Extreme learning machine with sparse connections

Bai, Zuo

2015

Bai, Z. (2015). Extreme learning machine with sparse connections. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/65656

https://doi.org/10.32657/10356/65656

EXTREME LEARNING MACHINE WITH SPARSE CONNECTIONS BAI ZUO 2015

NANYANG TECHNOLOGICAL UNIVERSITY

EXTREME LEARNING MACHINE WITH SPARSE CONNECTIONS

BAI ZUO

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

2015

EXTREME LEARNING MACHINE WITH SPARSE CONNECTIONS

BAI ZUO

SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING

A thesis submitted to the Nanyang Technological University

in fulfillment of the requirements for the degree of

Doctor of Philosophy

2015

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	BAI ZUO

Acknowledgments

I would like to express my sincere gratitude to all the people who are for me, with me and by me.

First of all, I would like to thank my supervisor Associate Prof. Huang Guangbin, for his excellent guidance and support during this whole PhD study. Not only he gave me the right direction of research, but also offered opportunities to improve my comprehensive competitiveness, ranging from proposal writing, conference arrangement, presentation skills, etc.

Secondly, I would like to express my sincere gratitude to my co-supervisor, Prof. Wang Danwei, for providing this great opportunity of pursuing my PhD degree at NTU and offering great suggestions in research.

Thirdly, I would like to thank all my friends and colleagues from Intelligent Robotics Lab, Internet of Things Lab and BMW-HMI Lab for the great discussions, joys and fun. To my roommates, from Nanyang Valley and 903 apartment: those enormous fun and joys, midnight talks and supports are my lifelong treasure.

Finally, my biggest thanks and love belong to my parents and my lovely girlfriend, Geng Hefang, for their love, care and support. I would like to explore the coming unknown journey together with you, make our life content, enjoyable, fruitful and meaningful.

Table of Contents

Summary

Motivated by the human brain, neural network has been widely used and extensively investigated in the past several decades. It works and operates in a highly-complex, nonlinear and parallel way. Single-hidden layer feedforward neural network (SLFN) is the most common type because of: 1) its simple structure; 2) the approximation capability of very complicated nonlinear function. The conventional training methods for SLFN, such as back-propagation (BP) algorithm [88], involve numerous gradient descent steps and suffer from trivial issues including slow convergence rate, local minima, tedious human involvement, etc.

In recent years, a novel algorithm named Extreme Learning Machine (ELM) [50, 51] has been attracting plenty of research attention. ELM was initially proposed as a variant and improvement on the classic SLFN and was later extended to the "generalized" SLFN, which was not necessarily neuron alike. Different from traditional methods, which demand tedious parameter tunings, ELM randomly generates the input weights and analytically calculates the output weights, thus providing a simple and deterministic solution.

However, in almost all ELM implementations realized in the past, hidden nodes are fully connected with the input ones. On one hand, it needs to store all the connection weights in memory and perform addition/multiplication operations for all the weights. Therefore, it requires plenty of storage space and testing time, especially when facing large-scale applications. On the other hand, the fully connected ELM does not explicitly model the local structures, and thus may not perform well for locally correlated applications, such as image processing, speech recognition, etc.

In Chapter 3 and Chapter 4, we propose the sparse ELM as an alternative solution to deal with classification and regression problems respectively. In terms of generalization performance, the sparse ELM is on par with the unified ELM and better than traditional support vector machine (SVM) [15]. Additionally, it provides a sparse solution, reducing the storage space and testing time significantly. Furthermore, we develop efficient training algorithms based on iterative update for classification and regression problems separately. They both break down the large quadratic programming (QP) problem into a series of sub-problems, each of which can be solved analytically. Different from the popular unified ELM [49], whose computational complexity is between quadratic and cubic with regard to the training size N, the proposed algorithms scale only quadratically with respect to N. The decrease of the complexity gives the proposed sparse ELM considerable advantages to handle large-scale applications.

In Chapter 5, we investigate the local receptive fields based extreme learning machine (ELM-LRF). It thoroughly discusses the issue: *can local receptive fields be implemented in ELM?* ELM theories prove that hidden nodes can be randomly generated according to any continuous probability distribution. Consequently, local receptive fields can be naturally extended and implemented in ELM [2]. Inspired by convolutional neural networks (CNNs) [55], we construct the network of ELM-LRF by randomly generating hidden nodes that are in sparse connections with the input ones and perform convolution operations. Experimental results on the NORB dataset, a benchmark for object recognition, show that the proposed ELM-LRF achieves the *best accuracy* and accelerates the learning speed up to *200 times*.

Subsequently, in Chapter 6 we suggest to use the proposed ELM-LRF as a general framework for generic object recognition. ELM-LRF is operated directly on the raw images without any pre-processing, thus suitable for different datasets. In addition, the simple structure only requires few computations and minimal human involvement as most connection weights are generated randomly. The general framework of ELM-LRF is evaluated on several generic object recognition datasets, NORB, ETH-80, COIL and ALOI. And it achieves the best accuracy on NORB, COIL and ALOI while comparable with state-of-the-art result on ETH-80.

In summary, this thesis studies ELM with sparse connections. We first present the sparse ELM as an alternative solution for the unified ELM, which significantly reduces the storage space and testing time. Additionally, the sparse ELM has better scalability, making it preferable to handle large-scale applications. Later, we conduct some research on the ELM-LRF and shows that it is especially suitable and efficient for highly correlated applications, such as image processing, speech recognition, natural language processing, etc.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background

Machine learning is a scientific discipline that learns from the data without explicitly modeling the underlying relationship of the data. It is highly relevant to other studies, including statistics, computer science, data mining and artificial intelligence [103]. Currently, one challenging issues for machine learning include how to generate proper representations for the original input and how to handle big data problems.

Neural network is a popular method in machine learning and has been thoroughly investigated in recent decades because of its approximation capability of complicated nonlinear functions [34]. It was inspired by human brain and operated in a highly complex, nonlinear and parallel manner. Single-hidden layer feedforward neural network (SLFN) is the most common one because of the simple structure and the superb approximation capability.

The conventional training methods for SLFN, such as back-propagation (BP) algorithm [88], are mostly gradient descent based and require numerous updates of the parameters. Consequently, these training methods are slow, while suffering from troubles like slow convergence rate, local minima, intensive human involvement, etc. [51, 50].

The emerging extreme learning machine (ELM) was originally proposed as an improvement for the classic SLFN, where the input weights (connection weights from input to hidden layer) are generated randomly based on any continuous probability distribution [46, 44]. Subsequently, ELM was further developed to the "generalized" SLFN and provides a unified framework for different learning methods, including SLFN, support vector machine (SVM), radial basis function (RBF) network, etc. [49, 41]. The unified ELM provides an efficient and deterministic solution for different applications, including classification, regression, clustering, as it randomly generates the input weights and analytically calculates the output weights.

Additionally, image processing, speech recognition and similar tasks are important yet difficult applications for machine learning techniques. These tasks involve plenty of local correlations and thus cannot be directly handled by any machine learning techniques. Traditionally, pre-processing steps first transform the raw inputs (like images, speeches) into well-designed features, such as shape models, SIFT features, etc. Then, machine learning techniques are followed to deal with these features. Recently, vast works are proposed to deal with the raw inputs directly due to the great success in deep learning [37] and convolutional neural networks (CNNs) [64, 98].

1.2 Motivations and Objectives

ELM is widely used because of its great generalization performance and exceptionally fast speed. It presents high accuracy with superb efficiency for different applications, including but not limited to biomedical analysis [108], system modeling [107], action recognition [75], etc. [41].

However, input and hidden nodes are in full connections for almost all ELM implementations studied before. Thus, it needs huge storage space and much testing time, since all the connection weights have to be stored in the memory and performed addition/multiplication operations in the testing phase. In addition, the fully connected ELM does not model local structures explicitly. Therefore, it usually performs poorly when facing applications with strong local correlations, such as image analysis, speech recognition, natural language processing [2, 70].

In this thesis, we aim to study ELM with sparse connections and solve the aforesaid problems associated with fully connected ELM.

In the beginning, we aim to construct a sparse ELM. There are three major approaches to train a SLFN [34]: 1) gradient descent based method; 2) least squares based method; 3) standard optimization based method. The conventional training methods are based on gradient descent and the unified ELM is based on least squares. In this thesis, we build up the ELM network based on standard optimization and provide a sparse solution. Comparing to the unified ELM, it significantly reduces the storage space and testing time, while achieving comparable accuracy. Furthermore, we develop the training algorithms for classification and regression separately. The complexity scales only quadratically with regard to the training size N, providing better scalability than the unified ELM.

Later, we discuss the open question: *can local receptive fields be implemented in ELM?* We show that local receptive fields are naturally valid in ELM and can be easily implemented. Hence, we propose the local receptive fields based extreme learning machine (ELM-LRF). The input and hidden nodes are in sparse connections and bounded by corresponding local receptive fields. In this way, the local structures are explicitly modeled, enabling ELM-LRF to be suitable for image processing and similar works.

Finally, we try to solve some real-world applications and suggest to use ELM-LRF as a general framework for generic object recognition. We show that ELM-LRF provides a deterministic and efficient solution for generic object recognition, setting new records for NORB, COIL [78], ALOI [31] and on par with the best result of ETH-80 [66].

1.3 Major Contributions

In this thesis, we thoroughly study ELM with sparse connections and well solve the issues associated with fully connected ELM. In this section, we summarize all the major contributions during the PhD study.

Firstly, we propose a sparse ELM for classification from the perspective of standard optimization. It presents comparable accuracy with the unified ELM, while providing a compact network and largely reducing the storage space and testing time. Moreover, we develop a training algorithm for the sparse ELM specifically. The large quadratic programming (QP) problem is broken into a series of sub-problems, each of which includes only one Lagrange multiplier. These sub-problems can be solved analytically in a sequential manner. The computational complexity is quadratic with respect to the training size N. Thus, the sparse ELM has better scalability than the unified ELM, making it superior when facing large-scale applications.

Secondly, we extend the preceding classification-applicable-only network into regression problems and propose the sparse ELM for regression. It shares resemblances with the aforementioned work, yet dealing with different constraints, loss functions and problems. It also largely reduces the storage space, testing time and provides better scalability.

Thirdly, we study another form of sparse connections: local receptive fields. Strong local correlations exist in some applications, such as image processing, speech recognition and similar tasks. Thus, it is reasonable to expect the network to learn these local correlations by sparse and local connections instead of full ones. We discover that local receptive fields can be easily realized by ELM and construct the ELM-LRF network. The input weights to the hidden nodes are generated randomly and the output weights are calculated analytically. Experimental results well demonstrate the superb performance and extraordinarily fast speed of ELM-LRF.

Finally, we set up a general framework for generic object recognition with ELM-LRF because of its distinct merits: 1) task non-specific as ELM-LRF uses no task-specific information; 2) simple to use because the users only need to choose several parameters through validation; 3) highly efficient since most connection weights are generated randomly. The general framework presents state-of-the-art accuracy with exceptionally high speed on several benchmark datasets, NORB, ETH-80, COIL and ALOI.

1.4 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 presents the literature reviews on current popular machine learning techniques, including support vector machine (SVM), extreme learning machine (ELM), convolutional neural networks (CNNs), etc.

Chapter 3 illustrates the sparse ELM for classification in details, showing how it provides a sparse solution and decreases the computational complexity.

Chapter 4 extends the sparse ELM for regression, since the method illustrated in Chapter 3 cannot solve regression problems. It produces a sparse solution and requires lower complexity.

Chapter 5 discusses the properties of local receptive fields for ELM and proposes ELM-LRF accordingly.

Chapter 6 constructs a general framework for generic object recognition with ELM-LRF and solves real-world applications.

Chapter 7 concludes the thesis and addresses the future work.

Chapter 2

Literature Review

2.1 Machine Learning

People encounter decision making in everyday life, such as whether going out or not when the weather is cloudy. Thus, it is attractive to enable the machine to make these kinds of decisions. For some problems, we can use human knowledge to model the latent factors and generate explicit mathematical expressions for the problem. This is called designing with domain knowledge. However for other problems, we can only collect some data (several measurements and corresponding labels) and learn the underlying relationship implicitly. This is *machine learning* with data-driven knowledge.

Machine learning learns from the data by optimizing a performance criterion without explicitly modeling the underlying relationship within the data. Key factors for successful machine learning include representation and generalization. All machine learning techniques consist of data representation and generalization. Data representation determines in what format the data are presented to the learning technique and the generalization decides how well the system performs on unseen data [103].

Machine learning is closely related to other fields, such as statistics, computer science, artificial intelligence, data mining, etc. Compared with other areas, machine learning concentrates on how to build up the model, to perform the optimization and to regularize in order to make the optimal use of the data.

Before we go any further on machine learning, let us describe the basic notations and assumptions first. There are two types of datasets:

(1) *Labeled dataset:* $\mathbf{X}_{N\times d} = \left[\mathbf{x}_1^T \ \cdots \mathbf{x}_N^T\right]^T, \mathbf{x}_i \in \mathbf{R}^{1\times d}$ and $\mathbf{T}_{N\times m} = \left[\mathbf{t}_1^T \ \cdots \mathbf{t}_N^T\right]^T, \mathbf{t}_i \in \mathbf{R}^{1\times m}$

(2) *Unlabeled dataset:* $\mathbf{X}_{N\times d} = \left[\mathbf{x}_1^T \ \cdots \mathbf{x}_N^T\right]^T, \mathbf{x}_i \in \mathbf{R}^{1\times d}$

where $\mathbf{X}_{N\times d}$ is the feature set of N samples. Each sample $\mathbf{x}_i \in \mathbf{R}^{1\times d}$ is a feature vector in row format. $\mathbf{T}$ denotes the target set and $\mathbf{t}_i \in \mathbf{R}^{1\times m}$ indicates the target of the feature vector. Additionally, $\mathbf{x}_i$ and $\mathbf{t}_i$ is called a data pair. In some cases, there are no targets observed, resulting in the unlabeled dataset.

In machine learning, the underlying distribution is the probability of all possible data pairs to occur in the real world [12]. However, we can only observe a subset of all these possible data pairs, denoted as *training data*. It is assumed that the training data are independently and identically (*i.i.d*) sampled from the underlying distribution. Thus, we can model the distribution from the training data and predict on the unseen *test data* using the model.

In general, there are several groups of machine learning techniques depending on the problem and the dataset provided:

(1) *Supervised learning:* Given a labeled dataset as the training data, supervised learning aims to discover the relationship between the feature set $\mathbf{X}$ and the target set $\mathbf{T}$. If the target is $\mathbf{t}_i = [t_{i,1} \ \cdots \ t_{i,m}] \in \mathbf{R}^{1\times m}$, it is a classification problem. If the target $\mathbf{t}_i \in \mathbf{R}$ is a real value, it is a regression problem.

(2) *Unsupervised learning:* The training data provided is unlabeled. Unsupervised

Table 2.1: List of common supervised learning methods

Category	Learning methods
Linear model	Neural networks Support vector machine (SVM) Support vector regression (SVR) Linear regression Logistic regression
Non-parametric model	K-nearest neighbors Kernel density estimation Kernel regression
Non-metric model	Classification and regression tree (CART), decision tree
Parametric model	Naive Bayes Gaussian discriminant analysis (GDA) Hidden Markov models (HMM) Probabilistic graphical models
Mixed methods	Bagging (bootstrap + aggregation) Adaboost Random forests

learning focuses on clustering [105], probability density estimation, dimensionality reduction, etc.

(3) *Semi-supervised learning:* Semi-supervised learning handles both labeled and unlabeled datasets and combines them together to solve the problem.

(4) *Reinforcement learning:* Reinforcement learning closely interacts with the environment and provides a sequence of decision makings, with the purpose of maximizing the long-term reward. Unlike the supervised learning, there are no targets, or correct actions, for any feature vectors [56].

There are many different types of machine learning techniques. For instance, linear discriminant analysis is originated from statistics, while rule-based classifiers or decision trees come from the field of data mining. Table 2.1 lists some common supervised learning methods [12].

2.2 Neural Networks

It has been discovered that human brain, as an information processing system, consists of numerous fundamental processing units, the biological neurons. Each neuron provides very basic processing capability. And the brain integrates all these neurons and performs various kinds of cognitive functions, such as logical reasoning, computing, memorizing, learning from experience and decision making.

Neural network is thus inspired by the human brain and becomes a popular machine learning technique. Unlike conventional digital computer, it is in nature highly complex, nonlinear and parallel. Neural network could approximate severely complicated functions and solve a wide variety of problems. Neurons (also called nodes), the basic components of neural networks, are combined together to perform certain machine learning tasks, such as pattern recognition, time series prediction, etc. In the past several decades, large amounts of works have been proposed about neural networks [34, 53].

2.2.1 The basic unit: node

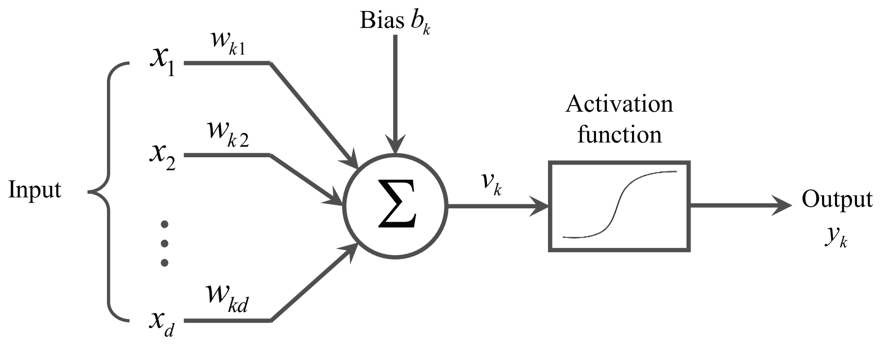


Figure 2.1: The basic unit of neural networks: node *k*

The node is the basic unit of neural networks. In order to mimic the human brain, the mathematical formulation of the node includes three fundamental elements:

(1) *Synapses (connection weights)*: The input signal x_j is connected to node k through

the connection weight w_{kj}. Different from the human brain, the weight w_{kj} in neural networks may have negative as well as positive values [34].

(2) *Summation operator*: This operator is a linear combiner that sums up the weighted input signals. Note that an additional bias term b_k may be included.

$$v_k = \sum_{j=1}^{d} w_{kj} x_j + b_k = \mathbf{x}\mathbf{w}_k + b_k, \quad \mathbf{x} \in \mathbf{R}^{1\times d}, \mathbf{w}_k \in \mathbf{R}^{d\times 1} \tag{2.1}$$

(3) *Activation function*: The purpose of activation function G is to limit the value of the output. At here, we show three typical activation functions:

i) *Threshold function*:

$$G(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \tag{2.2}$$

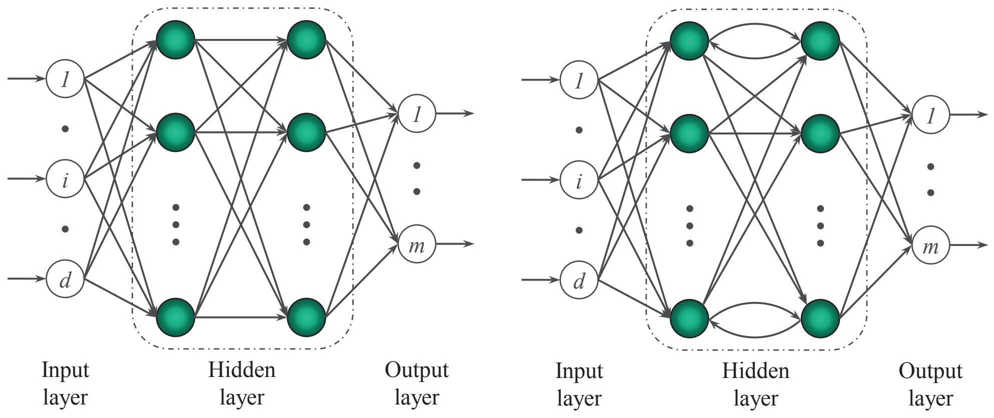
ii) *Piecewise-linear function*:

$$G(v) = \begin{cases} 1 & v \geq \frac{1}{2} \\ v & -\frac{1}{2} < v < \frac{1}{2} \\ 0 & v \leq -\frac{1}{2} \end{cases} \tag{2.3}$$

iii) *Sigmoid function*:

$$G(v) = \frac{1}{1 + \exp(-a \cdot v)} \tag{2.4}$$

More functions can be used in neural networks, such as *signum*, *hyperbolic tangent*, etc.



(a) Feedforward neural network (b) Recurrent neural network

Figure 2.2: Network architectures

2.2.2 Network architectures

In order to perform certain tasks, the nodes need to be linked and structured into some architectures with proper learning algorithm for training. Based on the network architectures, we can identify two general types:

(1) *Feedforward neural networks (FNNs):* The network is layered and organized in a strictly feedforward manner. The signals are transmitted in forward direction only. As depicted in Fig. 2.2(a), every node is connected to subsequent nodes in forward sense and the network is in full connections.

(2) *Recurrent neural networks (RNNs):* As shown in Fig. 2.2(b), the network consists of at least one feedback loop. RNN is in effect dynamic because the output will in turn change the input, until reaching an equilibrium state. Thus, one input would generate a series of outputs. Popular RNN networks include Hopfield network [38] and Kohonen self-organizing maps [60].

2.2.3 Single-hidden layer feedforward neural networks (SLFNs)

Before the great advancement of deep learning [36] on 2006, researchers had been attempting to train multilayer neural networks for decades, yet achieving no significant success. Therefore, SLFN was the most frequently used one amongst the family of FNNs. As implied by the name, only one hidden layer exists in SLFN. It provides great approximation capability with exceptionally simple structure. Fig. 2.3 shows the architecture of SLFN.

