	91.25	(2 ¹² , 2 ⁰)	91.65
Shuttle	(2 ³² , 350)	99.74	(2 ⁴⁴ , 750)	98.52	(2 ⁴⁴ , 1000)	98.75	(2 ²⁰ , 2 ⁻¹⁰)	99.91	(2 ²⁰ , 2 ⁻⁸)	98.70	(2 ²⁰ , 2 ⁻⁸)	99.43
USPS	(2 ²⁸ , 950)	97.11	(2 ²⁸ , 950)	96.96	(2 ²² , 1000)	96.76	(2 ⁴ , 2 ⁸)	98.90	(2 ¹⁶ , 2 ⁷)	98.60	(2 ¹⁶ , 2 ⁷)	98.60

Table 7.6: P-value of the t-test for pairwise comparisons between the algorithms over binary problems.

P-value	Sigmoid	Sigmoid+W1	Sigmoid+W2	Gaussian	Gaussian+W1
Sigmoid					
Sigmoid+W1	8.8054e-008				
Sigmoid+W2	1.1444e-008	0.0306			
Gaussian	0.0674	1.2105e-006	1.5216e-006		
Gaussian+W1	8.0126e-009	0.0478	6.9523e-004	1.7090e-007	
Gaussian+W2	1.6047e-008	0.0688	0.4570	2.5454e-007	1.6146e-004

7.6 Summary

In this work a weighted ELM based on original unweighted ELM was proposed for binary and multiclass classification tasks. To benefit from original unweighted ELM framework, the proposed method is simple in theory and implementation; wide types of feature mapping or kernels are available as options; comparable or better generalization performance is achievable compared to the conventional machine learning techniques. On the other hand, by incorporating the information of imbalance class distribution, the proposed method is able to deal with data with imbalanced class distribution. Two weighting schemes are tested in the experiments on datasets with various imbalance ratios, which demonstrates the better performance of weighted ELM compared with unweighted ELM.

Table 7.7: Performance result in terms of accuracy for binary problems with imbalance ratio $\in (0, 0.2)$.

Data	Sigmoid Node						Gaussian Kernel					
	Unweighted ELM		Weighted ELM W1		Weighted ELM W2		Unweighted ELM		Weighted ELM W1		Weighted ELM W2	
	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)
abalone19	(2 ⁻¹⁸ , 10)	99.23	(2 ⁴⁰ , 980)	93.56	(2 ⁻¹⁸ , 10)	99.23	(2 ⁻¹⁶ , 2 ⁻⁴)	99.23	(2 ⁻¹⁶ , 2 ⁻¹⁶)	98.80	(2 ⁻¹⁶ , 2 ⁴)	99.23
yeast6	(2 ¹⁶ , 400)	98.25	(2 ³² , 820)	92.59	(2 ⁻¹⁸ , 20)	97.64	(2 ⁰ , 2 ⁻⁴)	98.25	(2 ¹⁸ , 2 ⁻⁸)	97.71	(2 ¹⁸ , 2 ⁻⁸)	97.71
yeast5	(2 ²⁰ , 860)	97.91	(2 ⁻¹² , 40)	96.63	(2 ² , 60)	97.51	(2 ⁻² , 2 ⁻⁶)	98.59	(2 ¹⁴ , 2 ⁻⁸)	98.38	(2 ¹⁴ , 2 ⁻⁸)	98.38
yeast-1-2-8-9_vs_7	(2 ¹⁶ , 210)	97.05	(2 ⁻¹² , 120)	88.26	(2 ⁰ , 710)	96.94	(2 ⁴ , 2 ⁻⁴)	97.04	(2 ¹² , 2 ⁻⁸)	95.67	(2 ⁻¹⁸ , 2 ⁻²)	96.83
yeast4	(2 ²⁶ , 40)	96.90	(2 ⁴² , 700)	90.43	(2 ⁻⁸ , 960)	96.63	(2 ⁻² , 2 ⁻⁴)	96.97	(2 ¹⁴ , 2 ⁻⁸)	96.16	(2 ⁰ , 2 ⁰)	96.63
shuttle-C2_vs_C4	(2 ³² , 100)	99.23	(2 ³¹ , 10)	100	(2 ⁸ , 10)	100	(2 ⁻¹⁸ , 2 ⁻¹²)	99.23	(2 ⁻¹⁸ , 2 ⁰)	100	(2 ⁻¹⁸ , 2 ⁻¹²)	99.23
glass5	(2 ¹² , 90)	98.14	(2 ¹⁸ , 160)	98.14	(2 ¹⁶ , 220)	98.14	(2 ⁴ , 2 ⁰)	98.61	(2 ²⁰ , 2 ⁰)	98.14	(2 ¹² , 2 ⁰)	98.14
yeast-2_vs_8	(2 ⁻² , 160)	97.92	(2 ⁴ , 60)	98.13	(2 ⁶ , 120)	98.13	(2 ⁰ , 2 ⁰)	97.92	(2 ² , 2 ⁻²)	97.93	(2 ⁴ , 2 ⁻²)	98.13
yeast-1-4-5-8_vs_7	(2 ⁻¹⁸ , 10)	95.67	(2 ⁻⁶ , 50)	89.91	(2 ⁻¹⁸ , 10)	95.67	(2 ⁰ , 2 ⁻⁴)	95.82	(2 ¹² , 2 ⁻⁸)	94.81	(2 ⁻¹⁸ , 2 ⁻²)	95.67
glass-0-1-6_vs_5	(2 ³⁴ , 30)	98.36	(2 ¹⁰ , 80)	97.84	(2 ²² , 560)	98.36	(2 ⁻¹⁸ , 2 ⁻¹²)	97.30	(2 ¹⁴ , 2 ⁰)	97.84	(2 ¹⁶ , 2 ⁰)	97.84
abalone9_18	(2 ¹² , 640)	96.58	(2 ⁻¹⁸ , 80)	93.68	(2 ⁵⁰ , 10)	95.73	(2 ⁴ , 2 ⁻⁴)	96.75	(2 ¹⁴ , 2 ⁻⁴)	95.72	(2 ¹⁶ , 2 ⁻⁴)	95.89
page-blocks1	(2 ¹⁰ , 440)	99.57	(2 ¹⁶ , 300)	98.94	(2 ¹⁸ , 140)	98.94	(2 ² , 2 ⁻²)	99.79	(2 ¹⁰ , 2 ⁻²)	99.79	(2 ¹⁰ , 2 ⁻²)	99.79
glass4	(2 ⁸ , 60)	96.74	(2 ³⁰ , 20)	95.81	(2 ²⁴ , 30)	96.74	(2 ⁻⁴ , 2 ⁻²)	96.74	(2 ¹⁰ , 2 ⁻²)	96.28	(2 ¹⁰ , 2 ⁻²)	96.28
ecoli4	(2 ³⁸ , 30)	99.11	(2 ²⁸ , 90)	97.32	(2 ² , 110)	98.21	(2 ⁻² , 2 ⁻⁴)	98.81	(2 ⁴ , 2 ⁻¹⁰)	97.92	(2 ⁻¹⁸ , 2 ⁻¹⁰)	97.92