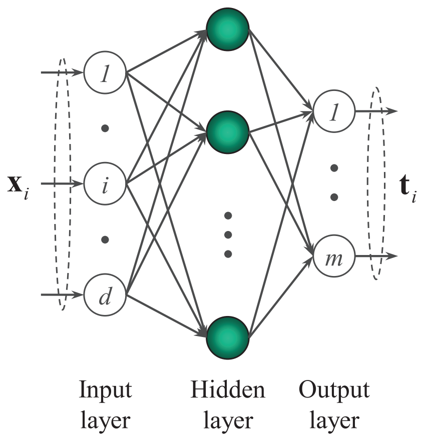


Figure 2.3: Single-hidden layer feedforward neural network (SLFN)

SLFN belongs to supervised learning, where the feature and target sets are both provided in the training data. Traditionally, all connection weights will be adjusted in order to learn from the training data and predict on the test (unseen) data. Back-propagation (BP) algorithm is one famous training method based on gradient descent [88]. It calculates the gradients of a loss function with respect to the weights and feeds them backward in order to minimize the loss function. BP algorithm delivers an elegant way to train the SLFN. However, it also faces problems, such as local minima, intensive human involvement and time consuming of parameter tunings. Extreme learning machine (ELM) [43, 51] was thus proposed, which generates the hidden nodes randomly and solves the issues associated with BP. We will introduce ELM in detail later.

2.3 Convolutional Neural Networks (CNNs)

In this section, we take a closer review at the convolutional neural network (CNN). It is a variant of multilayer feedforward neural networks (or called multilayer perceptrons (MLPs)) and an exception that presents nice performance with multilayer structure from decades ago. CNN was initially inspired by the visual cortex, particularly by the model proposed in [52]. The first computational model with local connections was introduced by Fukushima *et al.* [29].

CNN has been successfully adopted in many different applications, including but not limited to object recognition, image processing, speech recognition. Unlike CNN, traditional methods for these applications first transform the input into hand-crafted features and then perform classification on these features [70]. Consequently, the performance largely depends on the quality of the features, which are manually designed. In contrast, CNN is operated directly on the raw inputs, eliminating the design of features. Thus, it is less task-specific than traditional methods as it implicitly learns the features from the raw inputs rather than designing for any specific application.

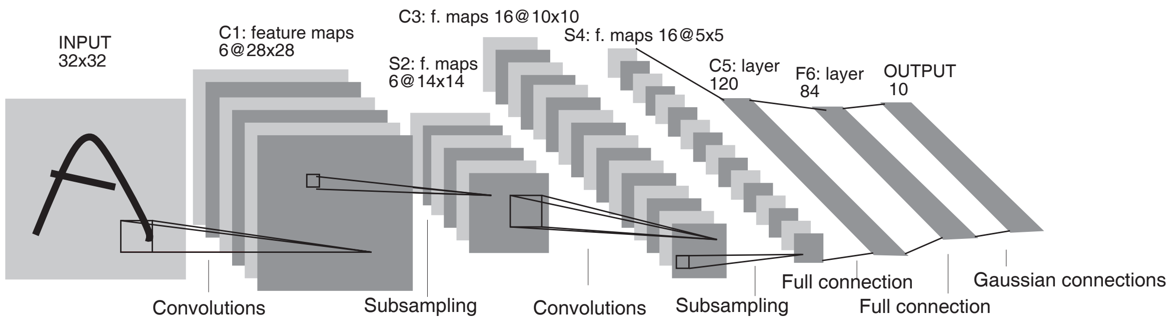


Figure 2.4: A classic convolutional neural network: LeNet-5

Fig. 2.4 displays a classic CNN, LeNet-5, which was proposed by LeCun *et al.* [64]. The input is the image to be classified. There are two main operations in the CNN: convolution and subsampling (also called pooling). Convolutional and subsampling layers are stacked together alternately until generating the high-level representations (layer F6 in Fig. 2.4). The representations (or features) will be fed into the subsequent classifier, such as the last two layers of Gaussian connections in Fig. 2.4.

(1) *Convolution*: Use the 1-st convolutional layer C1 in LeNet-5 as an example. Each unit in C1 has a receptive field in the lower layer: a 5×5 patch in the input layer. And the 6 feature maps in C1 are calculated with 6 different filters. Units in the same feature map share the same filter. This setting enables the network to learn different representations while keeping the number of parameters tractable. Each filter includes the connection weight $\mathbf{w} \in \mathbf{R}^{5\times 5}$ and a bias b. The value of a unit in C1 (y_1) is calculated as follows:

$$y_1 = G(\sum_{i=1}^{5}\sum_{j=1}^{5} w_{ij} \cdot x_{ij} + b) \tag{2.5}$$

where G is a nonlinear function.

(2) *Subsampling*: The feature detected at one location tends to be useful at different locations. Subsampling is a straightforward approach to introduce translational invariance. Different subsampling methods could be performed over local areas, such as averaging and max-pooling [14]. Additionally, it would alleviate the computational burden as it reduces the size of the maps.

Stimulated by the recent developments in deep learning [36, 6, 5], some variants of CNN, Deep CNN [62], GoogLeNet [98], show superior performance on super large image datasets, such as ImageNet [89], PASCAL [22]. The common approach to train a CNN is the BP algorithm. With so many parameters to be tuned, it requires a large training set and computational capability in order to train the network properly. Furthermore, it also faces the challenging issues associated with BP, including slow convergence rate, intensive human intervention, local minima, etc.

2.4 Support Vector Machine (SVM)

Support vector machine (SVM) and its variants, such as least squares SVM (LS-SVM), proximal SVM (PSVM), have been extensively used in the last two decades in various applications, including regression, classification and clustering [96, 30]. Originated from statistics, SVM can be regarded as a special form of SLFN and achieves better generalization performance than conventional neural networks for most problems [15].

2.4.1 Statistical learning theory

To begin with, we present a brief review of the statistical learning theory, on which SVM is built on. Given two random variables $\mathbf{x} \in \mathbf{R}^{1\times d}$ and $\mathbf{t} \in \mathbf{R}^{1\times m}$, a probabilistic relationship exists between them, defined as $P(\mathbf{x},\mathbf{t})$ over $\mathbf{R}^d \times \mathbf{R}^m$. One sample of $\mathbf{x}$ determines a probability distribution on $\mathbf{t}$ rather than a unique value. We are provided with N samples of $\mathbf{x},\mathbf{t}$, $\mathbf{X} \in \mathbf{R}^{N\times d}$ and $\mathbf{T} \in \mathbf{R}^{N\times m}$. $\mathbf{X}$ and $\mathbf{T}$ are the training data, from which we will learn and model the distribution.

The basic idea of statistical learning theory is: for a finite set of training data, the search for the optimal model of the distribution $P(\mathbf{x},\mathbf{t})$ needs to be constrained to a suitable small hypothesis space. Otherwise, the model may provide terrible generalization performance, even though it could fit the training data exactly. Vapnik formalized these concepts as *model capacity control* [23]. Subsequently, the *Structural Risk Minimization (SRM)* [100] was constructed and aimed to minimize the model capacity and empirical errors simultaneously. The mathematical expression is formulated as follows:

$$\begin{aligned} \min_{f\in\mathscr{H}} \quad & R_{\text{reg}} = \frac{\lambda}{2}\|f\|^2 + R_{\text{emp}}[f] \\ & R_{\text{emp}}[f] = \frac{1}{N}\sum_{i=1}^{N} c\big(\mathbf{t}_i, f(\mathbf{x}_i)\big) \end{aligned} \tag{2.6}$$

where $R_{\text{emp}}[f]$ is the empirical errors of the training data and $\frac{\lambda}{2}\|f\|^2$ is the model ca-

Table 2.2: List of common loss functions

Name	Loss function
ε-insensitive	$c(\xi) = \lvert\xi\rvert_\varepsilon$
Laplacian	$c(\xi) = \lvert\xi\rvert$
Gaussian	$c(\xi) = \frac{1}{2}\xi^2$
Polynomial	$c(\xi) = \frac{1}{p}\lvert\xi\rvert^p$

pacity. The loss function $c\big(\mathbf{t}_i, f(\mathbf{x}_i)\big)$ could have numerous forms. We should avoid very complicated function c as it may lead to difficult optimization problem and select the most suitable one for each particular problem. Table 2.2 contains some common loss functions [93].

2.4.2 SVM

SVM was initially developed to solve binary classification problems. The core idea of SVM is to construct a hyperplane in order to separate the training data with maximal margin. For multiclass problems, several binary classifiers are combined using one-against-one (OAO), one-against-all (OAA) or directed acyclic graph (DAG) methods [40].

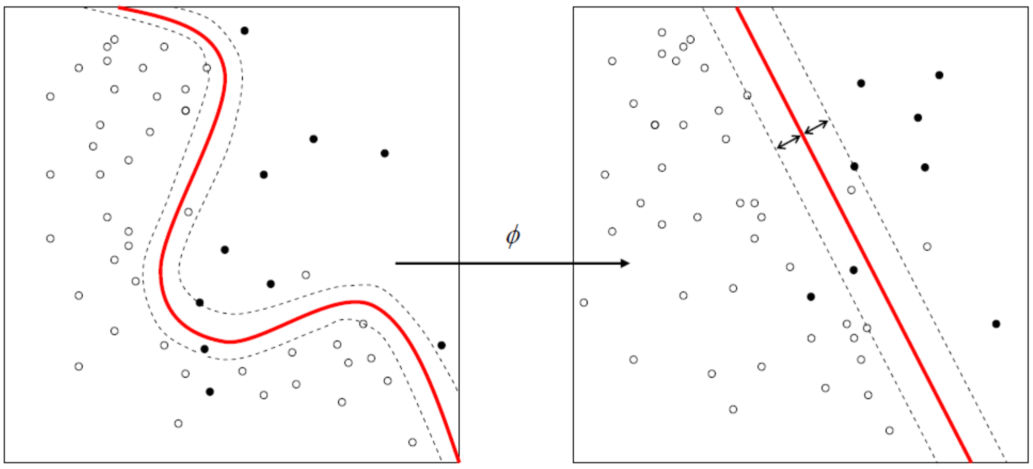


Figure 2.5: Nonlinear feature mapping

Assume we have the training data $\mathbf{X}_{N\times d} = \left[\mathbf{x}_1^T \ \cdots \mathbf{x}_N^T\right]^T, \mathbf{x}_i \in \mathbf{R}^{1\times d}, \mathbf{T}_{N\times 1} = [t_1 \ \cdots t_N]^T, t_i \in \{1, -1\}$. In most cases, the training data are linearly non-separable in the input space.

Thus, it requires nonlinear mappings from the input to a higher dimensional space: $\mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$ as shown in Fig. 2.5. In addition, even after the feature mapping, errors may still exist and should be allowed.

The primal form of SVM is constructed as follows:

$$\begin{aligned} &\text{Minimize: } \mathscr{L}_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i^{\sigma} \\ &\text{Subject to: } t_i\big(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b\big) \geq 1-\xi_i \\ &\qquad\qquad \xi_i \geq 0, \quad i=1,\cdots,N \end{aligned} \tag{2.7}$$

where C is a user-specified parameter and ξ_i is the slack variable to account for the errors. The SVM problem (2.7) is in general NP-complete. In order to avoid NP-completeness, $\sigma = 1$ is normally used [15]. And there are many efficient methods to find the solution if choosing $\sigma = 1$.

Standard optimization method is used to establish the Lagrangian function:

$$\mathscr{P} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\alpha_i\Big(t_i\big(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b\big)-(1-\xi_i)\Big) \tag{2.8}$$

where $\alpha_i > 0, \mu_i > 0$ are the Lagrange multipliers to be optimized. According to the Karush-Kuhn-Tucker (KKT) theorem [27], we could find the optimal solution (saddle point):

$$\begin{aligned} &\frac{\partial\mathscr{P}}{\partial\mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^{N}\alpha_i t_i\phi(\mathbf{x}_i) \\ &\frac{\partial\mathscr{P}}{\partial\xi} = 0 \Rightarrow C = \alpha_i + \mu_i, i=1,\cdots,N \\ &\frac{\partial\mathscr{P}}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N}\alpha_i t_i = 0 \end{aligned} \tag{2.9}$$

Table 2.3: List of common kernel functions

Name	Kernel function
Polynomial of degree d	$G(\mathbf{u},\mathbf{v}) = (1+\mathbf{u}\cdot\mathbf{v})^d$
Gaussian	$G(\mathbf{u},\mathbf{v}) = \exp(-\|\mathbf{u}-\mathbf{v}\|^2)$
Multiquadric	$G(\mathbf{u},\mathbf{v}) = \sqrt{(\|\mathbf{u}-\mathbf{v}\|^2+c^2)}$
Trigonometric polynomial of degree d	$G(\mathbf{u},\mathbf{v}) = \frac{\sin(d+1/2)(\mathbf{u}-\mathbf{v})}{\sin(\frac{\mathbf{u}-\mathbf{v}}{2})}$

Substituting (2.9) into (2.8), the dual form of SVM is generated:

$$\begin{aligned} &\text{Maximize: } \mathscr{L}_d = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) + \sum_{i=1}^{N}\alpha_i \\ &\text{Subject to: } \sum_{i=1}^{N}\alpha_i t_i = 0 \\ &\qquad 0 \le \alpha_i \le C, \quad i = 1,\cdots,N \end{aligned} \tag{2.10}$$

As observed from (2.10), the dual form only involves the dot product $\phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$. Thus, we do not need to know the explicit form of the feature mapping $\phi(\mathbf{x})$. Instead, kernel function, $K(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$ satisfying Mercer's conditions [15] can be adopted. Table 2.3 summarizes some common kernel functions.

The decision function of SVM is formulated as:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N}\alpha_i t_i K(\mathbf{x},\mathbf{x}_i) + b\right) = \text{sign}\left(\sum_{s=1}^{N_s}\alpha_s t_s K(\mathbf{x},\mathbf{x}_s) + b\right) \tag{2.11}$$

where $\mathbf{x}_s$ is Support Vector (SV) and N_s is the number of SVs. Apparently, the SVs ($\mathbf{x}_s$) are the samples ($\mathbf{x}_i$) with non-zero Lagrange multipliers (α_i).

2.4.3 Support vector regression (SVR)

The SVM formulation can be extended to deal with regression problems. Similarly, standard optimization method will be adopted. ε-insensitive loss function is chosen and the method is called ε-support vector regression (ε-SVR) [93].

$$c(t_i, f(\mathbf{x}_i)) = |t_i - f(\mathbf{x}_i)|_\varepsilon = \begin{cases} 0 & |t_i - f(\mathbf{x}_i)| < \varepsilon \\ |t_i - f(\mathbf{x}_i)| - \varepsilon & \text{otherwise} \end{cases} \tag{2.12}$$

Given the training data $\mathbf{X}_{N\times d} = \left[\mathbf{x}_1^T \ \cdots \mathbf{x}_N^T\right]^T, \mathbf{x}_i \in \mathbf{R}^{1\times d}, \mathbf{T}_{N\times 1} = [t_1 \ \cdots t_N]^T, t_i \in R$. The primal function of SVR is:

$$\begin{aligned} &\text{Minimize: } \mathscr{L}_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ &\text{Subject to: } t_i - \mathbf{w}\cdot\phi(\mathbf{x}_i) - b \le \varepsilon + \xi_i \\ &\qquad \mathbf{w}\cdot\phi(\mathbf{x}_i) + b - t_i \le \varepsilon + \xi_i^* \\ &\qquad \xi_i, \xi_i^* \ge 0, \quad i = 1, \cdots, N \end{aligned} \tag{2.13}$$

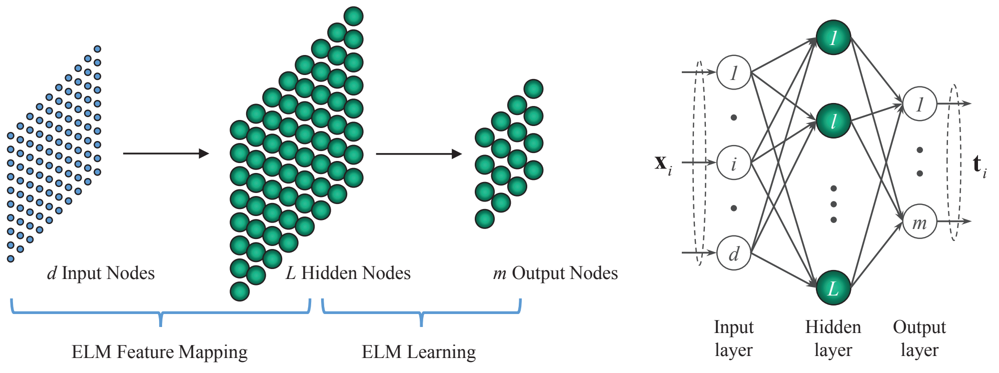
The constant C determines the trade-off between the model capacity and empirical errors. Thus, the Lagrangian of SVR is:

$$\begin{aligned} \mathscr{P} =& \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}\alpha_i\big(\varepsilon + \xi_i - t_i + \mathbf{w}\cdot\phi(\mathbf{x}_i) + b\big) \\ & - \sum_{i=1}^{N}\alpha_i^*\big(\varepsilon + \xi_i^* + t_i - \mathbf{w}\cdot\phi(\mathbf{x}_i) - b\big) - \sum_{i=1}^{N}(\eta_i\xi_i + \eta^*\xi_i^*) \end{aligned} \tag{2.14}$$

where the Lagrange multipliers $\alpha_i^{(*)}, \eta_i^{(*)} > 0$ and $\alpha_i^{(*)}$ denotes α_i and α^*.

Solving the Lagrangian (2.14) and substituting back into the primal form (2.13), we obtain the dual optimization problem:

$$\begin{aligned} &\text{Maximize: } \mathscr{L}_d = -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(\mathbf{x}_i, \mathbf{x}_j) - \varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N}t_i(\alpha_i - \alpha_i^*) \\ &\text{Subject to: } \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ &\qquad 0 \le \alpha_i, \alpha_i^* \le C, \quad i = 1, \cdots, N \end{aligned} \tag{2.15}$$



(a) Two-stage ELM general architecture (b) The network of initial ELM

Figure 2.6: The architecture of ELM

where $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ is the kernel function. Consequently, the decision function of ε-SVR is:

$$f(\mathbf{x}) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) K(\mathbf{x}, \mathbf{x}_i) + b = \sum_{s=1}^{N_s} (\alpha_s - \alpha_s^*) K(\mathbf{x}, \mathbf{x}_s) + b \tag{2.16}$$

There are more variants of SVM, proposed for different purposes. For instance, the least squares SVM (LS-SVM) [96] and proximal SVM (PSVM) [30] were proposed to avoid the quadratic programming (QP) problem by enforcing equality constraints rather than the inequality constraints used in the initial SVM. Moreover, ν-SVM was suggested which used the parameter $\nu \in [0,1]$ to replace the original $C \in [0, +\infty)$ to control the tradeoff between model capacity and empirical errors [13].

2.5 Extreme Learning Machine (ELM)

Extreme learning machine (ELM) was originally proposed as a variant and improvement of single-hidden layer feedforward neural networks (SLFNs). Unlike the common understanding of neural networks, ELM theories demonstrate that hidden nodes need not be adjusted even though they are indeed important [46, 44, 45]. Instead, it is able to perform the learning without iterative tuning of the hidden nodes as long as the activation functions of these hidden nodes are nonlinear piecewise continuous.

Subsequently, ELM was further developed to the “generalized” SLFNs. Fig. 2.6(a) depicts the general architecture of ELM in two stages: *ELM Feature Mapping* and *ELM Learning*. ELM feature mapping is composed of single or multiple types of random hidden nodes, independent of the training data. And ELM learning concentrates on how to obtain the output weights when facing different applications (representational learning, regression, classification, clustering, etc.). In essence, ELM tends to reach both smallest training error and smallest norm of output weights [51, 43].

In recent years, ELM has been thoroughly investigated and successfully applied to different tasks, including but not limited to chemical process [110], bio-informatics [108], remote sensing [112], computer vision [75]. Furthermore, ELM auto-encoder [58] also produces state-of-the-art accuracy on MNIST dataset, a common benchmark for deep learning methods, with much faster learning speed.

2.5.1 Basic ELM

Algorithm 1: Basic ELM learning algorithm

```
1: Initialization: Given the activation function $G$ satisfying universal approximation
   conditions and the number $L$;
2: Generate input weight $\mathbf{a}_i \in \mathbf{R}^{1\times d}$ and bias $b_i \in \mathbf{R}$ randomly, $i = 1,...,L$;
3: Obtain hidden layer output matrix $\mathbf{H}$;
4: Calculate output weight as $\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$;
```

Huang *et al.* theoretically proved and proposed the basic ELM in [51, 50]. Fig. 2.6(b) shows the network of ELM. For an input $\mathbf{x}$, L hidden nodes are constructed and denoted by $\mathbf{h}(\mathbf{x})$:

$$\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}) \ \cdots \ h_L(\mathbf{x})] = [G(\mathbf{a}_1, b_1, \mathbf{x}), \ \cdots, \ G(\mathbf{a}_L, b_L, \mathbf{x})] \in \mathbf{R}^{1\times L} \tag{2.17}$$

where $G(\mathbf{a}, b, \mathbf{x})$ is a nonlinear piecewise continuous function that needs to satisfy ELM universal approximation conditions [46, 44, 45]. $\{\mathbf{a}_i, b_i\}_{i=1}^{L}$ are input weights generated randomly based on any continuous probability distribution. Common activation functions G include *sigmoid* function, *hard-limit* function, *Gaussian* function, etc.

Given the training data $\mathbf{X}_{N\times d} = \left[\mathbf{x}_1^T \ \cdots \mathbf{x}_N^T\right]^T, \mathbf{x}_i \in \mathbf{R}^{1\times d}, \mathbf{T}_{N\times m} = \left[\mathbf{t}_1^T \ \cdots \mathbf{t}_N^T\right]^T, \mathbf{t}_i \in \mathbf{R}^{1\times m}$, the hidden layer matrix $\mathbf{H}$ is built up as follows:

$$\begin{aligned}
\mathbf{H} &= \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \mathbf{h}(\mathbf{x}_2) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix}_{N\times L}, \mathbf{T} = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \\ \vdots \\ \mathbf{t}_N \end{bmatrix}_{N\times m} \\
\mathbf{H}\boldsymbol{\beta} &= \mathbf{T} \\
\Rightarrow \boldsymbol{\beta} &= \mathbf{H}^{\dagger}\mathbf{T}
\end{aligned} \tag{2.18}$$

where $\boldsymbol{\beta}$ is the output weight connecting hidden and output layer.