yeast-1_vs_7	(2 ²⁶ , 100)	95.42	(2 ⁻¹⁶ , 90)	92.81	(2 ⁰ , 780)	94.34	(2 ¹² , 2 ⁰)	94.99	(2 ¹⁴ , 2 ⁻⁸)	92.16	(2 ² , 2 ⁰)	93.90
shuttle-C0_vs_C4	(2 ¹⁴ , 10)	100	(2 ³⁸ , 10)	100	(2 ²⁸ , 10)	100	(2 ¹⁴ , 2 ⁰)	100	(2 ²⁶ , 2 ⁰)	100	(2 ²⁶ , 2 ⁰)	100
glass2	(2 ⁻¹⁸ , 10)	92.06	(2 ¹⁸ , 630)	87.81	(2 ⁻¹⁸ , 10)	92.06	(2 ⁻¹⁸ , 2 ⁻¹⁸)	92.06	(2 ⁻¹⁸ , 2 ⁻¹⁸)	92.06	(2 ⁻¹⁸ , 2 ⁻¹⁸)	92.06
glass-0-1-6_vs_2	(2 ⁻¹⁸ , 10)	91.16	(2 ¹⁴ , 560)	87.49	(2 ⁻¹⁸ , 10)	91.16	(2 ⁻¹⁸ , 2 ⁻¹⁸)	91.16	(2 ⁻¹⁸ , 2 ⁻¹⁸)	91.16	(2 ⁻¹⁸ , 2 ⁻¹⁸)	91.16
vowel0	(2 ³⁸ , 100)	100	(2 ⁴⁰ , 100)	100	(2 ⁴⁶ , 90)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100
yeast-0-5-6-7-9_vs_4	(2 ¹² , 130)	93.37	(2 ²² , 80)	87.30	(2 ¹² , 130)	92.05	(2 ¹⁸ , 2 ⁶)	93.37	(2 ¹⁶ , 2 ⁻⁸)	90.91	(2 ⁻⁴ , 2 ⁻²)	92.05
yeast-2_vs_4	(2 ⁴⁰ , 40)	96.31	(2 ³⁰ , 200)	96.11	(2 ⁰ , 880)	96.89	(2 ² , 2 ⁻²)	96.50	(2 ¹² , 2 ⁻⁸)	95.72	(2 ² , 2 ⁻²)	95.92
page-blocks0	(2 ³⁰ , 890)	96.86	(2 ³⁸ , 500)	95.45	(2 ³⁸ , 610)	96.05	(2 ¹⁶ , 2 ⁰)	96.97	(2 ¹⁶ , 2 ⁻⁸)	95.98	(2 ¹⁶ , 2 ⁻⁴)	96.16
ecoli3	(2 ²⁶ , 20)	93.15	(2 ³⁰ , 10)	89.28	(2 ²⁴ , 20)	91.37	(2 ⁻⁶ , 2 ⁻⁴)	93.74	(2 ¹⁰ , 2 ⁻⁴)	91.08	(2 ¹⁰ , 2 ⁻⁴)	91.68
yeast3	(2 ¹⁶ , 220)	94.61	(2 ²⁴ , 160)	92.99	(2 ² , 720)	94.68	(2 ⁰ , 2 ⁻²)	94.88	(2 ¹⁰ , 2 ⁻⁸)	94.14	(2 ⁻² , 2 ⁻⁴)	95.22
glass6	(2 ⁴² , 50)	97.67	(2 ⁵⁰ , 50)	97.67	(2 ²⁶ , 30)	97.67	(2 ⁸ , 2 ²)	97.67	(2 ¹⁶ , 2 ⁶)	97.67	(2 ¹² , 2 ⁴)	97.67
segment0	(2 ⁸ , 720)	99.78	(2 ¹⁶ , 820)	99.83	(2 ¹⁸ , 260)	99.78	(2 ⁴ , 2 ⁰)	99.74	(2 ⁴⁸ , 2 ²⁰)	99.87	(2 ⁴⁴ , 2 ²⁰)	99.91
ecoli2	(2 ¹⁸ , 40)	95.54	(2 ²⁴ , 180)	94.94	(2 ¹⁶ , 170)	95.84	(2 ⁻¹⁸ , 2 ⁻⁶)	97.03	(2 ⁸ , 2 ⁻⁶)	95.54	(2 ⁸ , 2 ⁻²)	95.83
new-thyroid1	(2 ¹⁴ , 170)	99.07	(2 ¹⁸ , 30)	99.04	(2 ¹⁶ , 80)	100	(2 ¹⁰ , 2 ⁻⁴)	99.07	(2 ¹⁴ , 2 ⁰)	99.54	(2 ⁶ , 2 ⁰)	99.54
new-thyroid2	(2 ¹⁸ , 40)	98.61	(2 ¹⁶ , 290)	99.54	(2 ¹² , 400)	100	(2 ² , 2 ⁻⁴)	99.07	(2 ¹² , 2 ⁻⁴)	99.54	(2 ¹⁴ , 2 ⁰)	99.54