2.5.2 Unified ELM

ELM was further advanced by some later works, providing a single framework to simplify and unify different learning methods, such as SLFNs, LS-SVM, RBF network, etc. It was proved that ELM has universal approximation capability [49], thus enabling it to be applicable for different kinds of problems.

Theorem 2.1 *Universal approximation capability: given any continuous target function $f(\mathbf{x})$, there always exists a series of β_i in some nonlinear feature space $h_i(\mathbf{x})$ that can approximate it.*

$$\lim_{L\to+\infty} \|\sum_{i=1}^{L} \beta_i h_i(\mathbf{x}) - f(\mathbf{x})\|_2 = 0 \tag{2.19}$$

Theorem 2.2 *Classification capability: provided a random feature mapping $\mathbf{h}(\mathbf{x})$, if $\mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ is dense in $C(\mathbf{R}^d)$ or in $C(M)$, where M is a compact set of $\mathbf{R}^d$, then the "gen-*

eralized" SLFNs with $\mathbf{h}(\mathbf{x})$ as the hidden layer mapping can separate arbitrary disjoint regions in $\mathbf{R}^d$ or M [47, 49].

The unified ELM aims to minimize the combination of model capacity and empirical errors [49]:

$$\begin{aligned}&\text{Minimize: } \mathscr{L}_{p,\mathrm{ELM}} = \frac{1}{2}\|\mathbf{w}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 \\ &\text{Subject to: } \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = \mathbf{t}_i^T - \boldsymbol{\xi}_i^T, \quad i = 1, \cdots, N\end{aligned} \tag{2.20}$$

The solution is derived by following the standard optimization method. The feature mapping $\mathbf{h}(\mathbf{x})$ in ELM is randomly generated and known to users. Thus, different from SVM, ELM can be solved directly without invoking any kernel functions. However, we can also use an implicit mapping and adopt any kernel that satisfies Mercer's conditions.

(1) *Non-kernel case:*

$$\boldsymbol{\beta} = \begin{cases} \mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} & \text{if } N \le L \\ \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} & \text{if } N > L \end{cases} \tag{2.21}$$

(2) *Kernel case:*

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}\left(\frac{\mathbf{I}}{C} + \boldsymbol{\Omega}_{\mathrm{ELM}}\right)^{-1}\mathbf{T} \tag{2.22}$$

where $\boldsymbol{\Omega}_{\mathrm{ELM}}$ is called ELM kernel matrix with elements:

$$\boldsymbol{\Omega}_{\mathrm{ELM}} = \mathbf{H}\mathbf{H}^T : \ \Omega_{\mathrm{ELM}i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.23}$$

2.5.3 Other variants of ELM

More variants of ELM have been proposed for different applications. For instance, previous works determine the number of hidden nodes L by trial-and-error method. In order to solve this issue, some algorithms with adjustable structures were introduced to adjust L heuristically. Incremental ELM (I-ELM) adds one random hidden node and calculates the output weight between the new hidden node to the output node in each step [46]. Furthermore, convex I-ELM (CI-ELM) was later suggested in order to improve the convergence rate. The output weights of existing nodes would be re-calculated when a new node is included each time [44]. Moreover, the error minimized ELM (EM-ELM) allows adding the hidden nodes one-by-one or group-by-group, introducing more flexibility into the incremental ELM algorithms [24].

In addition, data may become available sequentially in some problems, making the previous batch-based ELM algorithms unsuitable to use. Liang *et al.* developed the online sequential ELM (OS-ELM), which could learn from sequential data of fixed or varying size [69].

Furthermore, ELM was further extended to solve semi-supervised or unsupervised learning problems based on manifold regularization [42, 67]. It presents great learning capability and high efficiency while being able to handle multiclass classification or clustering directly. Additionally, the ELM based auto-encoder (ELM-AE), which was proposed in [58], learns data representations with singular values and discovers the underlying relationship within the data.

2.6 Summary

Machine learning is data-driven and aims to enable machines for decision making. There are many different types of machine learning techniques, generically belonging to four categories, supervised, unsupervised, semi-supervised and reinforcement learning. Popular algorithms include neural networks, SVM, K-nearest neighbors (K-NN), decision tree, random forests, etc.

Within the family of neural networks, SLFN is the most popular type because of its simple structure and great approximation capability of complicated nonlinear mappings. Additionally, CNN shows advantages when facing problems with strong local correlations, such as image processing, speech recognition and similar tasks. The basic method to train the SLFN and CNN is the back-propagation (BP) algorithm, invoking numerous trivial issues, such as slow convergence rate, intensive parameter tunings, numerous computations. In contrast, ELM randomly generates the input weights and analytically calculates the output weights. Thus, it does not require any gradient descent steps and exhibits abundant advantages. Different variants of ELM have been developed, suitable for diverse applications, including classification, regression, clustering, representational learning, etc.

Chapter 3

Sparse Extreme Learning Machine for Classification

In this chapter, we present a sparse extreme learning machine (ELM) for classification. Unlike the unified ELM, which gives a dense solution, it provides a sparse solution, largely reducing the storage space and testing time. Furthermore, we specifically develop an efficient training algorithm for sparse ELM for classification. It breaks down the large quadratic programming (QP) problem into a series of smallest possible sub-problems, each of which includes only one Lagrange multiplier to be calculated. Consequently, each sub-problem can be easily solved in an analytical way. More importantly, it only requires computational complexity quadratic to the training size N, while that of the unified ELM is between quadratic and cubic. As a result, it greatly accelerates the training phase when facing large-scale applications.

In summary, compared with the traditional SVM, sparse ELM realizes better generalization performance with much faster training speed, up to *500 times*. And comparing to the unified ELM, it presents comparable generalization performance while greatly scaling down the training time, testing time and storage requirement, especially for large-scale problems.

3.1 Sparse ELM for Classification

An optimization method based ELM was initially developed in [48] to handle classification problems. It uses inequality constraints and generate a sparse network. However, it can only use random hidden nodes as the feature mapping.

In this section, we conduct a thorough investigation on sparse ELM, where both kernels and random hidden nodes are applicable. Furthermore, we show that sparse ELM unifies different classification methods, including but not limited to conventional SVM, SLFNs, radial basis function (RBF) networks, etc.

3.1.1 Problem formulation

The sparse ELM is proposed to solve binary classification. And when encountering multiclass problems, one-against-one (OAO), one-against-all (OAA), directed acyclic graph (DAG) methods will be utilized to combine several binary classifiers together [40]. The training data provided is: $\mathbf{X}_{N\times d} = \left[\mathbf{x}_1^T \ \cdots \mathbf{x}_N^T\right]^T, \mathbf{x}_i \in \mathbf{R}^{1\times d}, \mathbf{T}_{N\times 1} = [t_1 \ \cdots t_N]^T, t_i \in \{1,-1\}$.

Feature mapping

As discussed before, in most cases, the training data are not linearly separable in the input space. Consequently, a nonlinear mapping is necessary to transform the data from the input space to a higher dimensional feature space. The feature mapping could be generated randomly as proved in ELM theories [51, 46].

$$\mathbf{x}_i \in \mathbf{R}^{1\times d} \rightarrow \mathbf{h}(\mathbf{x}_i) \in \mathbf{R}^{1\times L}, \quad i = 1,\cdots,N \tag{3.1}$$

where L is the number of hidden nodes. Except for the random feature mapping in (3.1), kernels are also applicable as long as meeting Mercer's conditions.

Optimization

It was proved that a continuous function $f(\mathbf{x})$ is able to separate any any disjoint regions in $\mathbf{R}^d$. In addition, ELM provides universal approximation capability [46], which means ELM could approximate any target function $f(\mathbf{x})$ in order to separate any disjoint regions in $\mathbf{R}^d$:

$$\lim_{L\to+\infty} \|\sum_{i=1}^{L} \beta_i h_i(\mathbf{x}) - f(\mathbf{x})\| = 0 \tag{3.2}$$

As a consequence, the bias b that is necessary in conventional SVM can be removed. However, in real implementations, the number of hidden nodes L cannot grow infinitely. Hence, training errors ξ_i's should be allowed. Furthermore, the generalization performance is guaranteed by minimizing both empirical errors $\left(\sum_{i=1}^{N} \xi_i\right)$ and model capacity ($\|\boldsymbol{\beta}\|^2$) based on statistical learning theory and structural risk minimization (SRM) [100, 23]. Overfitting problems are well solved and a great generalization performance will be presented.

The sparse solution is produced by enforcing inequality constraints in the primal problem as follows:

$$\begin{aligned} &\text{Minimize: } \mathscr{L}_{p,\mathrm{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N} \xi_i \\ &\text{Subject to: } t_i\, \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} \geq 1 - \xi_i \\ &\qquad\qquad\quad \xi_i \geq 0, \quad i = 1, \dots, N \end{aligned} \tag{3.3}$$

Naturally, the Lagrangian function is built up:

$$\mathscr{P}_{\mathrm{ELM}}(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \mu_i \xi_i - \sum_{i=1}^{N} \alpha_i \cdot \left(t_i \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - (1 - \xi_i)\right) \tag{3.4}$$

The optimal solution will be obtained at the saddle point:

$$\begin{aligned}
&\frac{\partial \mathscr{P}_{\text{ELM}}}{\partial \boldsymbol{\beta}} = 0 \Rightarrow \boldsymbol{\beta} = \sum_{i=1}^{N} \alpha_i t_i \mathbf{h}(\mathbf{x}_i)^T = \sum_{s=1}^{N_s} \alpha_s t_s \mathbf{h}(\mathbf{x}_s)^T \\
&\frac{\partial \mathscr{P}_{\text{ELM}}}{\partial \boldsymbol{\xi}} = 0 \Rightarrow C = \alpha_i + \mu_i
\end{aligned} \tag{3.5}$$

The dual form of sparse ELM for classification is obtained by substituting the results of (3.5) into (3.4):

$$\begin{aligned}
&\text{Minimize: } \mathscr{L}_{d,\text{ELM}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j t_i t_j \Omega_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{N} \alpha_i \\
&\text{Subject to: } 0 \le \alpha_i \le C, \qquad i = 1, \ldots, N
\end{aligned} \tag{3.6}$$

where $\boldsymbol{\Omega}_{\text{ELM}}$ is the ELM kernel matrix:

$$\Omega_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j) \tag{3.7}$$

Therefore, the decision function of sparse ELM is calculated:

$$\begin{aligned}
&f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x}) \left(\sum_{i=1}^{N} \alpha_i t_i \mathbf{h}(\mathbf{x}_i)^T \right) = \mathbf{h}(\mathbf{x}) \left(\sum_{s=1}^{N_s} \alpha_s t_s \mathbf{h}(\mathbf{x}_s)^T \right) = \sum_{s=1}^{N_s} \alpha_s t_s \Omega_{\text{ELM}}(\mathbf{x}, \mathbf{x}_s) \\
&\text{sign}\big(f(\mathbf{x})\big) = \begin{cases} \text{sign}\left[\mathbf{h}(\mathbf{x}) \left(\sum_{s=1}^{N_s} \alpha_s t_s \mathbf{h}(\mathbf{x}_s)^T\right)\right] & \text{non-kernel case} \\ \\ \text{sign}\left(\sum_{s=1}^{N_s} \alpha_s t_s \Omega_{\text{ELM}}(\mathbf{x}, \mathbf{x}_s)\right) & \text{kernel case} \end{cases}
\end{aligned} \tag{3.8}$$

where $\mathbf{x}_s$ is the support vector (SV), and N_s is the number of SVs.

Remark 3.1 *Errors ξ_i's are allowed in the primal objective function* (3.3) *to account for the samples that are not correctly classified. Thus, the proposed sparse ELM is sub-optimal in the sense that it constructs a soft margin.*

3.1.2 Sparsity analysis

As in (3.5), the optimal solution is calculated at the saddle point. And the KKT conditions are:

$$\begin{aligned} &\alpha_i\big(t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - (1-\xi_i)\big) = 0 \\ &\mu_i\xi_i = 0 \end{aligned} \tag{3.9}$$

For the support vectors (SVs), corresponding Lagrange multipliers need to be non-zero. Two possible cases are displayed:

(1) $0 < \alpha_i < C$:

$$\begin{aligned} &\mu_i > 0 \Rightarrow \xi_i = 0 \\ &\alpha_i > 0 \Rightarrow t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - 1 = 0 \end{aligned} \tag{3.10}$$

In this case, the data is on the decision hyperplane (separating boundary).

(2) $\alpha_i = C$:

$$\begin{aligned} &\mu_i = 0 \Rightarrow \xi_i > 0 \\ &\alpha_i > 0 \Rightarrow t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - (1-\xi_i) = 0 \\ &\qquad\quad \Rightarrow t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - 1 < 0 \end{aligned} \tag{3.11}$$

In this case, the data is classified with error.

Remark 3.2 *In contrast to the unified ELM, in which almost all ξ_i's are non-zero, the sparse ELM invokes errors ξ_i's only when the inequality constraint $\big(t_i\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - 1 > 0\big)$ are violated as aforementioned.*

Let us consider the training data and the distribution from which the data are sampled. Apparently, only a part of them would be on the boundary or classified with errors.

Therefore, only a part of all training data are support vectors (SVs), making the solution sparse.

As easily observed from Fig. 3.1, the proposed method provides a sparse dual network since all non-SVs are excluded. Considering the primal network, the architecture remains the same as the number of hidden nodes L is fixed once chosen. However, sparsity also ease the computational burden for $\boldsymbol{\beta}$ since some components are removed as shown in (3.5). The computations required in the testing phase and storage space are both proportional to the number of SVs (N_S). As a consequence, sparse ELM largely reduces the storage space and testing time compared with the unified ELM.

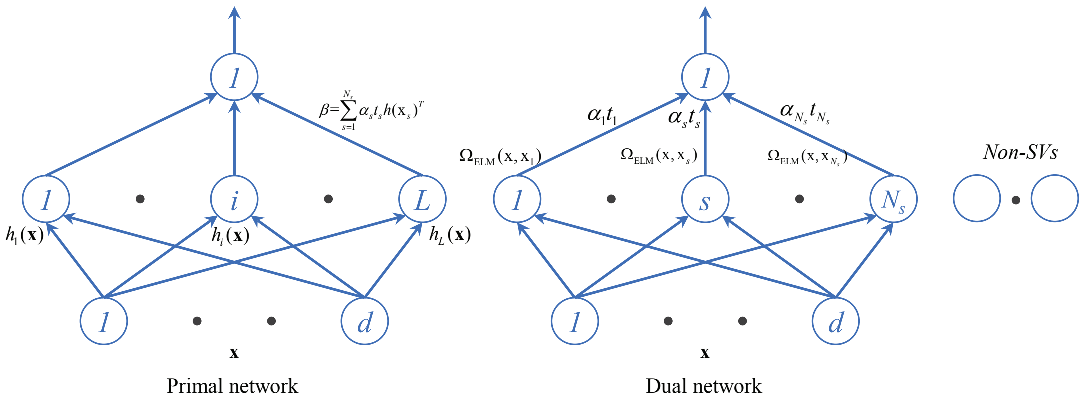


Figure 3.1: Primal and dual networks of sparse ELM for classification

3.1.3 Unified framework for different classification methods

As depicted in Fig. 3.1, the primal network of sparse ELM and "generalized" SLFNs share the same architecture. Additionally, the dual network of sparse ELM bears resemblance to the dual of SVM [15]. Furthermore, both RBF kernels (i.e. Gaussian kernel) and RBF hidden nodes are applicable in the sparse ELM. Therefore, the proposed sparse ELM provides a unified framework for different classification methods, including but not limited to "generalized" SLFNs, SVM, RBF networks, etc.

3.1.4 ELM kernel matrix $\mathbf{\Omega}_{\text{ELM}}$

The proposed sparse ELM could use a random feature mapping or a kernel function to obtain the solution. Hence, similar to the unified ELM [49], it also has the kernel and non-kernel cases. At here, we present them in detail.

Non-kernel case

The ELM kernel matrix $\mathbf{\Omega}_{\text{ELM}}$ is calculated from random hidden nodes directly.

$$\mathbf{h}(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \cdots, G(\mathbf{a}_L, b_L, \mathbf{x})] \tag{3.12}$$

where G is the activation function and $\{\mathbf{a}_i, b_i\}_{i=1}^{L}, \mathbf{a}_i \in \mathbf{R}^{1\times L}, b_i \in \mathbf{R}$ are the input weights connecting to the hidden nodes. $\{\mathbf{a}_i, b_i\}_{i=1}^{L}$ are generated randomly and the function G needs to satisfy ELM universal approximation conditions [46].

$$\mathbf{\Omega}_{\text{ELM}} = \mathbf{H}\mathbf{H}^T \tag{3.13}$$

It can use either additive hidden nodes or RBF ones. In the following, the former two are additive nodes and the latter two are RBF nodes.

(1) *Sigmoid function*:

$$G(\mathbf{a}, b, \mathbf{x}) = \frac{1}{1 + \exp\big(-(\mathbf{a}\cdot\mathbf{x} + b)\big)} \tag{3.14}$$

(2) *Sinusoid function*:

$$G(\mathbf{a}, b, \mathbf{x}) = \sin(\mathbf{a}\cdot\mathbf{x} + b) \tag{3.15}$$

(3) *Multiquadric function*:

$$G(\mathbf{a}, b, \mathbf{x}) = \sqrt{(\|\mathbf{x} - \mathbf{a}\|^2 + b^2)} \tag{3.16}$$

(4) *Gaussian function*:

$$G(\mathbf{a}, b, \mathbf{x}) = \exp(-\frac{\|\mathbf{x} - \mathbf{a}\|^2}{b}) \tag{3.17}$$

Kernel case

We can also obtain the ELM kernel matrix $\boldsymbol{\Omega}_{\mathrm{ELM}}$ with kernel functions directly as in (3.7). The only requirement for the kernel function K is to satisfy Mercer's conditions [15, 16]. The function K could be, but not limited to:

(1) *Gaussian kernel:*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}) \tag{3.18}$$

(2) *Laplacian kernel:*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma}), \quad \sigma > 0 \tag{3.19}$$

(3) *Polynomial kernel:*

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^d, \quad d \in \mathbf{Z}^+ \tag{3.20}$$

Remark 3.3 *ELM provides universal approximation ability if the activation function G satisfies the conditions given in [46]. Various types of functions are applicable with random input weights. Alternatively, they can also be constructed based on some implicit relationship. In this case, the feature mapping* $\mathbf{h}(\mathbf{x})$ *is unknown to the users and the kernel function K will be adopted, which needs to meet Mercer's conditions.*

Theorem 3.1 *The dual problem of sparse ELM for classification* (3.6) *is convex.*

Proof: The first-order partial derivative is calculated as:

$$\frac{\partial \mathscr{L}_{d,\mathrm{ELM}}}{\partial \alpha_s} = t_s \sum_{j=1}^{N} \alpha_j t_j \Omega_{\mathrm{ELM}}(\mathbf{x}_s, \mathbf{x}_j) - 1 \tag{3.21}$$

And the second-order partial derivative is:

$$\frac{\partial^2 \mathscr{L}_{d,\mathrm{ELM}}}{\partial \alpha_t \partial \alpha_s} = t_t t_s \Omega_{\mathrm{ELM}}(\mathbf{x}_t, \mathbf{x}_s) \tag{3.22}$$

Consequently, the Hessian matrix $\nabla^2 \mathscr{L}_{d,\mathrm{ELM}} = \mathbf{T}^T \boldsymbol{\Omega}_{\mathrm{ELM}} \mathbf{T}$ is obtained.

(1) When it is non-kernel case, $\boldsymbol{\Omega}_{\mathrm{ELM}}$ will be calculated from random hidden nodes directly as (3.13):

$$\nabla^2 \mathscr{L}_{d,\mathrm{ELM}} = \mathbf{T}^T \mathbf{H}\mathbf{H}^T \mathbf{T} = (\mathbf{T}^T \mathbf{H}) \mathbf{I}_{L\times L} (\mathbf{T}^T \mathbf{H})^T \tag{3.23}$$

Obviously, $\nabla^2 \mathscr{L}_{d,\mathrm{ELM}}$ is positive semi-definite.

(2) When it is kernel case, $\boldsymbol{\Omega}_{\mathrm{ELM}}$ will be calculated with the function K. Thus, $\boldsymbol{\Omega}_{\mathrm{ELM}}$ is guaranteed to be positive semi-definite by the Mercer's conditions. Hence, $\nabla^2 \mathscr{L}_{d,\mathrm{ELM}} = \mathbf{T}^T \boldsymbol{\Omega}_{\mathrm{ELM}} \mathbf{T} \geq 0$ is positive semi-definite.

The positive semi-definiteness of the Hessian matrix $\nabla^2 \mathscr{L}_{d,\mathrm{ELM}}$ is the sufficient condition for the function $\mathscr{L}_{d,\mathrm{ELM}}$ to be convex. As a result, the dual problem of sparse ELM (3.6) is convex. ■

3.2 Training Algorithm of Sparse ELM for Classification

In effect, sparse ELM for classification is a quadratic programming (QP) problem. It is similar to the conventional SVM (2.10) with an important distinction that sparse ELM does not have the sum constraint $\sum_{i=1}^{N} \alpha_i t_i = 0$ [1]. Therefore, the proposed sparse ELM searches for the optimal solution within a wider range, enabling it to present better generalization performance than the conventional SVM. Additionally, the training method will be easier as fewer constraints need to be satisfied. However, early works only discussed the sparse ELM from theoretical perspectives without implementation considerations [48].

Inspired by the ideas of sequential minimal optimization (SMO) [87], which divides the large QP problem into different sub-problems, we specifically develop a training algorithm for the proposed sparse ELM for classification and fully explore its advantages. It divides the large QP problem into a series of sub-problems, each of which includes only one Lagrange multiplier and can be solved analytically. Through iterative steps, we solve these sub-problems one-by-one until certain condition is reached to trigger the termination action.

3.2.1 Brief review of SMO

To begin with, the SMO algorithm proposed by Platt *et al.* will be briefly reviewed [87]. Prior to the SMO algorithm, the training of SVM requires numerical calculation of a large QP optimization problem. The requirement for memory and computational ability grows rapidly with the increase of the training size N.