Table 7.8: Performance result in terms of accuracy for binary problems with imbalance ratio $\in (0.2, 1)$.

Data Accuracy	Sigmoid Node						Gaussian Kernel					
	Unweighted ELM		Weighted ELM W1		Weighted ELM W2		Unweighted ELM		Weighted ELM W1		Weighted ELM W2	
	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C,L)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)	(C, γ)	Testing Result(%)
ecoli1	(2 ¹⁴ , 190)	92.57	(2 ³² , 580)	89.89	(2 ²⁶ , 20)	91.37	(2 ⁶ , 2 ⁻²)	92.57	(2 ¹⁴ , 2 ⁻²)	90.19	(2 ¹² , 2 ⁻²)	91.38
glass-0-1-2-3_vs_4-5-6	(2 ⁸ , 140)	94.86	(2 ¹⁰ , 160)	96.27	(2 ¹² , 70)	96.27	(2 ¹⁶ , 2 ⁰)	95.33	(2 ⁻¹⁸ , 2 ⁻⁴)	96.25	(2 ⁻¹⁸ , 2 ⁻⁴)	96.25
vehicle0	(2 ⁸ , 610)	99.53	(2 ¹⁶ , 540)	99.41	(2 ¹⁶ , 920)	99.53	(2 ¹² , 2 ⁴)	99.53	(2 ⁴² , 2 ¹²)	99.29	(2 ³² , 2 ⁸)	99.41
adult	(2 ⁻⁶ , 960)	84.66	(2 ⁻⁸ , 200)	82.10	(2 ⁶ , 460)	83.41	(2 ⁰ , 2 ⁵)	84.79	(2 ¹⁵ , 2 ⁰)	81.47	(2 ¹⁰ , 2 ⁵)	83.54
vehicle3	(2 ⁶ , 440)	86.41	(2 ¹⁶ , 730)	85.22	(2 ¹⁶ , 480)	85.22	(2 ¹⁰ , 2 ⁴)	86.05	(2 ²⁴ , 2 ⁶)	84.28	(2 ²⁴ , 2 ⁸)	85.11
vehicle1	(2 ⁶ , 850)	86.76	(2 ¹⁶ , 520)	85.58	(2 ¹⁴ , 660)	86.41	(2 ¹⁸ , 2 ⁸)	86.52	(2 ¹⁸ , 2 ⁴)	85.34	(2 ²⁶ , 2 ⁸)	85.81
vehicle2	(2 ⁸ , 710)	99.41	(2 ¹⁶ , 690)	99.53	(2 ²⁰ , 450)	99.53	(2 ¹⁴ , 2 ⁴)	99.41	(2 ²⁰ , 2 ²)	99.41	(2 ²² , 2 ⁴)	99.53
haberman	(2 ⁸ , 80)	75.49	(2 ⁻¹⁴ , 430)	75.16	(2 ⁴ , 180)	76.79	(2 ¹² , 2 ⁶)	75.49	(2 ¹² , 2 ⁶)	72.87	(2 ¹⁴ , 2 ¹⁰)	75.81
yeast1	(2 ¹⁴ , 160)	77.90	(2 ⁻¹⁰ , 340)	75.67	(2 ²² , 420)	77.90	(2 ⁴ , 2 ⁰)	77.90	(2 ¹² , 2 ⁻⁴)	73.38	(2 ¹⁴ , 2 ⁰)	77.49
Leukemia(Gene Sel)	(2 ²⁸ , 10)	100	(2 ⁻² , 10)	100	(2 ⁻¹⁴ , 10)	100	(2 ⁻¹⁸ , 2 ⁻²)	100	(2 ⁻¹⁸ , 2 ⁻²)	100	(2 ⁻¹⁸ , 2 ⁻²)	100
glass0	(2 ¹² , 960)	83.18	(2 ²⁴ , 990)	79.45	(2 ²⁰ , 980)	82.25	(2 ⁰ , 2 ⁻⁶)	85.98	(2 ⁴ , 2 ⁻¹⁰)	85.50	(2 ⁸ , 2 ⁻⁶)	85.52
iris0	(2 ⁻² , 10)	100	(2 ⁴ , 10)	100	(2 ² , 10)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100	(2 ⁻¹⁸ , 2 ⁻¹⁰)	100
glass1	(2 ¹⁴ , 700)	79.03	(2 ²² , 300)	78.99	(2 ²² , 240)	79.96	(2 ⁻¹⁸ , 2 ⁻⁶)	81.82	(2 ⁻¹⁸ , 2 ⁻¹²)	81.77	(2 ⁻¹⁸ , 2 ⁻⁶)	81.82
pima	(2 ⁰ , 480)	78.25	(2 ³⁸ , 20)	76.56	(2 ⁴² , 30)	78.25	(2 ¹⁰ , 2 ⁸)	77.99	(2 ⁸ , 2 ²)	76.04	(2 ¹⁸ , 2 ⁶)	77.60
wisconsin	(2 ¹⁰ , 50)	97.66	(2 ³² , 60)	97.66	(2 ⁴² , 30)	93.37	(2 ⁻² , 2 ²)	97.22	(2 ⁸ , 2 ⁴)	97.51	(2 ⁶ , 2 ²)	97.22
colon	(2 ²⁸ , 210)	93.75	(2 ²⁰ , 120)	90.63	(2 ³⁴ , 280)	90.63	(2 ⁴ , 2 ¹⁶)	87.50	(2 ⁸ , 2 ¹⁴)	81.25	(2 ⁸ , 2 ¹⁶)	87.50
Colon(Gene Sel)	(2 ⁻¹⁴ , 440)	93.75	(2 ⁻⁴ , 10)	87.50	(2 ⁻⁸ , 330)	93.75	(2 ⁻¹⁸ , 2 ⁸)	87.50	(2 ⁶ , 2 ⁶)	84.38	(2 ⁻¹⁸ , 2 ⁸)	87.50
banana	(2 ¹⁶ , 500)	89.71	(2 ³² , 160)	89.80	(2 ³⁰ , 60)	89.84	(2 ² , 2 ⁰)	90.00	(2 ¹² , 2 ⁰)	89.80	(2 ⁶ , 2 ⁻²)	89.82

(a) yeast6 with IR 0.0243

(b) yeast1 with IR 0.4064

(c) banana with IR 0.8605

Figure 7.5: Display of testing result in terms of accuracy (white bar) and G-mean (blue bar), using unweighted ELM, weighted ELM W1 and weighted ELM W2, with Sigmoid node or Gaussian kernel.

(a) yeast6 with IR 0.0243

(b) yeast1 with IR 0.4064

(c) banana with IR 0.8605

Figure 7.6: ROC graph of three ELM classifiers using Sigmoid node on three datasets with different imbalance ratios.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis focused on study of extreme learning machines (ELMs) using optimization method. The research results are summarized as follows.

- **Optimization method based ε -ELM for regression**

Standard optimization method has been adopted in ELM for regression problems, since improvement was witnessed in ELM using optimization method for classification problems. On the other hand, it can be viewed as a combination of ε -SVR and ELM random nodes. The proposed ε -ELM was demonstrated with similar performance and training time as ε -SVR. However, ε -ELM tends to have insensitive user specified parameters due to the milder optimization constraints compared to ε -SVR. Precisely, there are only one parameter users need to tune in ε -ELM when another parameter can be simply fixed as a large value.

- **Face recognition using ELM**

ELM with optimization method has been utilized in face recognition system for the first time. Without loss of generality, the face images are holistically rep-

resented on the pixel level as well as represented by feature extraction on the component level. Afterwards, according to the literature of face recognition system, classic (say, PCA and LDA) as well as state-of-art dimensionality reduction methods (DLA) were incorporated to remove noise and redundant information. Since optimization based ELM for classification is designed for binary classification, two strategies one-against-all (OAA) and one-against-one (OAO) are used to decompose the multiclass problem into binary sub-problems. The proposed multiclass classifiers have been shown with better generalization performance than nearest neighbour classifier (NN) and comparable performance as the powerful support vector machine (SVM), both of which are widely used in the literature of face recognition. In particular, the proposed classifier has shown insensitive to parameter tuning, which has made the implementation more convenient. An extension of the work was inspired by the publication of the breakthrough in ELM theory, where an ELM framework was built up to include the traditional ELM with additive random nodes as well as ELM with kernel version. Therefore, kernelized ELM has been tested on the application of face recognition. Note that, an extraordinary advantage of the ELM framework than standard optimization method based SVMs and ELM is that, the usage of multiclass classification would be straightforward without any other modification, such as OAA or OAO as mentioned previously. The performance of kernelized ELM was shown much better than LS-SVM.

- **Learning to rank with extreme learning machine**

RankELM based on the unified ELM framework has been proposed as a relevance ranking algorithm for applications such as information retrieval. With regard to how the ranking problems are perceived, pointwise RankELM and pairwise RankELM were proposed, respectively. In addition, ELM linear random node was proposed for the first time, which complies with the universal approximation theorem of ELM to guarantee the approximation capability on any continuous unknown functions. Both pointwise and pairwise RankELM can be implemented

with ELM linear random nodes and linear kernel due to the unified ELM framework. The performance of proposed ranking algorithms were shown better than state-of-art linear ranking functions on most of the cases.

- **OS-eELM-left and OS-eELM-right**

An online sequential mode of enhanced ELM was proposed to deal with large scale or sequentially arriving data, where data can be learned on a one-by-one basis as well as a chunk-by-chunk basis. The existing OS-ELM is based on original ELM with left Pseudo-Inverse. The enhanced ELM is proved and validated better than original ELM in terms of generalization performance. Therefore, the proposed OS-eELM-left and OS-eELM-right outperform OS-ELM in theory and are validated on six benchmarking datasets including both regression and classification problems. The comparison is only targeted on OS-ELM is due to the excellence performance and advantage in computational efficiency of OS-ELM among the machine learning techniques.

- **Weighted ELM for imbalanced learning**

Based on the unified ELM framework, a weighted ELM has been proposed to deal with imbalanced data learning. Among the techniques in the literature of imbalanced learning, to assign different costs on the training samples is probably the simplest and yet most efficient method. originated from unweighted ELM, the proposed method is simple in theory and implementation; wide types of feature mapping or kernels are available as options; comparable or better generalization performance is achievable compared to the conventional machine learning techniques. On the other hand, by incorporating the information of imbalance class distribution, the proposed method is able to deal with data with imbalanced class distribution. Two weighting schemes are tested in the experiments on datasets with various imbalance ratios, which demonstrates the better performance of weighted ELM compared with unweighted ELM.

8.2 Future work

This thesis focuses on ELM with optimization methods and studies ELM from both application point of view and algorithm point of view, by closely following the development in ELM theory. New algorithms were proposed based on ELM framework to tackle various problems raised in the applications, such as sequentially arriving data, large scale data, and imbalanced data distribution. Moreover, ELM was studied on two important and popular applications: face recognition and relevance ranking for information retrieval.

Based on the research results, there is still space that the author thinks need to be improved. The potential future work of the interest is listed as below.

- As an efficient way to learn sequentially arriving data or large scale data, online sequential is essentially an important topic in the field of machine learning. Based on the ELM framework, the online sequential learning mode with non-kernel feature mappings was proposed in this thesis. However, kernel learning has been a hot spot and has shown good performance on various applications in the literature [62]. It is necessary that online sequential mode with kernelized ELM is developed.
- In the study of relevance ranking problems, it is found that the dataset is normally huge, say, up to 100,000. This is one of the reasons why linear ranking functions are popular instead of nonlinear ranking functions. However, in empirical study, nonlinear functions generally result in better performance than linear functions. It will be interesting to deal with large scale data when incorporating nonlinear feature mappings.

In addition, in the literature of relevance ranking, three categories of algorithms exist to apply machine learning techniques in ranking problems. Among them, pointwise and pairwise RankELM were proposed. However, the third category of algorithms, named listwise ranking methods, are empirically assumed with better approximation performance. Therefore, to derive listwise RankELM is of our great interest.