SMO is developed accordingly to solve these issues. It splits the large QP problem into

[1] In some cases, SVM can also be trained without the bias, and thus no sum constraint, when satisfying certain conditions [95].

a series of smallest possible sub-problems. Each sub-problem includes two Lagrange multipliers and could be solved analytically. Consequently, the time-consuming numerical QP optimization is avoided, providing an efficient solution for SVM. In addition, one major computation of SMO comes from the evaluation of kernel function. Thus, it would be further accelerated when using linear kernel or dealing with sparse data.

3.2.2 Optimality conditions

We use the optimality conditions to decide whether the optimal solution has been reached or not. If the conditions are met, the optimal solution is thus reached, and *vice versa*.

The KKT conditions are provided in (4.6). Three possible cases are listed as follows:

(1) $\alpha_i = 0$:

$$\begin{aligned} &\alpha_i = 0 \Rightarrow t_i f(\mathbf{x}_i) - 1 \geq 0 \\ &\mu_i = C \Rightarrow \xi_i = 0 \end{aligned} \tag{3.24}$$

(2) $0 < \alpha_i < C$:

$$\begin{aligned} &\alpha_i > 0 \Rightarrow t_i f(\mathbf{x}_i) - 1 = 0 \\ &\mu_i > 0 \Rightarrow \xi_i = 0 \end{aligned} \tag{3.25}$$

(3) $\alpha_i = C$:

$$\begin{aligned} &\alpha_i = C \Rightarrow t_i f(\mathbf{x}_i) - 1 \leq 0 \\ &\mu_i = 0 \Rightarrow \xi_i > 0 \end{aligned} \tag{3.26}$$

3.2.3 Improvement strategy

When the optimality conditions are not fully satisfied, how should we further decrease the objective function $\mathscr{L}_{d,\text{ELM}}$ (3.6). The improvement strategy is formulated accordingly.

Based on the selection criteria, which will be presented later, α_c is chosen to be updated at the current step. The first- and second- order partial derivatives of the objective function $\mathscr{L}_{d,\text{ELM}}$ with regard to α_c are constructed:

$$\begin{aligned} \frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_c} &= t_c \sum_{j=1}^{N} \alpha_j t_j \Omega_{\text{ELM}}(\mathbf{x}_c, \mathbf{x}_j) - 1 = t_c f(\mathbf{x}_c) - 1 \\ \frac{\partial^2 \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_c^2} &= \Omega_{\text{ELM}}(\mathbf{x}_c, \mathbf{x}_c) \end{aligned} \tag{3.27}$$

It has been proved that the dual problem of sparse ELM for classification (3.6) is a convex quadratic one based on Theorem 3.1. Hence, the global minimum α_c^* does exist and can be reached [81]:

$$\alpha_c^* = \alpha_c + \frac{-\frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_c}}{\frac{\partial^2 \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_c^2}} = \alpha_c + \frac{1 - t_c f(\mathbf{x}_c)}{\Omega_{\text{ELM}}(\mathbf{x}_c, \mathbf{x}_c)} \tag{3.28}$$

The bounding constraints $[0, C]$ decide the limit of α_i's. Thus, constraints $[0, C]$ enforce the unstrained minimum α_c^* within the range and the constrained minimum α_c^{new} is thus calculated.

$$\alpha_c^{\text{new}} = \alpha_c^{*,\text{constrained}} = \begin{cases} 0 & \alpha_c^* < 0 \\ \alpha_c^* & 0 < \alpha_c^* < C \\ C & \alpha_c^* > C \end{cases} \tag{3.29}$$

3.2.4 Selection criteria

The selection of the Lagrange multiplier to be updated in each step is essential. Ideally speaking, it would be the best to choose the Lagrange multiplier that reduces the objective function $\mathscr{L}_{d,\mathrm{ELM}}$ the most. Nevertheless, it is time consuming and computation intensive to calculate the exact decrease of $\mathscr{L}_{d,\mathrm{ELM}}$ that each Lagrange multiplier could cause. Therefore, we recommend an estimate method: using the step size of α_i to estimate the decease of $\mathscr{L}_{d,\mathrm{ELM}}$ that α_i brings.

Definition 3.1 **d** *is the update direction.* d_i *denotes whether* α_i *should be increased or decreased: 1)* $d_i = 1$*, increased; 2)* $d_i = -1$*, decreased; 3)* $d_i = 0$*, increased or decreased are both acceptable.*

(1) $\alpha_i = 0$: α_i is on the left boundary of the constraints $[0, C]$. Thus, it can only be increased. Therefore, $d_i = 1$.

(2) $0 < \alpha_i < C$: d_i should be along the direction to reduce the objective function $\mathscr{L}_{d,\mathrm{ELM}}$. Therefore, $d_i = -\mathrm{sign}\left(\frac{\partial \mathscr{L}_{d,\mathrm{ELM}}}{\partial \alpha_i}\right)$.

(3) $\alpha_i = C$: α_i is on the right boundary of the constraints $[0, C]$ and can only be decreased. Therefore, $d_i = -1$.

Definition 3.2 **J** *is the selection parameter:*

$$J_i = \left(\frac{\partial \mathscr{L}_{d,ELM}}{\partial \alpha_i}\right) d_i, \quad i = 1, 2, \ldots, N \tag{3.30}$$

The Lagrange multiplier with the minimal selection parameter J_i will be selected in the current iteration.

$$c = \arg \min_{i=1,\ldots,N} J_i \tag{3.31}$$

Theorem 3.2 *The chosen Lagrange multiplier α_c will decrease the object function $\mathscr{L}_{d,ELM}$ as expected.*

Proof: In the training process, at least one data violates the optimality conditions (3.24)-(3.26). Otherwise, it would be determined that the optimal solution has been reached and the training algorithm will be terminated. Assume that the data corresponding to α_v violates the optimality conditions. Three possible cases are displayed:

(1) $\alpha_v = 0$:

$$\begin{aligned} \Rightarrow \frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_v} &= t_v f(\mathbf{x}_v) - 1 < 0 \\ J_v = \left(\frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_v}\right) &\cdot 1 < 0 \end{aligned} \tag{3.32}$$

(2) $0 < \alpha_v < C$:

$$\begin{aligned} \Rightarrow \frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_v} &= t_v f(\mathbf{x}_v) - 1 \neq 0 \\ J_v = \frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_v} &\cdot \left(-\text{sign}\left(\frac{\partial \mathscr{L}_d}{\partial \alpha_v}\right)\right) < 0 \end{aligned} \tag{3.33}$$

(3) $\alpha_v = C$:

$$\begin{aligned} \Rightarrow \frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_v} &= t_v f(\mathbf{x}_v) - 1 > 0 \\ J_v = \left(\frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \alpha_v}\right) &\cdot (-1) < 0 \end{aligned} \tag{3.34}$$

Therefore, $\left(\min\limits_{i=1,\dots,N} J_i\right)$ is always negative in the training process and the objective function $\mathscr{L}_{d,\text{ELM}}$ is guaranteed to be decreased after every iteration. ■

3.2.5 Termination condition

The training algorithm is based on iterative update. It would be excessively difficult to find the exact match of the optimality conditions. In fact, it would be enough to satisfy the optimality conditions within a tolerance ε. It has been discovered that a tolerance equals to the square root of the machine epsilon is good enough to produce stable results [26]. Similar to the SMO implementation [87], we choose $\varepsilon = 10^{-3}$.

When $\left(\min_{i=1,\ldots,N} J_i\right) > -\varepsilon$, the optimality conditions are fulfilled within the tolerance ε, and the training algorithm will be terminated.

3.2.6 Convergence analysis

Theorem 3.3 *The training algorithm proposed in this paper will converge to the global optimum in a finite number of iterations.*

Proof: As proved in Theorem 3.1, the dual problem of sparse ELM (3.6) is a convex QP problem. Additionally, the algorithm chooses a Lagrange multiplier α_c that violates the optimality conditions in each step. And the update of each step would make α_c satisfy the conditions and is guaranteed to monotonically reduce the objective function $\mathscr{L}_d$ as proved in Theorem 3.2.

Moreover, the Lagrange multipliers are all bounded within $[0, C]^N$. Based on the Osuna's theorem proved in [84], the algorithm will convergee to the global optimal solution in a finite number of iterations. ■

3.2.7 Training algorithm

Algorithm 2 summarizes the training algorithm of sparse ELM for classification. **g** is the gradient of $\mathscr{L}_{d,\mathrm{ELM}}$ and $g_i = \frac{\partial \mathscr{L}_{d,\mathrm{ELM}}}{\partial \alpha_i}$. Additionally, **d** is the update direction defined in Definition 3.1 and $\mathbf{J}$ is the selection parameter as in Definition 3.2. And $G_{i,j} = t_i t_j \Omega_{\text{ELM}}(\mathbf{x}_i, \mathbf{x}_j)$.

Algorithm 2: Sparse ELM for classification

```
Problem formulation: Provided the training data $\mathbf{X} \in \mathbf{R}^{N\times d}, \mathbf{T} \in \mathbf{R}^{N\times 1}$, we
construct the QP problem (3.6) with an appropriate ELM kernel matrix
$\boldsymbol{\Omega}_{\text{ELM}}$ and parameter $C$;
  1: Initialization: $\boldsymbol{\alpha} = \mathbf{0},\ \mathbf{g} = \mathbf{G}\boldsymbol{\alpha} - \mathbf{1},\ \mathbf{J} = \mathbf{g},\ \mathbf{d} = \mathbf{1},\ \boldsymbol{\alpha},\ \mathbf{g},\ \mathbf{J},\ \mathbf{d} \in \mathbf{R}^N$;
  2: While $\min_{i=1,\ldots,N} J_i < -\varepsilon$:
       1) Update the selection parameter $\mathbf{J}$, $J_i = g_i\, d_i$;
       2) Find the index $c$ with the minimal $J_i$, $c = \arg\min_{i=1,\ldots,N} J_i$. And update the
       corresponding Lagrange multiplier $\alpha_c$.
       3) Update the gradient and update direction $\mathbf{g},\ \mathbf{d}$;
       Endwhile
```

The newly-developed training algorithm is based on iterative computation. In each step, one Lagrange multiplier will be updated and the corresponding sub-problem is solved. The computational complexity is quadratic with regard to the training size N. In contrast, the unified ELM obtains the solution through matrix inversion, and thus scales between quadratically and cubically with respect to N. Consequently, sparse ELM for classification is expected to be faster than the unified ELM when N grows. Additionally, sparse ELM requires less testing time and storage space for problems of all scales. In summary, sparse ELM is quite promising for growing-scale problems, such as neuroscience, image processing, data compression, etc.

3.3 Experiments

The proposed sparse ELM for classification is extensively investigated in this section and compared with the conventional SVM and the unified ELM on some benchmark datasets. Except for COD RNA, marked with $*$ in the table, all experiments are conducted in MATLAB R2010b running on an Intel i5-2400 3.10 GHz CPU with 8 GB RAM. The dataset of COD RNA requires more memory and is evaluated on a VIZ

Table 3.1: Datasets of Binary Classification

Class	Dataset	# train	# test	# features
Low Dims Small Size	Australian	345	345	14
	Breast Cancer	342	341	10
	Diabetes	384	384	8
	Heart	135	135	13
	Ionosphere	176	175	34
Low Dims Large Size	Mushroom	4062	4062	22
	SVMguide1	3089	4000	4
	Magic	9510	9510	11
	$*$ COD RNA	29768	29767	8
High Dims Small Size	Colon Cancer	31	31	2000
	Colon (Gene Sel)	31	31	60
	Leukemia	38	34	7129
	Leukemia (Gene Sel)	38	34	60
High Dims Large Size	Spambase	2301	2300	57
	Adult	6414	26147	123

server with IBM system x3550 M3, dual quad-core Intel Xeon E5620 2.40 GHz CPU with 24 GB RAM. The SVM implementations are realized with the SVM and Kernel Methods Matlab toolbox [8].

As previously described, sparse ELM and the training algorithm are originally developed for binary classification only. Thus, when encountering multiclass problems, one-against-one (OAO) method is utilized to combine several binary sparse ELMs together. Likewise, SVM also uses the OAO method to handle multiclass problems.

3.3.1 Datasets description

In order to fully explore the properties and performance of sparse ELM for classification, many datasets are used in the experiments, including both binary and multiclass ones. Additionally, the datasets consist of high or low dimensions, and large or small sizes. They are taken from LIBSVM portal, UCI repository, etc. [28, 1, 33, 39, 99]. In total, 15 binary and 8 multiclass datasets are included. Tables 3.1 and 3.2 list the details of all datasets.

Preprocessing steps are performed for each individual feature, which scales the feature

Table 3.2: Datasets of Multiclass Classification

Dataset	# train	# test	# features	Classes
Iris	75	75	4	3
Wine	89	89	13	3
Vowel	528	462	10	11
Segment	1155	1155	19	7
Satimage	4435	2000	36	6
DNA	2000	1186	180	3
SVMguide2	196	195	20	3
USPS	7291	2007	256	10

in $[-1,1]$ linearly. Additionally, the features of the testing data will be scaled based on the factors to scale the training data. For binary problems, the label is either 1 or -1. For multiclass problems, the label is $1,2,\cdots,m$, where m is the number of classes. All experiments are conducted with 20 repetitions to produce stable results. In each repetition, the training and testing datasets are randomly permuted within themselves separately.

The datasets of Colon Cancer and Leukemia originally come from UCI repository. However, the dimensionality is too high, thus difficult to handle. Therefore, the features are selected with minimum-redundancy-maximum-relevance method [86] in order to make the datasets easier to deal with. Respectively, 60 features (genes) are selected from 2000 and 7129 ones.

3.3.2 Influence of the number of hidden nodes L

As described in (3.2), even though ELM has universal approximation capability, the number of hidden nodes L cannot grow infinitely. Therefore, training errors should be allowed. Intuitively speaking, the training errors will be reduced if increasing the number L. Furthermore, overfitting problems have been well solved by minimizing the model capacity and empirical errors altogether guided by the regularization theory.

As depicted in Fig. 3.2, both the training and testing accuracy improve with the increment of L for all values of C. Moreover, training and testing performance keep unchanged after L becomes large enough. Fig. 3.3 shows more results at the value $C = 1$. The relationship between accuracy and L is consistent with our analysis.

We aim to find a fixed L, which is suitable for almost all problems, to reduce the computational requirement in the parameter tuning stage. 5-fold cross-validation method is adopted accordingly. Binary and multiclass problems are handled separately because the complexities of these two generic types are different. For all the datasets considered in this thesis, $L = 200$ is fixed for binary problems and $L = 1000$ for multiclass ones. The effectiveness of these two values, 200 and 1000, is well verified by the great validation accuracy.

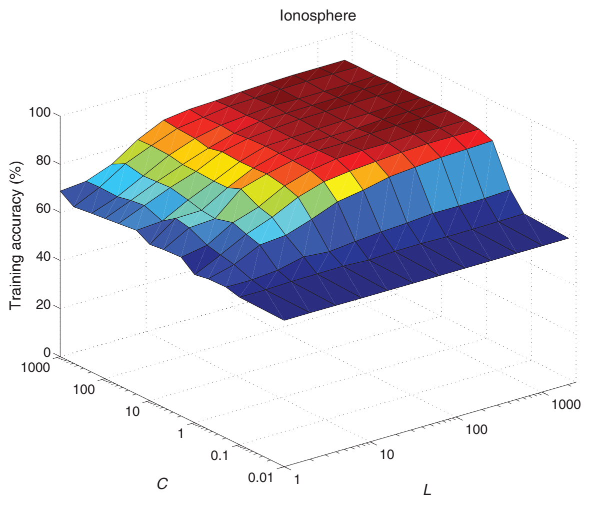
3.3.3 Parameter specifications

The kernel functions adopted are the Gaussian $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ and polynomial $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^d$. The generalization performance of SVM and sparse ELM with Gaussian kernel are shown in Fig. 3.4 and Fig. 3.5 respectively. The figures for the unified ELM is also similar. The combination of trade-off parameter C and kernel parameter σ or d need to be chosen *a priori*. The 5-fold cross-validation method is thus adopted. For C and σ, 14 different values are tried: [0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]. For d, 5 values are tried: [1, 2, 3, 4, 5].

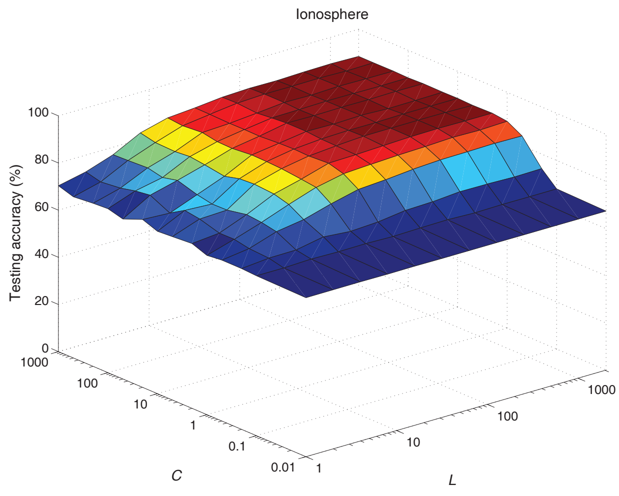
In addition, for non-kernel case of sparse ELM and unified ELM, L is fixed to 200 when dealing with binary problems and 1000 for multiclass ones. And the parameter C is also tried with 14 values: [0.01, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]. The optimal parameters, C and σ or d, for all these methods are specified in Table 3.3.

3.3.4 Performance comparison

The optimal parameters of C and σ or d are fixed once chosen and will be used for training and testing. The results recorded are average accuracy, standard deviation among

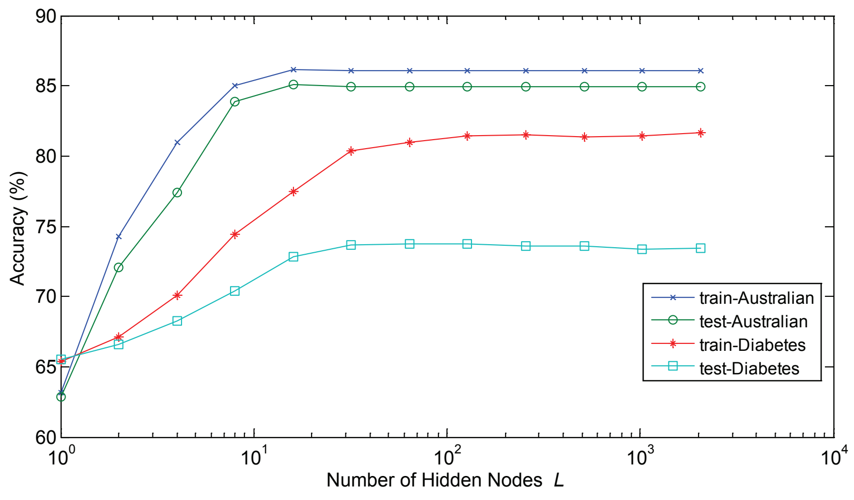


(a) Training accuracy

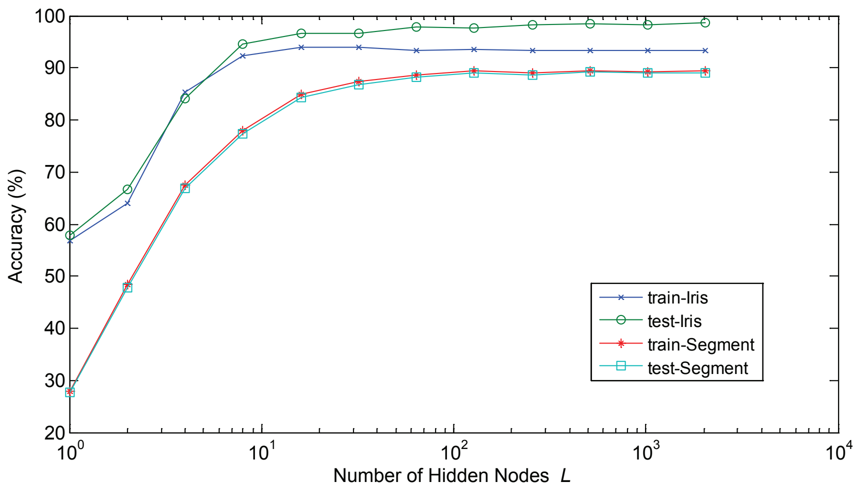


(b) Testing accuracy

Figure 3.2: The performance of sparse ELM (sinusoid nodes) with varying L



(a) Datasets Australian and Diabetes ($C = 1$)



(b) Datasets Iris and Segment ($C = 1$)