In general, relevance ranking is considered a learning process in the batch mode. For some time sensitive queries, say, in the commercial applications, to make use of information such as real-time click feedback helps obtain a fresh and accurate result. Few works [69, 26, 59] addressed this problem and provided preliminary but costly solutions. Since online sequential ELM is effective and convenient to implement, it is worth extending it to learn real-time click feedback in relevance ranking, and even some other real-time learning tasks.

In the era of big data, online sequential learning might not be an optimal solution given the requirement of fast response in relevance ranking. One possible way is to parallelize the computation through message passing interface (MPI) or MapReduce framework [19]. With possible rectification, ELM is able to realize both data parallelism and model parallelism, thus to greatly reduce the computational time on big data such as relevance ranking.

- The popularity of deep learning has brought back multi-layer neural network into researchers' attention since 2006 when Hinton [39] proposed the unsupervised layerwise pre-training as a good starting point for back-propagation (BP). Another effective way to pre-train the multi-layer perceptron (MLP) is auto-encoder [40, 90, 2, 78] which aims to reconstruct the input for every two layers. The idea of unsupervised pre-training actually provides a solution for the bad initial point of BP. And deep learning [9] has been successful in image/object classification and speech recognition, on which traditional MLP failed to perform well. It is also believed that except the low-level features deep architectures are able to extract high-level features, such as the object parts [10]. While shallow architectures can only extract low-level features like the edges. This can be considered as another reason why deep learning is so popular nowadays.

Contrary to the deep architectures, both ELM and SVM are considered shallow architectures since there is only one hidden layer. However, the pre-training process of every two adjacent layers is time consuming and involves tedious iterative tuning of many parameters, say, the learning rate, momentum, weight-decay factor and so on. It will be

interesting to see how ELM can simplify such complex learning procedure in the future.

Publication list

Journal Papers:

1. **Weiwei Zong** and Guang-Bin Huang, “Face Recognition Based on Extreme Learning Machine,” *Neurocomputing*, vol.74, no.16, pp.2541-2551, 2011.
2. **Weiwei Zong**, Guang-Bin Huang, and Yiqiang Chen, “Weighted Extreme Learning Machine for Imbalance Learning,” *Neurocomputing*, vol.101, no.0, pp.229-242, 2013.
3. **Weiwei Zong** and Guang-Bin Huang, “Learning to Rank with Extreme Learning Machine,” *Neural Processing Letters*, doi: 10.1007/s11063-013-9295-8, 2013.

Conference Papers:

1. **Weiwei Zong**, Hongming Zhou, Guang-Bin Huang, Zhiping Lin, “Face Recognition Based on Kernelized Extreme Learning Machine,” *International Conference on Autonomous and Intelligence Systems (AIS 2011)*, Burnaby, Canada, June 22-24, 2011.
2. **Weiwei Zong**, Yuan Lan, Guang-Bin Huang, “Sequential Learning based on Enhanced Extreme Learning Machine using Left or Right Pseudo-inverse,” *International Conference on Pattern Recognition Applications and Methods (ICPRAM 2012)*, Vilamoura, Algarve, Portugal, Feb 6-8, 2012.
3. Weimin Huang, Ning Li, Ziping Lin, Guang-Bin Huang, **Weiwei Zong**, Jiayin Zhou, Yuping Duan, “Liver Tumor Detection and Segmentation using Kernel-based Ex-

treme Learning Machine,” *Accepted by 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2013), Osaka, Japan, July 3-7, 2013.*

Bibliography

- [1] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *In Proceedings of the 15th European Conference on Machine Learning (ECML)*, pages 39–50, Pisa, Italy, September 20-24 2004.
- [2] G. Alain, Y. Bengio, and S. Rifai. Regularized auto-encoders estimate local statistics. *CoRR*, abs/1211.4246, 2012.
- [3] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Multiple-Valued Logic and Soft Computing*, 17(2-3):255–287, 2011.
- [4] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141, 2001.
- [5] V. S. Asirvadam, S. F. McLoone, and G. W. Irwin. Parallel and separable recursive levenberg-marquardt training algorithm. In *12th IEEE Workshop on Neural Networks for Signal Processing*, pages 129–138. IEEE, 2002.
- [6] R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
- [7] 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.