Figure 3.3: The performance of sparse ELM (sinusoid nodes) with varying L

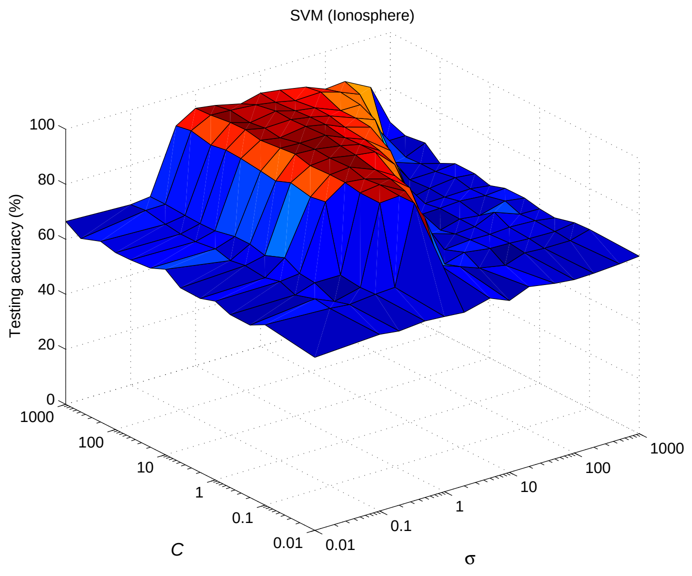


Figure 3.4: SVM (Gaussian kernel) for dataset Ionosphere

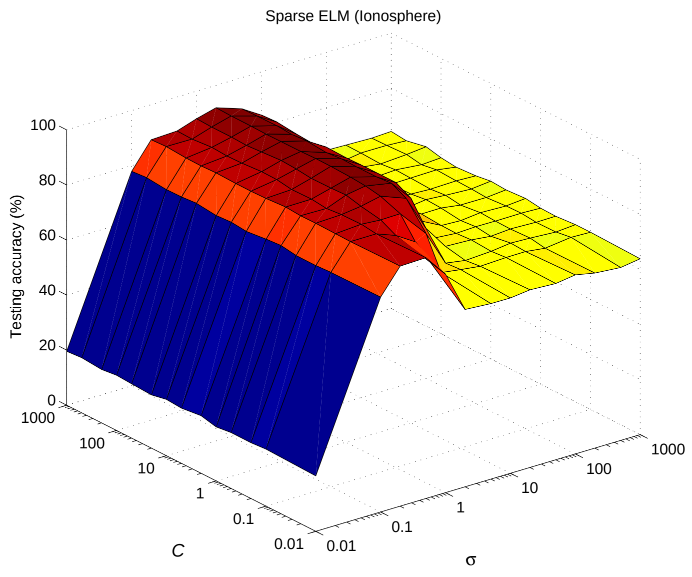


Figure 3.5: Sparse ELM (Gaussian kernel) for dataset Ionosphere

Table 3.3: Parameter Specifications

Dataset	SVM				Unified ELM						Sparse ELM					
	Gaussian Kernel		Polynomial Kernel		Gaussian Kernel		Polynomial Kernel		Sigmoid Nodes	Sinusoid Nodes	Gaussian Kernel		Polynomial Kernel		Sigmoid Nodes	Sinusoid Nodes
	C	σ	C	m	C	σ	C	m	C	C	C	σ	C	m	C	C
Binary Classification																
Australian	1	20	0.1	2	5	2	1	2	10	0.2	200	2	1	3	20	50
Breast Cancer	2	1	1	3	2	1	2	2	100	200	200	1	1	3	100	5
Diabetes	10	5	0.1	2	10	5	1	2	5	200	0.2	0.5	0.2	3	1000	1000
Heart	1	2	10	1	20	10	5	1	2	100	5	5	0.5	1	500	50
Ionosphere	1	2	2	1	1	2	0.01	2	10	1000	1	2	0.01	2	200	5
Mushroom	1	1	1	3	1	1	1	2	20	20	1	1	1	4	5	2
SVMguide1	1	0.5	2	2	50	0.5	0.1	5	100	200	20	0.2	1	5	20	5
Magic	2	1	1	3	200	1	1	4	5	50	50	0.5	1	4	5	10
$*$ COD RNA	2	1	2	3	5	1	1	3	5	5	1	0.5	2	3	50	20
Colon Cancer	1	1	1	1	5	50	0.01	1	5	10	1	20	1	1	10	100
Colon (Gene Sel)	2	2	0.2	1	1	0.1	1	4	5	5	5	2	1	4	0.2	100
Leukemia	50	500	1	1	500	1000	1	1	500	50	1	20	1	1	20	10
Leukemia (Gene Sel)	2	20	1	1	1	1	1	5	2	2	1	1	1	3	10	2
Spambase	5	0.5	1	3	10	1	0.01	3	100	1000	2	0.5	5	5	20	1
Adult	2	2	0.2	2	5	10	1	2	2	5	2	5	1	4	0.2	2
Multiclass Classification																
Iris	10	1	1	3	500	2	10	3	1000	500	1	0.5	2	2	1000	1000
Wine	5	1	1	3	1	2	0.5	1	2	1	5	0.5	10	2	1000	5
Vowel	10	1	10	3	20	0.5	20	4	1000	1000	2	0.2	10	4	200	100
Segment	1000	0.2	1	4	1	0.1	0.1	5	1000	1000	1	0.1	10	4	1000	50
Satimage	500	1	2	3	1	0.2	0.1	4	500	1000	1	0.2	1	3	1000	1000
DNA	500	20	1	3	1	1	1	3	2	200	1	20	10	3	10	10
SVMguide2	5	0.5	1	3	1	0.2	0.01	3	20	1000	50	0.2	1	1	1000	5
USPS	10	10	1	4	1	1	0.01	3	1000	1000	1	1	1	4	1000	1000

20 repetitions, training time and testing time. In the table, the best testing accuracy and shortest training time are highlighted in each row.

Binary problems

(1) *Comparison with SVM*: Tables 3.4 and 3.5 display the results of sparse ELM in kernel case and SVM. It is easily observed that sparse ELM in kernel case provides better generalization performance than SVM for most datasets. Additionally, sparse ELM in non-kernel case (sigmoid and sinusoid random nodes) provide comparable generalization performance with SVM, sometimes better and sometimes worse, as seen in Tables 3.6 and 3.7. Furthermore, sparse ELM, in both kernel and non-kernel cases, are superior to SVM regarding to training speed, accelerating the training phase up to *500 times*. In addition, sparse ELM and SVM both construct compact networks, resulting in similar testing speed.

(2) *Comparison with the Unified ELM*: Tables 3.4-3.7 show comprehensive comparisons between the proposed sparse ELM and the unified ELM. The training speed of sparse ELM is much faster than the unified ELM when dealing with large datasets, while being slower when facing small datasets. Nevertheless, when the dataset is small, training speed is not very important. Furthermore, sparse ELM largely reduces the testing time for almost all the datasets except two cases: Colon (Gene Sel) and Leukemia (Gene Sel) with sigmoid hidden nodes. In these two cases, the number of training data N is extremely small. Hence, sparse ELM only reduces a little computation in the testing phase, even though it does provide a more compact network. Thus, unaccounted random reasons may dominate the computation and lead to this outcome.

Multiclass problems

(1) *Comparison with SVM*: The kernel case of sparse ELM presents better generalization performance for most datasets. On the contrary, the non-kernel case of sparse

ELM is not able to produce better performance than SVM. This is caused by the OAO method. The non-kernel case of sparse ELM is in effect a random method and thus has higher variation than SVM. Therefore, when combining several binary sparse ELMs together by OAO method, the effects of higher variation will be amplified, resulting in the degradation of generalization performance. Additionally, sparse ELM, in both kernel and non-kernel cases, achieves much faster training speed than SVM

(2) *Comparison with the Unified ELM*: As observed from Tables 3.4-3.7, the generalization performance of sparse ELM is on par with the unified ELM. However, sparse ELM is only directly applicable for binary problems, while the unified ELM can handle both binary and multiclass ones straightforwardly. Therefore, the proposed sparse ELM is sub-optimal than the unified ELM when facing multiclass problems. The unified ELM realizes faster training and testing speed for most datasets. Moreover, the variations of training and testing accuracy of sparse ELM are higher than the unified ELM. Therefore, when facing multiclass problems, the unified ELM is a better choice.

Number of support vectors (SVs) and storage space

When facing multiclass problems, SVM and sparse ELM use OAO method to combine several binary classifiers together, while the unified ELM provides the solution directly. Consequently, the number of total vectors are different, making the number of SVs of the unified ELM incommensurable with the other two methods.

Table 3.8 lists the number of SVs for all these methods. The unified ELM produce a dense network so that all vectors are SVs. In contrast, both SVM and the proposed sparse ELM provide sparse networks as only a proportion of vectors are SVs. However, the sparsity varies for SVM and sparse ELM when dealing with different datasets. It is not definite which one provides a more sparse network.

3.4 Conclusions

ELM was originally developed as an improvement for the classic SLFNs and extended to a unified framework for different applications, including regression, classification, clustering, etc. However, the unified ELM produces a dense solution, thus requiring much storage space and testing time. In this chapter, we propose a sparse ELM for classification as an alternative solution, significantly reducing the storage space and testing time. Moreover, it is also demonstrated that the proposed sparse ELM unifies different classification methods, including SVM, SLFNs, RBF networks, etc.

Furthermore, an efficient training algorithm is specifically designed for the proposed sparse ELM for classification. In summary, sparse ELM is favorable over SVM and the unified ELM for: 1) presenting better generalization performance with much faster training speed (up to *500 times*) than SVM; 2) largely reducing the storage space and testing time than the unified ELM. Additionally, when facing large-scale binary problems, sparse ELM is highly recommended for achieving even faster training speed than the unified ELM, which is already exceptionally efficient.

Table 3.4: Performance of sparse ELM, unified ELM and SVM with Gaussian Kernel

Dataset	SVM (Gaussian Kernel)				Unified ELM (Gaussian Kernel)				Sparse ELM (Gaussian Kernel)			
	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)
Binary Classification												
Australian	90.96±0	83.09±0	0.2411	0.0039	93.62±0	84.35±0	0.0098	0.0043	90.61±0.40	**84.62±0.60**	**0.0088**	0.0012
Breast Cancer	98.25±0	97.36±0	0.0550	0.0010	99.12±0	**98.24±0**	0.0092	0.0039	99.05±0.22	98.21±0.13	**0.0075**	0.0009
Diabetes	78.65±0	73.96±0	0.1319	0.0026	83.33±0	74.48±0	**0.0119**	0.0062	84.92±0.18	**74.67±0.16**	0.0164	0.0030
Heart	92.59±0	82.96±0	0.0385	0.0007	84.44±0	**84.44±0**	**0.0033**	0.0011	85.48±0.49	**84.44±0.74**	0.0042	0.0006
Ionosphere	93.75±0	**93.71±0**	0.0667	0.0009	96.02±0	91.43±0	**0.0036**	0.0015	95.45±0	90.31±0.38	0.0063	0.0007
Mushroom	100±0	100±0	41.4878	0.3148	100±0	100±0	2.3835	0.6463	100±0	100±0	**0.8188**	0.0584
SVMguide1	97.09±0	96.90±0	5.1869	0.1154	97.38±0	96.85±0	1.1208	0.4803	97.42±0.07	**97.00±0.06**	**0.4809**	0.0703
Magic	84.29±0	85.73±0	311.7731	2.2336	88.46±0	**86.88±0**	24.0994	4.5080	87.47±0.05	86.20±0.07	**5.1139**	1.4432
∗ COD RNA	95.31±0	**95.25±0**	3858.0860	11.8995	95.33±0	95.22±0	354.8308	51.7553	94.26±0.10	94.44±0.00	**62.7069**	19.0089
Colon Cancer	100±0	70.97±0	0.0494	0.0382	96.77±0	87.10±0	0.0412	0.0395	95.16±1.61	**90.16±2.16**	**0.0357**	0.0316
Colon (Gene Sel)	100±0	**93.55±0**	0.0128	0.0004	100±0	90.32±0	0.0027	0.0008	100±0	**93.55±0**	**0.0020**	0.0006
Leukemia	100±0	**82.35±0**	0.4327	0.4389	100±0	**82.35±0**	0.4134	0.3949	100±0	79.41±0	**0.4093**	0.3856
Leukemia (Gene Sel)	100±0	100±0	0.0114	0.0013	100±0	100±0	0.0017	0.0008	100±0	100±0	**0.0016**	0.0004
Spambase	96.61±0	92.83±0	9.7045	0.1706	95.13±0	**93.70±0**	0.5707	0.2841	95.10±0.12	93.02±0.10	**0.3197**	0.0956
Adult	90.47±0	84.33±0	172.3218	7.2712	85.02±0	**84.66±0**	6.6666	9.7930	85.03±0.11	84.48±0.04	**2.5282**	3.4892
Multiclass Classification												
Iris	100±0	93.33±0	0.0253	0.0009	100±0	**97.33±0**	0.0029	0.0008	98.40±0.53	97.27±1.60	**0.0028**	0.0007
Wine	100±0	97.75±0	0.0304	0.0011	100±0	**98.89±0**	**0.0027**	0.0008	100±0	97.92±0.82	0.0060	0.0013
Vowel	99.81±0	62.55±0	0.6316	0.0310	100±0	57.79±0	**0.0230**	0.0098	100±0	**63.55±1.25**	0.1355	0.0475
Segment	100±0	91.43±0	5.1300	0.3360	100±0	**96.10±0**	**0.2311**	0.0641	100±0	95.77±0.34	0.2357	0.2303
Satimage	100±0	90.55±0	11.5949	0.4161	100±0	**90.95±0**	2.6646	0.3495	99.85±0.02	90.08±0.26	**2.4090**	1.5518
DNA	100±0	**94.10±0**	2.0669	0.1056	100±0	85.24±0	**0.4383**	0.1628	100±0	86.94±0.38	0.5307	0.3038
SVMguide2	100±0	56.41±0	0.1832	0.0031	100±0	**63.08±0**	**0.0028**	0.0022	100±0	**63.08±0**	0.0153	0.0033
USPS	99.88±0	**95.07±0**	20.0226	1.9588	99.99±0	94.97±0	**10.4227**	0.8524	99.99±0	94.82±0.08	10.4365	9.2329

Table 3.5: Performance of sparse ELM, unified ELM and SVM with Polynomial Kernel

Dataset	SVM (Polynomial Kernel)				Unified ELM (Polynomial Kernel)				Sparse ELM (Polynomial Kernel)			
	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)
Binary Classification												
Australian	90.72±0	**84.93±0**	0.0749	0.0003	92.46±0	**84.93±0**	**0.0056**	0.0027	90.46±0.40	84.23±0.78	0.0108	0.0022
Breast Cancer	100	95.31±0	0.0359	0.0015	98.86±0	97.65±0	**0.0050**	0.0016	99.23±0.28	**98.53±0.21**	0.0093	0.0011
Diabetes	83.07±0	74.74±0	0.1180	0.0005	83.59±0	**75.26±0**	**0.0071**	0.0073	81.95±0.73	73.89±0.43	0.0173	0.0066
Heart	87.41±0	82.22±0	0.0348	0.0003	82.96±0	83.70±0	**0.0023**	0.0008	83.85±1.04	**84.00±1.16**	0.0033	0.0003
Ionosphere	94.89±0	89.71±0	0.0410	0.0001	97.73±0	**91.43±0**	**0.0020**	0.0008	92.59±0.99	90.40±0.50	0.0044	0.0005
Mushroom	100±0	100±0	4.5436	0.0760	100±0	100±0	1.3134	0.2049	100±0	100±0	**1.0398**	0.0468
SVMguide1	96.60±0	96.25±0	4.3610	0.0175	97.02±0	**96.63±0**	1.2677	0.7589	96.44±0.09	96.19±0.08	**0.6873**	0.1456
Magic	87.19±0	86.11±0	361.8243	2.6178	87.78±0	**86.42±0**	18.2095	5.3064	86.14±0.12	85.61±0.12	**6.4395**	2.0246
∗ COD RNA	95.22±0	95.00±0	4342.7460	14.4478	95.00±0	**95.01±0**	208.9760	26.0668	94.92±0.03	94.98±0.05	**36.6408**	5.0626
Colon Cancer	100±0	77.42±0	0.0121	0.0003	100±0	80.65±0	0.0039	0.0047	98.48±2.39	**89.84±2.34**	**0.0018**	0.0016
Colon (Gene Sel)	100±0	90.32±0	0.0126	0.0007	100±0	90.32±0	**0.0007**	0.0006	100±0	**93.55±0**	0.0015	0.0006
Leukemia	100±0	85.29±0	0.0121	0.0019	100±0	**88.24±0**	0.0062	0.0052	100±0	83.53±3.40	**0.0029**	0.0021
Leukemia (Gene Sel)	100±0	97.06±0	0.0088	0.0001	100±0	**100±0**	0.0029	0.0017	100±0	**100±0**	**0.0020**	0.0008
Spambase	97.83±0	91.87±0	8.0709	0.1114	94.18±0	**92.39±0**	0.5882	0.3232	88.53±0.17	88.53±0.18	**0.4283**	0.1936
Adult	90.38±0	82.15±0	244.3654	1.6786	90.04±0	82.14±0	5.4775	4.0977	89.14±0.12	**84.31±0.09**	**2.9209**	3.6742
Multiclass Classification												
Iris	100±0	96.00±0	0.0264	0.0009	100±0	97.33±0	0.0076	0.0010	99.00±0.93	**97.47±1.66**	**0.0016**	0.0003
Wine	100±0	97.75±0	0.0332	0.0013	100±0	**98.88±0**	**0.0018**	0.0009	99.44±0.75	97.46±1.89	0.0042	0.0004
Vowel	100±0	59.74±0	0.7054	0.0714	100±0	62.64±0	**0.0526**	0.0202	97.97±0.51	**64.86±3.89**	0.1561	0.1501
Segment	99.83±0	96.45±0	0.4502	0.0792	99.13±0	**96.88±0**	**0.1603**	0.0871	95.90±0.36	88.00±0.28	0.2332	0.0965
Satimage	98.35±0	89.55±0	11.2106	0.4514	95.96±0	89.05±0	4.5560	0.6345	93.96±0.14	**90.96±2.76**	**3.0376**	0.4097
DNA	100±0	94.86±0	25.6512	0.3876	100±0	94.86±0	0.5727	0.2203	99.76±0.02	**95.10±1.52**	**0.5695**	0.2369
SVMguide2	100±0	56.41±0	0.0744	0.0039	94.39±0	56.41±0	**0.0046**	0.0027	94.52±0.69	**58.85±4.00**	0.0091	0.0005
USPS	100±0	95.52±0	27.9954	2.3716	99.99±0	94.92±0	12.2367	0.9945	99.27±0.03	**96.66±5.99**	**9.9507**	1.6330

Table 3.6: Performance of sparse ELM and unified ELM with Sigmoid Hidden Nodes

Dataset	Unified ELM (Sigmoid Hidden Nodes)				Sparse ELM (Sigmoid Hidden Nodes)			
	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)
Binary Classification								
Australian	89.00±0.02	84.62±0.03	**0.0100**	0.0067	87.10±0.94	**85.32±0.72**	0.0181	0.0023
Breast Cancer	97.42±0.01	97.89±0.02	0.0103	0.0073	98.01±0.21	**97.99±0.39**	**0.0085**	0.0029
Diabetes	82.60±0.00	**74.32±0.00**	**0.0118**	0.0076	81.41±0.93	74.11±0.59	0.0148	0.0052
Heart	85.70±0.01	83.63±0.01	**0.0039**	0.0023	85.74±0.90	**83.81±1.18**	0.0079	0.0016
Ionosphere	94.46±0.01	90.63±0.01	**0.0047**	0.0028	92.05±0.80	**90.66±1.26**	0.0073	0.0022
Mushroom	99.91±0	**99.84±0**	2.0256	0.3225	98.61±0.44	98.17±0.51	**0.5291**	0.0912
SVMguide1	94.57±0.01	**94.35±0.01**	0.9222	0.2601	94.33±0.31	94.30±0.48	**0.3313**	0.0955
Magic	82.89±0.02	**82.84±0.02**	11.0930	1.4562	81.48±0.27	81.55±0.30	**3.0090**	0.8074
∗ COD RNA	94.63±0	**94.63±0**	203.9884	9.4621	94.29±0.04	94.36±0.05	**32.0594**	2.2913
Colon Cancer	100±0	83.39±0.06	**0.0085**	0.0037	94.03±3.27	**89.03±4.26**	0.0097	0.0036
Colon (Gene Sel)	100±0	93.06±0.02	0.0024	0.0012	98.98±0.02	**93.55±0**	**0.0015**	0.0015
Leukemia	100±0	76.91±0.05	0.0379	0.0143	98.03±2.18	**78.09±2.37**	**0.0294**	0.0087
Leukemia (Gene Sel)	100±0	98.82±0.01	0.0026	0.0013	100±0	**99.12±1.35**	**0.0025**	0.0015
Spambase	91.29±0.00	**91.18±0.00**	0.5428	0.1239	89.03±1.58	84.78±1.44	**0.2374**	0.0921
Adult	84.46±0.00	**84.29±0**	7.8926	3.1285	83.28±0.58	83.41±0.57	**1.3478**	1.3899
Multiclass Classification								
Iris	98.67±0	97.20±0.00	**0.0045**	0.0046	97.00±1.33	**97.40±0.66**	0.0085	0.0115
Wine	100±0	**99.16±0.01**	**0.0061**	0.0061	100±0	**99.16±0.70**	0.0103	0.0142
Vowel	94.63±0.08	57.85±0.07	**0.0405**	0.0443	96.13±1.67	**59.84±2.46**	0.2709	1.3603
Segment	97.71±0.00	**95.88±0.00**	**0.1809**	0.1505	91.68±0.45	91.57±0.60	0.3961	1.5215
Satimage	92.88±0.00	**89.89±0.00**	3.8516	0.7572	87.86±0.17	85.65±0.22	**2.8562**	4.3452
DNA	98.06±0.00	**93.68±0.01**	0.5864	0.2724	94.45±1.88	88.19±2.55	**0.5002**	0.5810
SVMguide2	92.59±0.01	**54.74±0.15**	**0.0129**	0.0148	84.44±1.09	53.56±5.10	0.0233	0.0359
USPS	99.09±0	93.51±0.00	11.1521	1.2797	98.13±0.08	**93.64±0.15**	**9.0268**	15.0449

Table 3.7: Performance of sparse ELM and unified ELM with Sinusoid Hidden Nodes

Dataset	Unified ELM (Sinusoid Hidden Nodes)				Sparse ELM (Sinusoid Hidden Nodes)			
	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)	Training Accuracy	Testing Accuracy	Training Time (s)	Testing Time (s)
Binary Classification								
Australian	86.84±0.00	**85.84±0.00**	**0.0094**	0.0064	86.42±0.62	85.30±0.62	0.0095	0.0029
Breast Cancer	98.03±0.00	**98.56±0.00**	0.0089	0.0057	98.02±0.23	97.82±0.39	**0.0077**	0.0025
Diabetes	83.48±0.00	74.49±0.00	**0.0105**	0.0068	81.72±0.57	**74.77±0.66**	0.0127	0.0038
Heart	88.07±0.01	83.15±0.01	**0.0038**	0.0019	85.22±1.11	**84.56±1.13**	0.0048	0.0010
Ionosphere	94.91±0.01	88.09±0.01	**0.0036**	0.0025	89.03±0.68	**88.63±1.02**	0.0065	0.0015
Mushroom	99.92±0	**99.88±0**	1.9781	0.3237	97.79±0.27	97.32±0.31	**0.5043**	0.0860
SVMguide1	95.22±0.00	**94.86±0.00**	0.6956	0.2490	94.50±0.17	94.61±0.24	**0.3210**	0.0867
Magic	84.02±0.00	**83.77±0.00**	11.3833	1.4718	82.20±0.13	82.82±0.13	**2.9645**	0.7939
$*$ COD RNA	94.28±0	**94.38±0**	222.0702	9.3984	93.95±0.06	94.01±0.08	**33.0523**	2.7118
Colon Cancer	100±0	82.10±0.06	**0.0084**	0.0031	90.16±2.39	**88.06±4.22**	0.0094	0.0030
Colon (Gene Sel)	99.89±0	91.29±0.03	0.0015	0.0014	98.89±2.90	**93.71±1.24**	**0.0013**	0.0011
Leukemia	100±0	81.03±0.07	0.0345	0.0148	98.03±1.64	**81.47±4.02**	**0.0290**	0.0084
Leukemia (Gene Sel)	100±0	**99.12±0.01**	0.0029	0.0017	100±0	98.82±1.44	**0.0016**	0.0012
Spambase	90.32±0.00	**90.87±0.00**	0.5050	0.1201	87.39±3.39	85.79±2.10	**0.2281**	0.0086
Adult	84.80±0	84.55±0	4.6063	2.9173	84.71±0.16	**84.66±0.07**	**1.3163**	1.2399
Multiclass Classification								
Iris	98.60±0.00	96.20±0.01	**0.0043**	0.0037	97.27±0.29	**97.33±0.42**	0.0057	0.0082
Wine	100±0	**99.10±0.01**	**0.0056**	0.0046	100±0	**99.10±0.84**	0.0084	0.0122
Vowel	97.23±0.00	59.77±0.01	**0.0389**	0.0428	97.40±1.49	**60.74±1.98**	0.2407	1.0784
Segment	96.29±0.00	**95.28±0.00**	**0.1373**	0.1423	91.34±0.46	91.16±0.57	0.4104	1.5421
Satimage	86.09±0.00	83.61±0.00	**2.1227**	0.6790	87.84±0.14	**85.70±0.23**	2.8198	4.4148
DNA	98.11±0.00	**94.57±0.00**	**0.4008**	0.2478	96.64±0.18	94.13±0.42	0.4793	0.5367
SVMguide2	95.58±0.01	**59.77±0.07**	**0.0116**	0.0133	84.21±1.12	59.38±5.56	0.0221	0.0344
USPS	97.97±0.00	**93.56±0.00**	**6.7664**	1.1829	96.76±0.06	92.98±0.11	9.1235	14.9499

Table 3.8: Number of Support Vectors

Dataset	# Total	SVM		Sparse ELM				Unified ELM			
	Vectors	Gaussian Kernel	Polynomial Kernel	Gaussian Kernel	Polynomial Kernel	Sigmoid Nodes	Sinusoid Nodes	Gaussian Kernel	Polynomial Kernel	Sigmoid Nodes	Sinusoid Nodes
Binary Classification											
Australian	345	306	109	240.35	112.85	122.7	119.7	345	345	345	345
Breast Cancer	342	80	41	68	51.3	53.2	52.25	342	342	342	342
Diabetes	384	213	202	278	204.1	219.7	217.55	384	384	384	384
Heart	135	72	46	79.45	58.75	68.8	68.05	135	135	135	135
Ionosphere	176	93	48	98.95	85.65	94.15	101.45	176	176	176	176
Mushroom	4062	956	135	323.15	174.9	582.8	727.75	4062	4062	4062	4062
SVMguide1	3089	429	354	464.35	575.4	910.15	886.75	3089	3089	3089	3089
Magic	9510	3469	3190	3262.1	3599.75	4429.25	4428.6	9510	9510	9510	9510
$*$ COD RNA	29767	5002	3912	11359	5972.3	7578.8	7853.6	29767	29767	29767	29767
Colon Cancer	31	31	24	29.3	25.2	27.6	26.85	31	31	31	31
Colon (Gene Sel)	31	30	17	25.85	24.6	29.35	19.55	31	31	31	31
Leukemia	38	33	32	38	32.9	27.8	27.05	38	38	38	38
Leukemia (Gene Sel)	38	12	7	38	22.45	12.3	11.1	38	38	38	38
Spambase	2301	772	392	810.95	1311.8	1586.2	1590.4	2301	2301	2301	2301
Adult	6414	2729	2261	2531.15	2450.85	2918.45	2666.1	6414	6414	6414	6414
Multiclass Classification											
Iris	150	29	23	54.45	38.4	46.8	47.4				
Wine	178	72	46	149.05	49.9	54.55	53.25				
Vowel	5280	1281	1066	4146.25	1692.3	2487.4	2483.8				
Segment	6930	5010	328	6116.75	866.45	1323.5	1370.05				
Satimage	22175	2376	1528	19197.1	2073.3	2705	2690.2				
DNA	4000	612	2237	3830.95	1967.05	1136.75	1040.85				
SVMguide2	392	386	163	392	175.1	188.95	187.5				
USPS	65619	3179	5404	58532.4	5581.3	6205.7	6573.1				

Chapter 4

Sparse Extreme Learning Machine for Regression

In the last chapter, a sparse ELM is proposed for classification, significantly reducing the storage space, testing time and computational complexity. However, it cannot be applied to regression problems. In this chapter, we propose a sparse ELM for regression and develop a specific training algorithm based on iterative computation. It is favored over the unified ELM for large-scale applications by providing better scalability. Additionally, it is also advantageous over support vector regression (SVR) by producing better generalization performance with much faster learning speed.

4.1 Sparse ELM for Regression

When facing regression problems, the training data provided is: $\mathbf{X}_{N\times d} = \left[\mathbf{x}_1^T \ \cdots \mathbf{x}_N^T\right]^T, \mathbf{x}_i \in \mathbf{R}^{1\times d}$ and $\mathbf{T}_{N\times 1} = [t_1 \ \cdots t_N]^T, t_i \in \mathbf{R}$. Unlike classification, the targets t_i's in regression problems are real numbers rather than binary labels as in (3.3). Therefore, the sparse ELM in Chapter 3 cannot deal with the infinite possible values of t_i's and is not applicable for regression problems. In this section, we develop the sparse ELM for regression.

4.1.1 Problem formulation

When facing regression problems, the target is to find a function $f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ to minimize the expected risks on the probability distribution from which the training and testing data are sampled [93]. However, the distribution is unknown and needs to be estimated with some criteria. According to the regularization theory and structural risk minimization (SRM) [23, 100], the weighted sum of model capacity $||\boldsymbol{\beta}||^2$ and empirical errors $R_{\text{emp}}[f]$ is chosen as the estimate criteria.

$$\begin{aligned} \text{Minimize: } R_{\text{reg}} &= \frac{\lambda}{2}||\boldsymbol{\beta}||^2 + R_{\text{emp}}[f] \\ R_{\text{emp}}[f] &= \frac{1}{N}\sum_{i=1}^{N} c\big(t_i, f(\mathbf{x}_i)\big) \end{aligned} \tag{4.1}$$

where $c\big(t_i, f(\mathbf{x}_i)\big)$ is the the loss function. $c\big(t_i, f(\mathbf{x}_i)\big)$ may have various forms as long as it is convex [27, 93].

In order to derive a sparse solution, we choose the ε-insensitive loss function, similar to ε-insensitive SVR [20, 93]. Errors will be ignored if they are within a ε-wide tube around the target function $f(\mathbf{x})$.

$$c\big(t_i, f(\mathbf{x}_i)\big) = \begin{cases} 0 & \text{for} \quad |t_i - f(\mathbf{x}_i)| < \varepsilon \\ |t_i - f(\mathbf{x}_i)| - \varepsilon & \text{for} \quad \text{otherwise} \end{cases} \tag{4.2}$$

Remark 4.1 *The loss function depends on the problem to be solved. If we select the* $c\big(t_i, f(\mathbf{x}_i)\big) = \big(t - f(\mathbf{x})\big)^2$, *a matrix inversion problem will be generated, leading to a specific case of the unified ELM.*

4.1.2 Optimization

Using the ε-insensitive loss function (4.2) in the problem formulation (4.1), we can obtain the primal problem of sparse ELM for regression:

$$\begin{aligned}
&\text{Minimize: } \mathscr{L}_{p,\text{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\
&\text{Subject to: } t_i - \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} \le \varepsilon + \xi_i \\
&\qquad\qquad \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - t_i \le \varepsilon + \xi_i^* \\
&\qquad\qquad \xi_i^{(*)} \ge 0, \quad i = 1, \dots, N
\end{aligned} \tag{4.3}$$

where $\xi_i^{(*)}$ denotes $\xi_i, \xi_i^* \ge 0$ and $C = 1/(\lambda N)$.

Remark 4.2 *Noted that the output function $f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta}$ is different from SVR: $f(\mathbf{x}) = \phi(\mathbf{x})\boldsymbol{w} + b$. The bias b is removed in ELM because of the universal approximation capability* (3.2). *Therefore, it completely solves the computational inefficiency associated with b and gives sparse ELM for regression distinct merits.*

Following the procedure of standard optimization, the Lagrangian $\mathscr{P}_{\text{ELM}}$ is constructed as follows:

$$\begin{aligned}
\mathscr{P}_{\text{ELM}} = &\frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}\alpha_i\Big(\varepsilon + \xi_i - t_i + \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}\Big) \\
&- \sum_{i=1}^{N}\alpha_i^*\Big(\varepsilon + \xi_i^* + t_i - \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta}\Big) - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\mu_i^*\xi_i^* \\
\alpha^{(*)}, \mu_i^{(*)} \ge 0 &
\end{aligned} \tag{4.4}$$

where $\alpha_i^{(*)}$denotes α_i, α_i^* and $\mu_i^{(*)}$ denotes μ_i, μ_i^*.

The optimal solution will be calculated at the saddle point:

$$\begin{aligned}
&\frac{\partial \mathscr{P}_{\text{ELM}}}{\partial \boldsymbol{\beta}} = 0 \Rightarrow \boldsymbol{\beta} = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)\mathbf{h}(\mathbf{x}_i)^T \\
&\frac{\partial \mathscr{P}_{\text{ELM}}}{\partial \xi_i^{(*)}} = 0 \Rightarrow C = \alpha_i^{(*)} + \mu_i^{(*)}
\end{aligned} \tag{4.5}$$

Accordingly, the KKT conditions are obtained:

$$\begin{aligned}
&\alpha_i\big(t_i - f(\mathbf{x}_i) - \varepsilon - \xi_i\big) = 0 \\
&\alpha_i^*\big(f(\mathbf{x}_i) - t_i - \varepsilon - \xi_i^*\big) = 0 \\
&\mu_i^{(*)}\xi_i^{(*)} = 0
\end{aligned} \tag{4.6}$$

Remark 4.3 *The condition $\alpha_i \cdot \alpha_i^* = 0, i = 1, \cdots, N$ is guaranteed to be satisfied.*

For the KKT conditions (4.6), we can derive the following two cases:

(1) If $\alpha_i > 0, t_i - f(\mathbf{x}_i) = \varepsilon + \xi_i \Rightarrow t_i - f(\mathbf{x}_i) \geq \varepsilon$

(2) If $\alpha_i^* > 0, f(\mathbf{x}_i) - t_i = \varepsilon + \xi_i^* \Rightarrow t_i - f(\mathbf{x}_i) \leq -\varepsilon$

It can be easily observed that these two cases are contradicted with each other. Therefore, at least one of α_i and α_i^* should be zero.

4.1.3 QP problem and convexity

The dual form of sparse ELM for regression is derived by putting the saddle point conditions (4.5) into the Lagrangian function (4.4).

Minimize: $\mathscr{L}_{d,\text{ELM}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\Omega_{\text{ELM}i,j} - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)t_i + \varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*)$

Subject to: $\alpha_i \cdot \alpha_i^* = 0$ and $\alpha_i, \alpha_i^* \in [0, C]$

(4.7)

where $\mathbf{\Omega}_{\text{ELM}}$ is called ELM kernel matrix. And it has two cases:

(1) *Non-kernel case*: Please refer to (3.12) and (3.13) for details.

(2) *Kernel case*: (3.18)-(3.20) list several common kernel functions: *Gaussian*, *Laplacian* and *Polynomial*.

For the conciseness of the objective function $\mathscr{L}_{d,\text{ELM}}$, we use λ_i to replace $\alpha_i - \alpha_i^*$ and λ_i is the Lagrange multiplier which needs to be optimized in the training phase. In addition, as explained before, at least one of α_i and α_i^* needs to be zero. Consequently, $|\lambda_i| = \alpha_i + \alpha_i^*$.

$$\begin{aligned}&\text{Minimize: } \mathscr{L}_{d,\text{ELM}} = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j\Omega_{\text{ELM}i,j} - \sum_{i=1}^{N}\lambda_i t_i + \varepsilon\sum_{i=1}^{N}|\lambda_i| \\ &\text{Subject to: } -C \le \lambda_i \le C\end{aligned} \tag{4.8}$$

Straightforwardly, we can obtain the output function of the sparse ELM for regression as follows:

$$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\left(\sum_{i=1}^{N}\lambda_i\mathbf{h}(\mathbf{x}_i)^T\right) = \mathbf{h}(\mathbf{x})\left(\sum_{s=1}^{N_s}\lambda_s\mathbf{h}(\mathbf{x}_s)^T\right) = \sum_{s=1}^{N_s}\lambda_s K(\mathbf{x},\mathbf{x}_s) \tag{4.9}$$

where $\mathbf{x}_s$ is support vector (SV) and N_s is the number of SVs.

Remark 4.4 *The primal problem* (4.3) *involves two constraints and two slack variables* ξ_i, ξ_i^*. *Hence more dual variables* $(\alpha_i, \alpha_i^*, \mu_i, \mu_i^*)$ *exist in* (4.7) *compared with classification. Moreover, the dual form (QP problem) in* (4.8) *includes an additional term* $\varepsilon\sum_{i=1}^{N}|\lambda_i|$, *making it necessary to carefully avoid crossing the value 0, because the partial derivative* $\partial\mathscr{L}_{d,ELM}/\partial\lambda_i$ *is not defined at* $\lambda_i = 0$.

Remark 4.5 *Different from support vector regression (SVR) [100, 93], sparse ELM is freed from the constraint $\sum_{i=1}^{N} \lambda_i = 0$. As a result, sparse ELM for regression searches the optimal values of λ_i's in a wider area than SVR and provides a better solution.*

Theorem 4.1 *The dual problem of sparse ELM for regression* (4.8) *is convex.*

Proof:

$$\begin{aligned} \frac{\partial \mathscr{L}_{d,\mathrm{ELM}}}{\partial \lambda_i} &= \sum_{j=1}^{N} \lambda_j \Omega_{\mathrm{ELM}i,j} - t_i + \varepsilon\big(\mathrm{sign}(\lambda_i)\big) \\ \frac{\partial^2 \mathscr{L}_{d,\mathrm{ELM}}}{\partial \lambda_i^2} &= \Omega_{\mathrm{ELM}i,i} = \nabla^2 \mathscr{L}_{d,\mathrm{ELM}} \end{aligned} \tag{4.10}$$

where $\nabla^2 \mathscr{L}_{d,\mathrm{ELM}} = \mathbf{\Omega}_{\mathrm{ELM}}$ is the Hessian matrix.

(1) *Non-kernel case*:

$$\begin{aligned} &\mathbf{z}^T \left(\nabla^2 \mathscr{L}_{d,\mathrm{ELM}}\right) \mathbf{z} = \mathbf{z}^T \mathbf{H}\mathbf{H}^T \mathbf{z} = \left(\mathbf{H}^T \mathbf{z}\right)^T \mathbf{I}_{L\times L} \left(\mathbf{H}^T \mathbf{z}\right) \geq 0 \\ &\forall \mathbf{z} \in \mathbf{R}^N \end{aligned} \tag{4.11}$$

Obviously, $\nabla^2 \mathscr{L}_{d,\mathrm{ELM}}$ is a positive semi-definite matrix.

(2) *Kernel case*: The kernel function K needs to meet Mercer's conditions, which ensures that $\nabla^2 \mathscr{L}_{d,\mathrm{ELM}}$ is positive semi-definite.

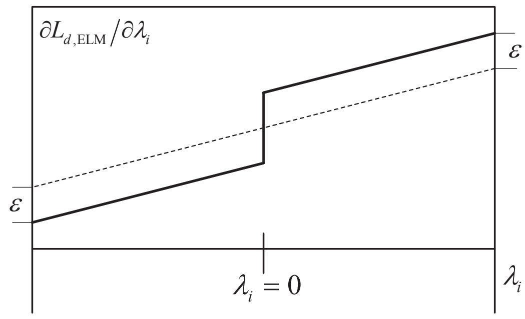


Figure 4.1: The partial derivative of $\mathscr{L}_{d,\mathrm{ELM}}$ with respect to λ_i

The first-order partial derivative $\partial \mathscr{L}_{d,\mathrm{ELM}}/\partial \lambda_i$ is monotonically increasing as the Hessian matrix $\nabla^2 \mathscr{L}_{d,\mathrm{ELM}}$ is positive semi-definite. Fig. 4.1 illustrates the general trend of $\partial \mathscr{L}_{d,\mathrm{ELM}}/\partial \lambda_i$. As observed from Fig. 4.1, the discontinuity at $\lambda_i = 0$ will not influence the monotonicity of $\mathscr{L}_{d,\mathrm{ELM}}$. Consequently, the dual problem of sparse ELM for regression (4.8) is proved to be convex. ■

4.1.4 Sparsity analysis

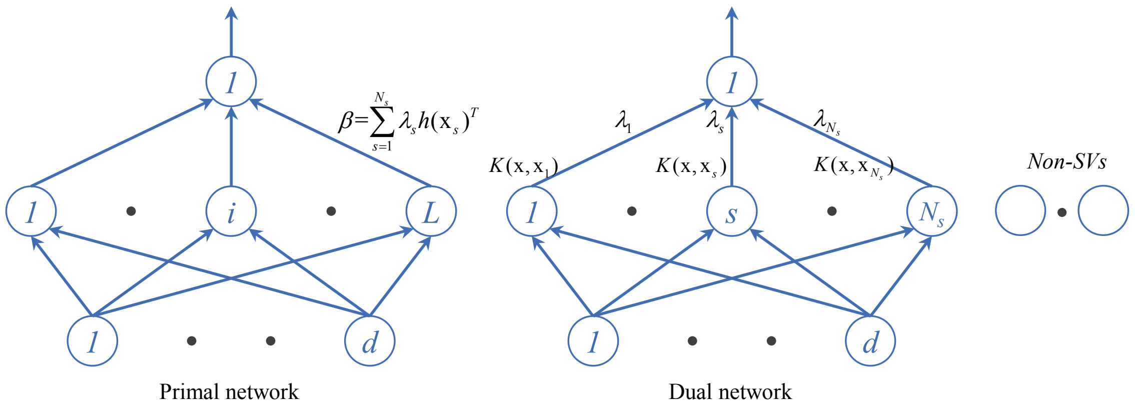


Figure 4.2: The primal and dual networks of sparse ELM for regression

In the case $|t_i - f(\mathbf{x}_i)| < \varepsilon$, α_i and α_i^* both need to be zero in order to fulfill the KKT conditions (4.6). Naturally, $\lambda_i = \alpha_i - \alpha_i^*$ leads to $\lambda = 0$ and the corresponding component will be excluded in (4.9) when calculating the output weight $\boldsymbol{\beta}$. And the data i is not a member of support vectors (SVs).

Fig. 4.2 displays the primal and dual networks of sparse ELM for regression. In the dual network, non-SVs are removed, producing a compact network. In the primal network, the number of hidden nodes L is fixed once chosen. However, the calculation of the output weights $\boldsymbol{\beta}$ (4.9) gets easier as some components are removed. Furthermore, the proposed method largely reduces the storage space and testing time, since they are proportional to the number of SVs.

4.2 Training Algorithm of Sparse ELM for Regression

In this section, an efficient training algorithm based on iterative computation is particularly designed for the proposed sparse ELM for regression. Essentially, the dual form (4.8) is a convex quadratic programming (QP) problem as proved in Theorem 4.1. Akin to the training algorithm of sparse ELM for classification in Chapter 3, the large QP problem is partitioned into a group of small sub-problems, each of which includes only one Lagrange multiplier. Thus, these sub-problems can be solved analytically one-by-one. Additionally, the proposed sparse ELM for regression (4.8) is released from the sum constraint $\sum_{i=1}^{N} \lambda_i = 0$.

4.2.1 Optimality conditions

The optimality conditions are derived based on the KKT conditions (4.6) obtained at the saddle point (4.5). The optimal solution is determined as achieved if the optimality conditions are satisfied; and *vice versa*. The error between the actual output and target is: $e_i = t_i - f(\mathbf{x}_i)$.