- [8] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.
- [9] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.
- [10] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- [11] C. Blake and C. Merz. UCI repository of machine learning databases. In <http://www.ics.uci.edu/~mllearn/MLRepository.html>. Department of Information and Computer Sciences, University of California, Irvine, USA, 1998.
- [12] C. L. Blake and C. J. Merz. *UCI Repository of machine learning databases*, 1998.
- [13] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, 1994.
- [14] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning, ICML '05*, pages 89–96, New York, NY, USA, August 7-11 2005. ACM.
- [15] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [16] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to Rank with Nonsmooth Cost Functions. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Proceedings of Advances in Neural Information Processing Systems*, pages 193–200, 2006.
- [17] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM*

- SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 186–193, 2006.
- [18] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, 2007.
- [19] C. T. Chu, S. K. Kim, Y. A. Lin, Y. Yu, G. R. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Proceedings of Advances in Neural Information Processing Systems*, pages 281–288. MIT Press, 2006.
- [20] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):681–685, 1998.
- [21] C. Cortes and V. N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
- [22] D. Cossock and T. Zhang. Subset ranking using regression. In *Proceedings of the 19th annual conference on Learning Theory*, COLT'06, pages 605–619, 2006.
- [23] D. Cossock and T. Zhang. Statistical analysis of Bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.
- [24] K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of Advances in Neural Information Processing Systems*, pages 641–647. MIT Press, 2001.
- [25] W. Deng, Q. Zheng, and L. Chen. Regularized extreme learning machine. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM 09)*, pages 389–395, March 30–April 2, 2009.
- [26] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *Proceedings of the*

- third ACM international conference on Web search and data mining, WSDM '10*, pages 11–20, 2010.
- [27] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, June 2006.
- [28] G. Feng, G.-B. Huang, Q. Lin, and R. Gay. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, 20(8):1352–1357, 2009.
- [29] R. Fletcher. *Practical Methods of Optimization: Volume 2 Constrained Optimization*. Wiley, New York, 1981.
- [30] A. Frank and A. Asuncion. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2010. <http://archive.ics.uci.edu/ml>.
- [31] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.
- [32] J.-Y. Gan and S.-B. He. Face recognition based on 2DLDA and support vector machine. In *Proceedings of International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR 2009)*, pages 211–214, Baoding, China, July 12–15, 2009.
- [33] D. B. Graham and N. M. Allinson. Characterizing virtual eigensignatures for general purpose face recognition. In H. Wechsler, J. P. Phillips, V. Bruce, F. F. Soulie, and T. S. Huang, editors, *Face Recognition: From Theory to Applications, NATO ASI Series F, Computer and System Sciences*, volume 163, pages 446–456. Springer, Berlin, 1998.
- [34] G. Guo, S. Z. Li, and K. Chan. Face recognition by support vector machines. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 196–201, Grenoble, France, March 26–30, 2000.

-
- [35] S. Haykin. *Neural Networks: A comprehensive foundation*. New Jersey: Prentice Hall, 1999.
- [36] H. He and E. A. Garcia. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, September 2009.
- [37] B. Heisele, P. Ho, J. Wu, and T. Poggio. Face recognition: component-based versus global approaches. *Computer Vision and Image Understanding*, 91(1-2):6–21, 2003.
- [38] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.
- [39] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.
- [40] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [41] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.
- [42] X. Hong, S. Chen, and C. J. Harris. A kernel-based two-class classifier for imbalanced data sets. *IEEE Transactions on Neural Networks*, 18(1):28–41, 2007.
- [43] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.
- [44] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 3 2002.
- [45] G.-B. Huang. Learning capability and storage capacity of two-hidden-layer feed-forward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281, 2003.

- [46] G.-B. Huang and H. A. Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE Transactions on Neural Networks*, 9(1):224–229, 1998.
- [47] G.-B. Huang and L. Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70:3056–3062, 2007.
- [48] G.-B. Huang and L. Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71(16-18):3460–3468, 2008.
- [49] G.-B. Huang and L. Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71:3460–3468, 2008.
- [50] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006.
- [51] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006.
- [52] G.-B. Huang, X. Ding, and H. Zhou. Optimization method based extreme learning machine for classification. *Neurocomputing*, 74:155–163, 2010.
- [53] 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: Cybernetics*, 34, 2004.
- [54] 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.
- [55] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. Extreme learning machine for regression and multi-class classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(2):513–529, 2012.

- [56] G.-B. Huang, Q.-Y. Zhu, K. Z. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan. Can threshold networks be trained directly? *IEEE Transactions on Circuits and Systems II*, 53(3):187–191, 2006.
- [57] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1-3):489 – 501, 2006.
- [58] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: A new learning scheme of feedforward neural networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN 2004)*, volume 2, pages 985–990, Budapest, Hungary, July 25-29, 2004.
- [59] Y. Inagaki, N. Sadagopan, G. Dupret, A. Dong, C. Liao, Y. Chang, and Z. Zheng. Session based click features for recency ranking. In M. Fox and D. Poole, editors, *AAAI*. AAAI Press, 2010.
- [60] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 41–48, New York, NY, USA, July 24-28 2000. ACM.
- [61] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.
- [62] J. Kivinen, A. J. Smola, and R. C. Williamson. Online learning with kernels. *IEEE Transaction on Signal Processing*, 52(8):2165–2176, 2004.
- [63] Y. Lan, Y. C. Soh, and G.-B. Huang. Extreme learning machine based bacterial protein subcellular localization prediction. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1859–1863, Hong Kong, China, June 1-6 2008.