(1) $\lambda_i = C$:

$$\begin{gathered} \alpha_i = C, \alpha_i^* = 0 \Rightarrow \mu_i = 0, \xi_i > 0 \\ t_i - f(\mathbf{x}_i) - \varepsilon - \xi_i = 0 \Rightarrow e_i > \varepsilon \end{gathered} \tag{4.12}$$

(2) $0 < \lambda_i < C$:

$$\begin{gathered} \alpha_i \in (0, C), \alpha_i^* = 0 \Rightarrow \mu_i \in (0, C), \xi_i = 0 \\ t_i - f(\mathbf{x}_i) - \varepsilon = 0 \Rightarrow e_i = \varepsilon \end{gathered} \tag{4.13}$$

(3) $\lambda_i = 0$:

$$\begin{aligned} \alpha_i^{(*)} = 0 \Rightarrow \mu_i^{(*)} = C, \xi_i^{(*)} = 0 \\ |t_i - f(\mathbf{x}_i)| < \varepsilon \Rightarrow |e_i| < \varepsilon \end{aligned} \tag{4.14}$$

(4) $-C < \lambda_i < 0$:

$$\begin{aligned} \alpha_i^* \in (0, C), \alpha_i = 0 \Rightarrow \mu_i^* \in (0, C), \xi_i^* = 0 \\ f(\mathbf{x}_i) - t_i - \varepsilon = 0 \Rightarrow e_i = -\varepsilon \end{aligned} \tag{4.15}$$

(5) $\lambda_i = -C$:

$$\begin{aligned} \alpha_i^* = C, \alpha_i = 0 \Rightarrow \mu_i^* = 0, \xi_i^* > 0 \\ f(\mathbf{x}_i) - t_i - \varepsilon - \xi_i^* = 0 \Rightarrow e_i < -\varepsilon \end{aligned} \tag{4.16}$$

4.2.2 Update rule

When the optimal solution has not yet been reached, we need to decide how to proceed further to decrease the objective function $\mathscr{L}_{d,\text{ELM}}$. Assume the Lagrange multiplier chosen to be updated at the current step is λ_c.

$$\begin{aligned} &\mathscr{L}_{d,\text{ELM}} = \varepsilon|\lambda_c| - \lambda_c t_c + \frac{1}{2}\lambda_c^2 K_{cc} + \lambda_c z_c^{\text{old}} + W_{\text{const}} \\ &z_c^{\text{old}} = f_c^{\text{old}} - \lambda_c^{\text{old}} K_{cc} \end{aligned} \tag{4.17}$$

in which W_{const} is a constant term with no relevance to λ_c and the superscript " old" indicates the last step. And K_{cc}, f_c^{old} respectively denotes $K(\mathbf{x}_c, \mathbf{x}_c)$, $f(\mathbf{x}_c)^{\text{old}}$ for conciseness.

$$\frac{\partial \mathscr{L}_{d,\text{ELM}}}{\partial \lambda_c} = \varepsilon\big(\text{sign}(\lambda_c)\big) - t_c + \lambda_c K_{cc} + f_c^{\text{old}} - \lambda_c^{\text{old}} K_{cc} \tag{4.18}$$

Denote the optimal value with $\lambda^\dagger$. Then, the first-order partial derivative of $\mathscr{L}_{d,\text{ELM}}$ at

$\lambda^\dagger$ is 0.

$$\Rightarrow \lambda_c^\dagger = \lambda_c^{\text{old}} + \frac{1}{K_{cc}}\Big(t_c - f_c^{\text{old}} - \varepsilon\big(\text{sign}(\lambda_c^\dagger)\big)\Big) \tag{4.19}$$

The optimal value $\lambda_c^\dagger$ is bounded by the constraints $[-C, C]$. Furthermore, there is a discontinuity for $\partial\mathscr{L}_d/\partial\lambda_i$ at the value 0, as observed from Fig. 4.1, which must not be crossed during the training phase. Thus, we utilize more stringent constraints for $\lambda_c^\dagger$ as follows:

(1) $\lambda_c^{\text{old}} > 0$: the constraint is: $[0, C]$.

(2) $\lambda_c^{\text{old}} < 0$: the constraint is: $[-C, 0]$.

Apparently, only the Lagrange multipliers that violate the optimality conditions will be considered for update. Thus, if index c is selected, it implicitly means that λ_c^{old} violates the conditions.

(1) $\lambda_c^{\text{old}} = 0$:

$$\begin{aligned} \text{If: } t_c - f_c^{\text{old}} \geq \varepsilon \Rightarrow \lambda_c^\dagger \geq 0 \Rightarrow \lambda_c = \left[\lambda_c^\dagger\right]_0^C = \left[\lambda_c^\dagger\right]_{-C}^C \\ \text{Else: } t_c - f_c^{\text{old}} \leq -\varepsilon \Rightarrow \lambda_c^\dagger \leq 0 \Rightarrow \lambda_c = \left[\lambda_c^\dagger\right]_{-C}^0 = \left[\lambda_c^\dagger\right]_{-C}^C \end{aligned} \tag{4.20}$$

Rewrite the update rule for $\lambda_c^{\text{old}} = 0$ concisely:

$$\begin{aligned} &\text{sign}(\lambda_c^\dagger) = \text{sign}(t_c - f_c^{\text{old}}) \\ \Rightarrow &\lambda_c^\dagger = \frac{1}{K_{cc}}\Big(t_c - f_c^{\text{old}} - \varepsilon\big(\text{sign}(t_c - f_c^{\text{old}})\big)\Big) \\ &\lambda_c = \left[\lambda_c^\dagger\right]_{-C}^C \end{aligned} \tag{4.21}$$

(2) $\lambda_c^{\text{old}} > 0$:

$$\begin{aligned} \lambda_c^\dagger &= \lambda_c^{\text{old}} + \frac{1}{K_{cc}}\left(t_c - f_c^{\text{old}} - \varepsilon\right) \\ \lambda_c &= \left[\lambda_c^\dagger\right]_0^C \end{aligned} \tag{4.22}$$

(3) $\lambda_c^{\text{old}} < 0$:

$$\begin{aligned} \lambda_c^\dagger &= \lambda_c^{\text{old}} + \frac{1}{K_{cc}}\left(t_c - f_c^{\text{old}} + \varepsilon\right) \\ \lambda_c &= \left[\lambda_c^\dagger\right]_{-C}^0 \end{aligned} \tag{4.23}$$

4.2.3 Selection criteria

It is vital to choose which Lagrange multiplier to be updated. An intuitive method is to choose the one with the fastest convergence to the minimum:

$$c \in \arg\min_{i=1,\ldots,N}\left(\mathscr{L}_{d,\text{ELM}}(\lambda_i) - \mathscr{L}_{d,\text{ELM}}(\lambda_i^{\text{old}})\right) \tag{4.24}$$

Nevertheless, it is computation intensive to calculate the exact decrease of $\mathscr{L}_{d,\text{ELM}}$ that each Lagrange multiplier will bring. Thus, a more reasonable method is to estimate the decrease of $\mathscr{L}_{d,\text{ELM}}$. Consequently, the violated degree of KKT conditions is naturally used as the estimate and the Lagrange multiplier λ_c with the highest degree of violation will be chosen.

Definition 4.1 *The degree of violation of the KKT conditions is denoted by* $\boldsymbol{d}$. *The positive* d_i *indicates the violation of KKT conditions.*

(1) $\lambda_i = C$:

$$d_i = \varepsilon - e_i \tag{4.25}$$

(2) $0 < \lambda_i < C$:

$$d_i = |e_i - \varepsilon| \tag{4.26}$$

(3) $\lambda_i = 0$:

$$d_i = |e_i| - \varepsilon \tag{4.27}$$

(4) $-C < \lambda_i < 0$:

$$d_i = |e_i + \varepsilon| \tag{4.28}$$

(5) $\lambda_i = -C$:

$$d_i = \varepsilon + e_i \tag{4.29}$$

As a result, the selection criteria is constructed as:

$$c \in \arg \max_{i=1,\ldots,N} d_i \tag{4.30}$$

4.2.4 Termination condition

In real implementation, the optimality conditions only need to be fulfilled within a tolerance γ rather than exactly. Therefore, whenever $\max\limits_{i=1,\ldots,N} d_i < \gamma$ is met, the training algorithm will be terminated. It was discovered that when the tolerance γ is equal to the square root of the machine epsilon, stable results would be generated [26]. Thus, we use $\gamma = 0.001$ accordingly.

4.2.5 Convergence proof

Theorem 4.2 *The training algorithm of sparse ELM for regression will converge to the global optimal solution in a finite number of iterations.*

Proof: Our algorithm satisfies all conditions of Osuna's decomposition theorem [84] as listed below. Consequently, the algorithm is convergent to the global optimum within a finite number of iterations:

(1) The QP problem should be convex: it is proved in Theorem 3.1.

(2) In each step, at least one Lagrange multiplier that violates the KKT conditions should be changed: the Lagrange multiplier chosen λ_c in each step does violate the KKT conditions before the change.

(3) In each step, the objective function should be reduced: the change of λ_c makes it satisfy KKT conditions and reduce the objective function $\mathscr{L}_{d,\text{ELM}}$ definitely.

(4) Lagrange multipliers should be bounded: constraints $[-C,C]^N$ limit the range for all the multipliers.

■

4.2.6 Training algorithm

Algorithm 3 summarizes the training algorithm of sparse ELM for regression.

4.2.7 Merits of sparse ELM for regression

The proposed sparse ELM for regression provides several distinct merits over other regression methods. Conclusively, the proposed method is superior over the unified

Algorithm 3: Sparse ELM for regression

```
Problem formulation: Provided the training data $\mathbf{X} \in \mathbf{R}^{N\times d}, \mathbf{T} \in \mathbf{R}^{N\times 1}$, build
up the dual form as in (4.8);
1: Initialization: $\boldsymbol{\lambda} = \mathbf{0}, \mathbf{e} = \mathbf{T}, \mathbf{d} = \mathbf{0}; \boldsymbol{\lambda}, \mathbf{e}, \mathbf{d} \in \mathbf{R}^N$;
2: While $\max_{i=1,\ldots,N} d_i \geq \gamma$:
     1) Select the Lagrange multiplier $\lambda_c$ with the highest degree of violation;
          $c = \arg\max_{i=1,\ldots,N} d_i$                                   (4.31)
     2) Use the update rules listed in (4.21)-(4.23) to update the chosen multiplier $\lambda_c$;
     3) Use the equations (4.25)-(4.29) to update the violated degree $\mathbf{d}$ and the error $\mathbf{e}$ as
     follow:
          $\mathbf{e} = \mathbf{e} - (\lambda_c - \lambda_c^{\text{old}}) \cdot K(:,c)$                    (4.32)
     where $K(:,c)$ is the $c$-th column of the ELM kernel matrix $\mathbf{\Omega}_{\text{ELM}}$.
   Endwhile
```

ELM and SVR to solve large-scale regression problems for providing a more compact network, better scalability and less memory requirement.

(1) *No bias b and sum constraint $\sum_{i=1}^{N} \lambda_i = 0$*: The bias term b is eliminated due to the universal approximation capability of ELM. On one hand, the computational inefficiency associated with b is resolved completely. On the other hand, the dual problem of sparse ELM for regression (4.8) is freed from the sum constraint $\sum_{i=1}^{N} \lambda_i = 0$ due to the removal of b. Thus, it searches for the optimal solution within a wider area and presents better accuracy.

(2) *Sparse network*: It constructs a sparse network and largely reduces the storage space and testing time compared with the unified ELM.

(3) *No memory issue*: Memory issue is getting more important along with the explosion of the problem scale. For instance, the unified ELM and conventional QP methods [81] need to store the whole kernel matrix $\mathbf{\Omega}_{\text{ELM}}$ in the memory, invoking serious problems. The proposed method is based on iterative computation and only requires to store the values encountered in each step. Thus, it solves the

memory issue completely.

(4) *Computational complexity*: The training algorithm is convergent and the number of iterations is proportional to N. In each iteration, the computational complexity is $\mathrm{O}(N)$. Thus, the complexity of the proposed algorithm is $\mathrm{O}(N^2)$. In contrast, the unified ELM obtains the solution through matrix inversion with complexity between $\mathrm{O}(N^2)$ and $\mathrm{O}(N^3)$ [102, 104]. Consequently, the proposed sparse ELM for regression provides better scalability and realizes faster training speed when solving large-scale problems.

(5) *Easy parallel implementation*: We can easily realize parallel implementation to further accelerate the training algorithm by dividing the whole dataset into several subsets to be handled with several CPU processors separately [9].

4.3 Experiments

The proposed method is extensively investigated in this section for numerous regression datasets. Except for the CASP, marked with $*$, all datasets are evaluated in Matlab R2010b, running on an Intel i5-2400, 3.10 GHz CPU and 8 GB RAM. Experiments on the large dataset, CASP, are conducted in Matlab R2013a, running on an Intel Xeon E5-2650, 2 GHz CPU and 256 GB RAM, otherwise the unified ELM and SVR will be out of memory. SVR is implemented by the SVM and Kernel Methods Matlab Toolbox downloaded from [8]. And through the entire section, all experiments are repeated 20 times in order to get stable and reliable results.

4.3.1 Datasets description

Plenty of datasets are used from the UCI repository [28] and LIBSVM portal [11]. We equally divide each dataset into two parts, one for training and the other for testing. Before the experiments, preprocessing steps are performed over the data. Training data

Table 4.1: The description of datasets

Dataset	# train	# test	# features
Bodyfat	126	126	14
Mpg	196	196	7
Housing	253	253	13
Concrete	515	515	8
Mg	693	692	6
Spacega	1554	1553	6
Abalone	2089	2088	8
Winequality	2449	2449	11
Cpusmall	4096	4096	12
Cadata	10320	10320	8
CASP*	22865	22865	9

Table 4.2: The details within the training phase

Dataset	# of λ_i (samples)	# of iterations	# λ_i changed in the process	Average $\frac{\Delta_i(1)}{\Delta_i}$
Bodyfat	126	134	33	2.01
Concrete	515	2622	217	7.88
Abalone	2089	9177	607	12.60
Cpusmall	4096	3998	225	12.34

are linearly scaled into $[0,1]$ for the targets and $[-1,1]$ for the features. And the factors used to scale the training data are adopted to scale the testing data linearly. All the datasets are detailed in Table 4.1.

4.3.2 Improvement of convergence speed

In this section, we conduct experiments to check the convergence speed of the proposed method. The kernel function and parameters are naively selected as $K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})$ and $C = 1, \sigma = 1$. Fig. 4.3 displays the ratio between the number of iterations and the training size N. It is easily observed that the bar called "original" are much larger than 1 for most datasets. Hence, it is necessary to improve the convergence speed and to further accelerate the learning phase.

Table 4.2 shows the details of the training process of sparse ELM for regression. The four datasets are chosen as examples. And all other datasets present similar process. As seen from the table, only a part of λ_i's are changed during the process, while the

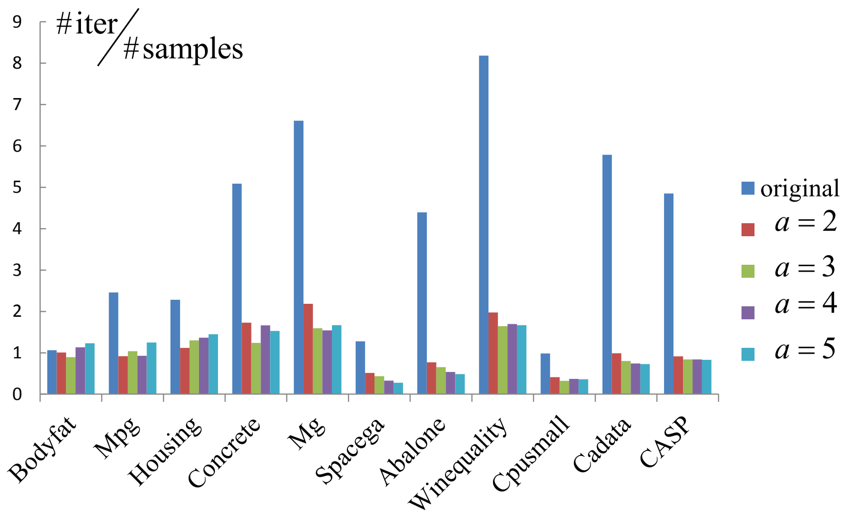


Figure 4.3: Number of iterations for the original and improved algorithms

total number of iterations is much bigger than the number of changed λ_i's. Thus, it is concluded that some λ_i's are updated several times. The first value change and the total change of λ_i are respectively denoted by $\Delta_i(1)$ and Δ_i. It is observed that the ratios $\frac{\Delta_i(1)}{\Delta_i}$ are larger than 1. Thus, a straightforward approach is to add a learning rate $\eta_i = 1 + a \cdot \exp\big(-(\mathrm{TI}_i - 1)\big)$ in the update rules (4.21)-(4.23). Consequently, the training algorithm will change the values of λ_i's by a bigger gap in the first several times so that the number of iterations can be reduced. Which time (1st time, 2nd time or more) the i-th Lagrange multiplier λ_i is updated is denoted by TI_i.

The parameter of a is tested with 4 different values: $[2,3,4,5]$. It is illustrated in Fig. 4.3 that all the 4 values of a greatly improve the convergence speed, while the difference between them is not significant. It is not our main concern to find out the best learning rate in this work. And we choose $a = 3$ for the following experiments: $\eta_i = 1 + 3 \cdot \exp\big(-(\mathrm{TI}_i - 1)\big)$.

4.3.3 Influence of the parameter ε

At here, we study the influence of ε on the sparse ELM for regression. The aim is not to choose the optimal parameters or kernels. Thus, we naively choose the Gaussian kernel with parameters $C = 1, \sigma = 1$.

The number of SVs

We use the toy data "sinc" function to investigate of relationship between ε and number of SVs:

$$t = f(x) = \begin{cases} \sin(x)/x, & x \neq 0 \\ 1, & x = 0 \end{cases} \tag{4.33}$$

in which x is a random variable with uniform distribution in the range $(-10, 10)$. It is respectively sampled 1000 times to generate the training and testing data. Uniform noise in the range (-0.2, 0.2) is added to the targets t of the training data. And the testing data is noise-free.

ε is investigated with 3 values: [0.1, 0.2, 0.3]. Bigger value of ε will lead to fewer support vectors (SVs) as shown in Fig. 4.4.

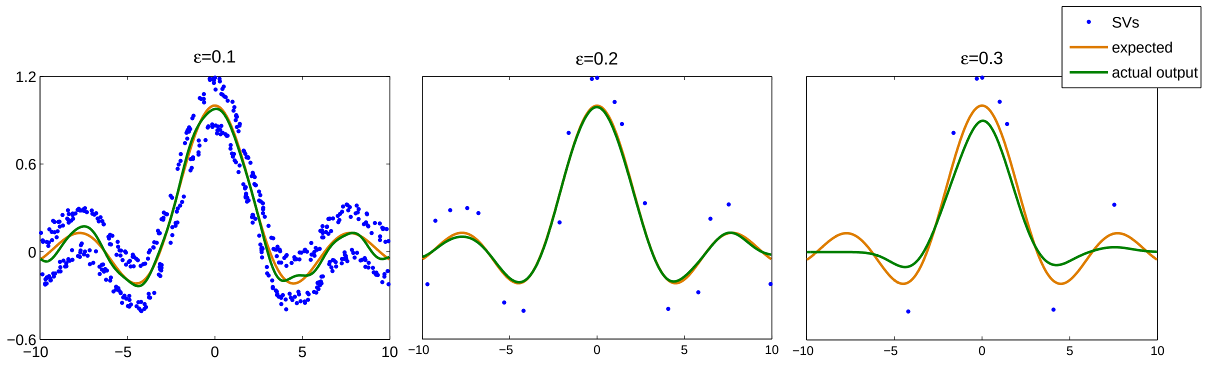


Figure 4.4: The expected and actual outputs and SVs with different ε

Table 4.3: 10-fold cross-validation RMSE with different ε

	$\varepsilon = 0$	$\varepsilon = 0.02$	$\varepsilon = 0.04$	$\varepsilon = 0.06$	$\varepsilon = 0.08$	$\varepsilon = 0.10$	$\varepsilon = 0.12$	$\varepsilon = 0.14$
Bodyfat	**0.0608**	0.0676	0.0747	0.0817	0.0882	0.0962	0.0993	0.1041
Mpg	**0.0476**	0.0504	0.0497	0.0525	0.0570	0.0643	0.0684	0.0734
Housing	0.0612	**0.0569**	0.0607	0.0668	0.0756	0.0829	0.0905	0.0990
Concrete	**0.0898**	0.0904	0.0915	0.0938	0.0967	0.0974	0.1000	0.1038
Mg	0.1442	0.1442	0.1441	**0.1408**	0.1410	0.1409	0.1423	0.1445
Spacega	**0.0320**	0.0331	0.0343	0.0374	0.0422	0.0498	0.0589	0.0627
Abalone	0.0781	0.0784	0.0778	**0.0777**	0.0783	0.0800	0.0824	0.0887
Winequality	0.1240	0.1234	**0.1225**	**0.1225**	0.1238	0.1234	0.1230	0.1240
Cpusmall	0.0313	**0.0307**	0.0325	0.0363	0.0428	0.0530	0.0575	0.0626
Cadata	0.1214	0.1210	0.1205	**0.1202**	0.1203	0.1205	0.1211	0.1224
CASP	0.2348	0.2343	0.2334	0.2320	0.2310	0.2300	0.2293	**0.2287**

10-fold cross-validation error

The optimal relationship between the noise (model and level) and ε has been discussed in [92]. However, when facing real applications, the details of noise are usually unknown, making it impossible to calculated the optimal ε exactly.

Comparing to the unified ELM, ε is an additional parameter, causing bigger computational burden to choose the optimal parameters. Thus, we target to find a value of ε that is good enough, though not optimal, to handle different datasets.

8 values of ε are studied: [0, 0.02, 0.04, 0.06, 0.08, 0.1, 0.12, 0.14]. The 10-fold cross-validation root mean square error (RMSE) is calculated for the training data and summarized in Table 4.3. The cross-validation RMSE at 0, 0.02 and 0.06 are all acceptable, while the values producing the best results are problem dependent. In addition, Fig. 4.4 illustrates that more compact network will be provided by bigger ε. In summary, we choose $\varepsilon = 0.06$ and keep it fixed for the following experiments.

4.3.4 Influence of L

It has been discussed in Chapter 3 that bigger L will lead to better generalization performance [3]. At here, the relationship between the number L and the performance is studied for sparse ELM for regression. *Sigmoid* activation function is utilized:

$$G(\mathbf{a}, b, \mathbf{x}) = \frac{1}{1 + \exp\big(-(\mathbf{a} \cdot \mathbf{x} + b)\big)} \tag{4.34}$$

C is fixed to 1 and the number L is tested with 11 values: $[2^1, 2^2, ..., 2^{11}]$. Fig. 4.5 plots the 10-fold cross-validation RMSE and number L. All other datasets present similar relationships.

It is clear that the 10-fold cross-validation RMSE decreases along with the increase of L. In addition, more hidden nodes should be required when the dataset gets larger and more complex [46, 101]. Thus, we fix the L to a large enough value $L = 2^9$ for the subsequent experiments.