- [64] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867, 1993.
- [65] P. Li, C. J. C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *NIPS*. Curran Associates, Inc., 2007.
- [66] Z. Li and X. Tang. Using support vector machines to enhance the performance of bayesian face recognition. *IEEE Transactions on Information Forensics and Security*, 2(2):174–180, 2007.
- [67] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 17(6):1411–1423, 2006.
- [68] S.-H. Lin, S.-Y. Kung, and L.-J. Lin. Face recognition/detection by probabilistic decision-based neural network. *IEEE Transactions on Neural Networks*, 8(1):114–132, 1997.
- [69] T. Moon, L. Li, W. Chu, C. Liao, Z. Zheng, and Y. Chang. Online learning for recency search ranking using real-time user feedback. In *Proceedings of the 19th ACM international conference on Information and knowledge management, CIKM '10*, pages 1501–1504, 2010.
- [70] R. Nallapati. Discriminative models for information retrieval. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71. ACM, July 25-29 2004.
- [71] L. S. Ngia, J. Sjöberg, and M. Viberg. Adaptive neural nets filter using a recursive levenberg-marquardt search direction. In *the 32nd Asilomar Conference on Signals, Systems and Computers*, CA, USA, November 1998.
- [72] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and T. Salakoski. Learning to rank with pairwise regularized least-squares. In T. Joachims, H. Li, T.-Y. Liu,

- and C. Zhai, editors, *Proceedings of the SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 27–33, 2007.
- [73] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.
- [74] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.
- [75] P. J. Phillips. Support vector machines applied to face recognition. In *Proceedings of Advances in neural information processing systems II*, pages 803–809, Denver, Colorado, USA, 1999.
- [76] T. Qin, T.-Y. Liu, J. Xu, and H. Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval Journal*, 2010.
- [77] C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and its Applications*. John Wiley & Sons, Inc, New York, 1971.
- [78] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on Machine learning*, ICML '11, pages 833–840, 2011.
- [79] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Vol: Foundations*, pages 318–362. MIT Press, Cambridge, MA, 1986.
- [80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagation errors. *Nature*, 323:533–536, 1986.
- [81] D. Serre. *Matrices: Theory and Applications*. Springer-Verlag New York, Inc, 2002.

- [82] A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In *Proceedings of Advances in Neural Information Processing Systems*, 2002.
- [83] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
- [84] D. Tao, X. Li, X. Wu, and S. J. Maybank. Geometric mean for subspace selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):260–274, 2009.
- [85] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08*, pages 77–86, 2008.
- [86] K.-A. Toh. Deterministic neural classification. *Neural Computation*, 20(6):1565–1595, 2008.
- [87] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '07*, pages 383–390, New York, NY, USA, 2007. ACM.
- [88] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
- [89] V. N. Vapnik. *Statistical learning theory*. Wiley, New York, 1998.
- [90] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.
- [91] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

- [92] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1192–1199, 2008.
- [93] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 391–398, 2007.
- [94] J. Yuan, J. Li, and B. Zhang. Learning concepts from large scale imbalanced data sets using support cluster machines. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 441–450, Santa Barbara, California, USA, October 23-27 2006.
- [95] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 271–278, 2007.
- [96] G.-Y. Zhang, S.-Y. Peng, and H.-M. Li. Combination of dual-tree complex wavelet and svm for face recognition. In *Proceedings of International Conference on Machine Learning and Cybernetics*, volume 5, pages 2815–2819, Kunming, China, July 12-15, 2008.
- [97] T. Zhang, D. Tao, and J. Yang. Discriminative locality alignment. In *Proceedings of the Tenth European Conference on Computer Vision (ECCV 08)*, pages 725–738, Marseille, France, October 12-18, 2008.
- [98] L. Zhao, Y. Song, Y. Zhu, C. Zhang, and Y. Zheng. Face recognition based on multi-class svm. In *Proceedings of Chinese Control and Decision Conference (CCDC 09)*, pages 5871 –5873, Guilin, China, June 17-19, 2009.
- [99] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, December 2003.

- [100] W. Zhao, A. Krishnaswamy, and R. Chellappa. Discriminant analysis of principal components for face recognition. In *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 336–341, Nara, Japan, April 14-16, 1998.
- [101] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search neur. In *Proceedings of Advances in Neural Information Processing Systems*, 2007.
- [102] W. Zong and G.-B. Huang. Face recognition based on extreme learning machine. *Neurocomputing*, May 2011.
- [103] W. Zong, H. Zhou, G.-B. Huang, and Z. Lin. Face recognition based on kernelized extreme learning machine. In *Proceedings of the Second international conference on Autonomous and intelligent systems*, AIS’11, pages 263–272. Springer-Verlag, 2011.