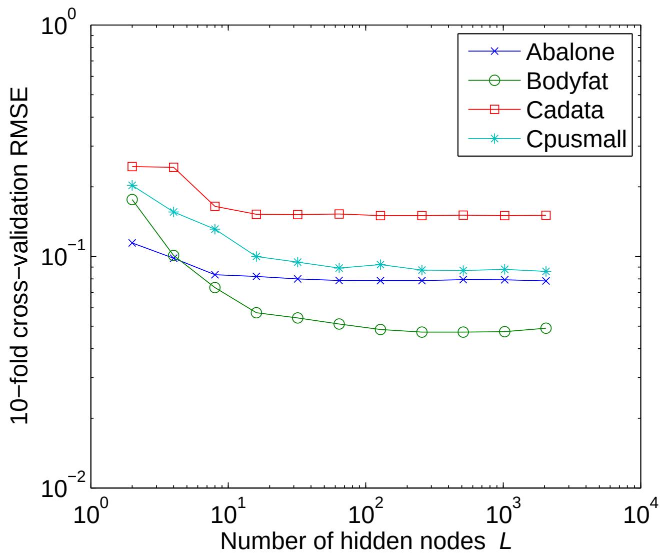


Figure 4.5: The relationship between 10-fold cross-validation RMSE and L

4.3.5 Parameter specification

We evaluate the performance with linear kernel $K(\mathbf{u}, \mathbf{v}) = \mathbf{u} \cdot \mathbf{v}$ and Gaussian kernel $K(\mathbf{u}, \mathbf{v}) = \exp(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2})$. The parameters C and σ are both tried with 20 different values: $[2^{-9}, 2^{-8}, \cdots, 2^{10}]$. And $\varepsilon = 0.06$ is fixed for sparse ELM and SVR.

Table 4.4: Parameter Specifications

Datasets	Sparse ELM				Unified ELM				SVR		
	Linear Kernel	Gaussian Kernel		Sigmoid Nodes	Linear Kernel	Gaussian Kernel		Sigmoid Nodes	Linear Kernel	Gaussian Kernel	
	C	C	σ	C	C	σ	C	C	C	C	C
Bodyfat	2^0	2^1	2^2	2^2	2^6	2^{10}	2^4	2^6	2^2	2^0	2^2
Mpg	2^2	2^5	2^1	2^3	2^7	2^{10}	2^2	2^{10}	2^0	2^5	2^2
Housing	2^0	2^2	2^1	2^0	2^4	2^{10}	2^5	2^4	2^{-2}	2^3	2^3
Concrete	2^3	2^1	2^0	2^2	2^1	2^7	2^0	2^4	2^{-2}	2^1	2^{-1}
Mg	2^0	2^0	2^0	2^3	2^1	2^3	2^0	2^{10}	2^6	2^1	2^0
Spacega	2^3	2^1	2^1	2^1	2^9	2^9	2^3	2^6	2^3	2^2	2^2
Abalone	2^0	2^0	2^0	2^0	2^3	2^8	2^1	2^{10}	2^2	2^2	2^0
Winequality	2^{-5}	2^{-2}	2^1	2^{-2}	2^6	2^1	2^0	2^5	2^1	2^1	2^2
Cpusmall	2^{-4}	2^2	2^0	2^1	2^3	2^8	2^0	2^{10}	2^{-2}	2^{-1}	2^0
Cadata	2^0	2^0	2^0	2^1	2^5	2^1	2^0	2^6	2^1	2^1	2^0
CASP	2^0	2^0	2^0	2^0	2^2	2^0	2^{-2}	2^{10}	2^0	2^0	2^0

For the non-kernel case, the proposed sparse ELM and the unified ELM both adopt the *sigmoid* activation function, where C is also chosen from the 20 values: $[2^{-9}, 2^{-8}, \cdots, 2^{10}]$ and $L = 2^9$ is fixed. The optimal parameters are selected based the 10-fold cross-validation error and summarized in Table 4.4.

4.3.6 Performance comparison

The three methods, the proposed one, unified ELM and SVR, are thoroughly evaluated and compared. The average RMSE is calculated over 20 repetitions. The implicit assumption is that the mean RMSE over 20 repetitions is a reliable estimate of the performance on this particular dataset. The details of all methods are shown in Tables 4.7-4.9. And the shortest training time and best testing RMSE are highlighted in each row.

Generalization performance

The generalization performance of the proposed sparse ELM is separately compared with the unified ELM and SVR. Based on [19], Wilcoxon signed ranks test is suitable for the statistical comparison between two methods on multiple datasets. And Table 4.5 demonstrates the detailed Wilcoxon test between sparse ELM and SVR with Gaussian

Table 4.5: Wilcoxon test between sparse ELM and SVR with Gaussian kernel

Dataset	Difference of testing RMSE (d_i)	Absolute value of difference ($\|d_i\|$)	Rank
Bodyfat	0.0014	0.0014	5
Mpg	0.0017	0.0017	6
Housing	0.0225	0.0225	9
Concrete	0.0268	0.0268	10
Mg	0.0006	0.0006	2
Spacega	0.0037	0.0037	8
Abalone	0.0004	0.0004	1
Winequality	0.0008	0.0008	4
Cpusmall	-0.0024	0.0024	7
Cadata	-0.0006	0.0006	3
CASP	0.0328	0.0328	11

kernel. $|d_i|$ is the absolute value of the difference between the testing RMSE on the two methods.

$$\begin{aligned}
R^+ &= \sum_{d_i>0} \text{rank} + \frac{1}{2}\sum_{d_i=0} \text{rank} = 56 \\
R^- &= \sum_{d_i<0} \text{rank} + \frac{1}{2}\sum_{d_i=0} \text{rank} = 10 \\
T &= \min(R^+, R^-) = 10 \\
z &= \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} = \frac{10 - \frac{1}{4}\cdot 11 \cdot 12}{\sqrt{\frac{1}{24}\cdot 11\cdot 12\cdot 23}} \\
&= -2.0449 < -1.96 = z_{\frac{0.05}{2}}
\end{aligned} \tag{4.35}$$

It is conclude that the proposed sparse ELM produces better generalization performance than SVR at a significance level $\alpha = 0.05$ when adopting the Gaussian kernel because of $z < -1.96$. The same test is performed to compare the proposed sparse ELM and SVR with linear kernel: $z = -1.9560 > -1.96$. Even though we cannot claim the difference to be statistically significant, it is highly probable that sparse ELM outperforms SVR with linear kernel.

Moreover, the Wilcoxon test is also performed to compare sparse ELM and the unified

ELM, where the z values are all bigger than -1.96: 1)-0.9780 for linear kernel; 2) -0.5335 for Gaussian kernel; 3) -1.6893 for non-kernel case with *sigmoid* activation function.

In conclusion, sparse ELM is highly favored than than SVR and on par with the unified ELM in terms of the generalization performance.

Training and testing speed

Consistent with our theoretical analysis of the computational complexity aforementioned, the proposed method achieves much faster training speed than the unified ELM when facing large-scale datasets. Additionally, the proposed method reduces the testing time compared with the unified ELM for datasets of all scales.

Compared with SVR, much faster training and testing speed are provided by the proposed method as seen from Tables 4.8 and 4.9.

Number of SVs

It is verified that the proposed method provides a sparse network, as listed in Table 4.6. Consequently, the storage space is reduced than the unified ELM because the storage requirement is proportional to the number of SVs. Furthermore, the testing speed is also accelerated as fewer computations need to be performed in the testing phase.

Nevertheless, both the proposed method and SVR construct a sparse network, while it depends on the individual dataset to determine which one is more compact.

Table 4.6: Number of support vectors

Dataset	# Total vectors	Sparse ELM			Unified ELM			SVR	
		Linear kernel	Gaussian kernel	Sigmoid Nodes	Linear kernel	Gaussian kernel	Sigmoid Nodes	Linear kernel	Gaussian kernel
Bodyfat	126	19	15	13.65	126	126	126	16	20
Mpg	196	47	38	35.30	196	196	196	58	47
Housing	253	89	69	83.45	253	253	253	95	90
Concrete	515	313	212	278.25	515	515	515	335	233
Mg	693	497	430	429.55	693	693	693	500	418
Spacega	1554	93	112	152.60	1554	1554	1554	181	174
Abalone	2089	642	659	631.40	2089	2089	2089	703	671
Winequality	2449	1601	1572	1585.55	2449	2449	2449	1574	1602
Cpusmall	4096	858	215	745.80	4096	4096	4096	745	280
Cadata	10320	5684	4798	5305.40	10320	10320	10320	6041	4879
CASP	22865	18232	16442	17892.10	22865	22865	22865	18609	16644

4.4 Conclusions

In this chapter, we extend the sparse ELM for regression problems and construct a comprehensive framework of sparse ELM. Compared with the unified ELM, which is state-of-the-art method, it produces a sparse network and largely reduces the storage space and testing time. Subsequently, we design an efficient training algorithm based on iterative computation. Several distinct merits make it superior to other methods: 1) no sum constraint $\sum_{i=1}^{N} \lambda_i = 0$ and bias b, resolving the computational inefficiency associated completely; 2) no memory issue as it only needs to store the values used in each step; 3) computational complexity of lower magnitude than the unified ELM; 4) building up a sparse network and thus reducing the storage space and testing time.

Conclusively, the proposed method is more suitable for large-scale regression problems, such as neuroscience, image processing, time series prediction, etc. Later, we plan to implement the sparse ELM in parallel manner and to use kernel cache [11] to further improve the efficiency of sparse ELM.

Table 4.7: Performance of sparse ELM and unified ELM with Sigmoid hidden nodes

Dataset	Sparse ELM				Unified ELM			
	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)
Bodyfat	0.0524	0.0448	0.0105	0.0032	0.0331	**0.0198**	**0.0067**	0.0047
Mpg	0.0603	0.1389	0.0583	0.0030	0.0439	**0.1231**	**0.0086**	0.0085
Housing	0.0835	**0.1367**	0.0166	0.0081	0.0717	0.1436	**0.0090**	0.0091
Concrete	0.1224	**0.1374**	0.1630	0.0188	0.1243	0.1407	**0.0534**	0.0465
Mg	0.1555	0.1450	0.5045	0.0286	0.1359	**0.1330**	**0.0400**	0.0370
Spacega	0.0474	0.0450	0.3166	0.0369	0.0374	**0.0401**	**0.1823**	0.1246
Abalone	0.0819	0.0785	**0.3397**	0.0920	0.0756	**0.0741**	0.4459	0.2208
Winequality	0.1318	0.1221	**0.5067**	0.2089	0.1267	**0.1210**	0.6475	0.3216
Cpusmall	0.0345	**0.0373**	**1.0375**	0.2389	0.0333	0.0376	2.1991	0.8132
Cadata	0.1409	**0.1497**	**13.4122**	1.9611	0.1322	0.1498	21.3416	4.0056
CASP	0.2467	0.2495	**59.3273**	6.7365	0.2314	0.2347	70.7220	9.4233

Table 4.8: Performance of sparse ELM, unified ELM and SVR with linear kernel

Dataset	Sparse ELM				Unified ELM				SVR			
	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)
Bodyfat	0.0534	0.0436	0.0066	0.0001	0.0336	**0.0161**	**0.0051**	0.0010	0.0402	0.0368	0.0268	0.0002
Mpg	0.0660	**0.1341**	0.0424	0.0004	0.0602	0.1500	**0.0039**	0.0014	0.0604	0.1452	0.0562	0.0007
Housing	0.0743	**0.1455**	0.0193	0.0004	0.0716	0.1507	**0.0073**	0.0012	0.0723	0.1543	0.0832	0.0008
Concrete	0.1485	**0.1272**	0.3420	0.0012	0.1457	0.1402	**0.0217**	0.0025	0.1462	0.1294	0.7283	0.0034
Mg	0.1674	**0.1570**	0.0840	0.0029	0.1666	0.1582	**0.0259**	0.0044	0.1668	0.1572	1.8135	0.0070
Spacega	0.0477	**0.0406**	0.3424	0.0026	0.0395	0.0435	**0.1390**	0.0291	0.0398	0.0436	4.5092	0.0205
Abalone	0.0802	**0.0771**	**0.3049**	0.0131	0.0806	0.0774	0.3171	0.0502	0.0802	0.0776	19.6114	0.0447
Winequality	0.1345	0.1273	**0.3792**	0.0368	0.1294	**0.1243**	0.4751	0.0562	0.1297	0.1266	46.9333	0.0952
Cpusmall	0.1147	0.1001	**0.5004**	0.0356	0.1036	**0.0961**	2.1691	0.1694	0.1212	0.1009	75.7302	0.0657
Cadata	0.1437	**0.1516**	**10.5743**	0.4547	0.1423	0.1536	15.1593	0.8153	0.1436	0.1537	3548.7487	1.3921
CASP	0.2390	**0.2413**	**102.9173**	3.5126	0.2457	0.2483	116.4548	4.7878	0.2500	0.2529	14400.5332	22.9362

Table 4.9: Performance of sparse ELM, unified ELM and SVR with Gaussian kernel

Dataset	Sparse ELM				Unified ELM				SVR			
	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)	Training RMSE	Testing RMSE	Training time (s)	Testing time (s)
Bodyfat	0.0516	0.0425	0.0066	0.0001	0.0333	**0.0162**	**0.0034**	0.0009	0.0522	0.0439	0.0224	0.0003
Mpg	0.0504	**0.1211**	0.0193	0.0004	0.0405	0.1216	**0.0042**	0.0012	0.0464	0.1228	0.0471	0.0007
Housing	0.0510	**0.1284**	0.0480	0.0008	0.0715	0.1496	**0.0061**	0.0033	0.0700	0.1509	0.0647	0.0012
Concrete	0.0671	**0.1241**	**0.0742**	0.0027	0.0503	0.1394	0.0827	0.0099	0.0583	0.1509	0.7481	0.0058
Mg	0.1307	0.1341	0.0987	0.0093	0.1272	**0.1319**	**0.0359**	0.0166	0.1279	0.1347	1.7136	0.0143
Spacega	0.0412	**0.0385**	**0.1499**	0.0083	0.0358	0.0401	0.1905	0.0880	0.0414	0.0422	4.3292	0.0307
Abalone	0.0753	0.0738	0.3580	0.0460	0.0738	**0.0735**	**0.3428**	0.1477	0.0736	0.0742	18.6725	0.1281
Winequality	0.1262	**0.1200**	0.5233	0.1328	0.1177	0.1204	**0.4816**	0.2035	0.1274	0.1208	45.1656	0.3300
Cpusmall	0.0317	0.0407	**1.2119**	0.0382	0.0248	**0.0352**	1.6629	0.5505	0.0348	0.0383	61.2297	0.1354
Cadata	0.1197	0.1423	**8.9124**	1.5977	0.1203	0.1433	15.3376	3.2928	0.1171	**0.1417**	2813.5930	3.0710
CASP	0.1874	**0.2026**	**51.6688**	15.2476	0.1935	0.2046	121.2519	24.9627	0.2307	0.2354	12269.6436	41.0614

Chapter 5

Local Receptive Fields Based Extreme Learning Machine

Extreme learning machine (ELM) in full connections produces superior generalization performance with high efficiency. However, when facing applications like image analysis, speech recognition, natural language processing, etc., fully connected network may not present satisfactory performance because it does not model the local correlations in these tasks. In this chapter, we propose the local receptive fields based extreme learning machine (ELM-LRF), where the connections between input and hidden layer are sparse and locally bounded. In addition, we implement the ELM-LRF network with convolutional hidden nodes and evaluate the performance on the benchmark object recognition dataset, NORB. The proposed method produces the *best accuracy* and accelerates the learning speed up to *200 times*.

5.1 Introduction

In the ELM implementations studied in the past, hidden nodes are in full connections with the input ones. On one hand, these implementations provide excellent accuracy with fast speed for a wide types of applications. On the other hand, when facing applications with strong local correlations, such as image analysis, speech recognition, etc., it may be more reasonable to construct a network with sparse, local connections rather than full ones. In fact, bioscience has discovered that local receptive field is the key factor for the brain to deal with visual information. Naturally, we raise an open question for ELM: *can local receptive fields be implemented in ELM?* In this chapter, we discuss this question in detail.

It has been proved that ELM can generate the hidden nodes randomly based on any continuous probability distribution in [46, 44, 45]. Therefore, we can perform a straightforward extension to generate the hidden nodes based on some probability distribution, which are denser around the center while sparser farther away, to deal with these locally correlated applications. Different shapes of local receptive fields may be used as long as they are continuous and local-wise. For instance, McDonnell *et al.* utilize random sampling method to generate the local receptive fields and present superior performance on the MNIST, NORB and SVHN datasets [73, 74]. The selection of the shapes relies on the specific problem to be dealt with. Inspired by the convolutional neural networks (CNNs), we implement the local receptive fields by randomly generating convolutional hidden nodes in this chapter.

5.2 Local Receptive Fields Based Extreme Learning Machine

In this section, we describe the details of ELM-LRF and apply it as a generic architecture to solve image processing and similar tasks, where different density levels of connections can be used. The sparse connections between input and hidden nodes decide the local receptive fields and could be sampled by any continuous probability distribution. Subsequently, combinatorial nodes are followed, grouping several hidden nodes together into sub-network and providing translational invariance to the network. The training speed is exceptionally fast as no gradient descent steps are performed.

5.2.1 Hidden nodes in full and local connections

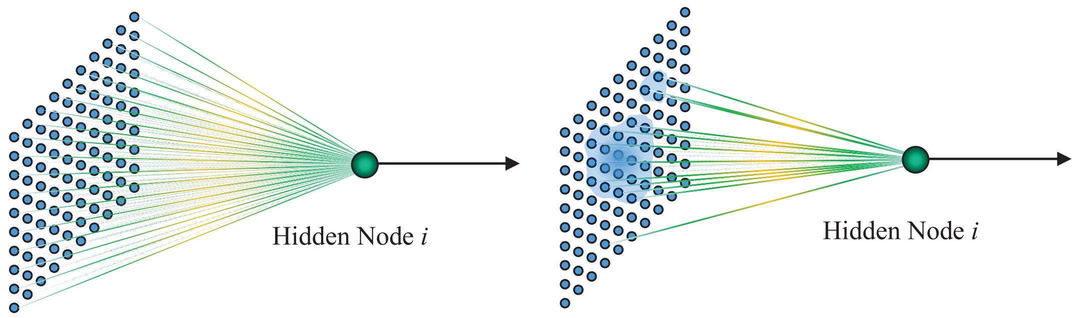
ELM theories demonstrate that hidden nodes can be constructed randomly based on any probability distribution. In effect, two types of randomness are involved:

(1) *Random connections*: The connections between input and hidden nodes can be sampled randomly according to diverse probability distributions.

(2) *Random weights*: The weights between input and hidden nodes can also be generated randomly.

Hidden nodes in full connections, as depicted in Fig. 5.1(a), have been thoroughly investigated and produce state-of-the-art performance in numerous applications, such as bio-medical applications [108], text analysis [111], power systems [80], remote-sensing image classification [85], etc. In essence, these works concentrate on the aspect of random weights, while giving no attention to the random connections.

However, for image analysis, speech recognition, natural language processing, etc. [63], the strong local correlations within the input degenerate the effectiveness of full connections. In order to solve these problems, we naturally construct the hidden nodes in local connections. As shown in Fig. 5.1(b), the connections between input and hidden nodes are denser around the center while sparser farther away. The local receptive fields have been verified by concrete biological evidence, which shows that different cells in the visual cortex are sensitive to different sub-regions of the retina (input layer) [4, 57, 94]. Consequently, locally connected network is suitable for image processing and similar works because more meaningful representations are produced in the hidden layer by explicitly modeling the local structures.



(a) Hidden nodes in full connections (b) Hidden nodes in local connections

Figure 5.1: Hidden nodes in full and local connections

5.2.2 Combinatorial nodes

In biological systems, it is unclear what forms of local receptive fields are adopted. And it is probable to generate them by diversified methods. One promising method is to construct the combinatorial node as proposed in [44]. It proves that hidden node in ELM could be a sub-network (combination) of several hidden nodes. Fig. 5.2 shows an example where the combinatorial node i is actually a sub-network of hidden nodes that are in sparse connections with the input ones bounded by several local areas. The combinatorial node i connects with the three hidden nodes and performs a weighted summation, linear or nonlinear, over them.

In this manner, the feature generated at one location (obtained by one particular combinatorial node) is likely to be suitable at different locations (different combinatorial nodes). As a consequence, the ELM-LRF network will be invariant to translations and rotations of the input. Additionally, the connections between the input and combinatorial nodes are able to model the local structures even better. The overlaps of the three receptive fields make the connections denser around the center while sparser farther away. Subsequently, we use average methods to calculate the values of these combinatorial nodes. Of course, it deserves further investigation to find out other suitable methods, such as max-pooling, learning based methods, etc.

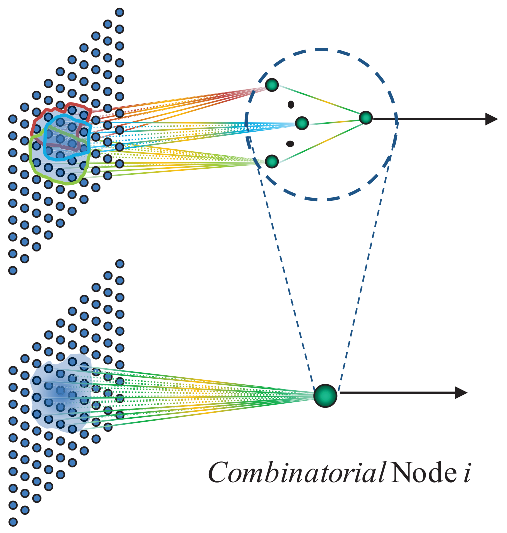


Figure 5.2: The combinatorial node *i*

5.3 The Implementation

5.3.1 One feasible network of ELM-LRF

Even though the local receptive fields can be sampled by various types of probability distribution, we utilize the simple step function to construct one feasible network of ELM-LRF for convenience. The receptive field of each hidden node is bounded by a pre-determined distance to the center. Moreover, the input weights to different hidden nodes in the same feature map are shared. In this sense, our hidden nodes actually perform convolution operations similar to CNNs. Additionally, the combinatorial nodes are formulated with square/square-root pooling structure.

The network of ELM-LRF that we build up in this chapter is displayed in Fig. 5.3:

(1) *Random convolutional nodes*: The node in the feature maps is one case of the hidden node in local connections depicted in Fig. 5.1(b).

(2) *Combinatorial nodes*: The node in the pooling maps is one case of the combinatorial node depicted in Fig. 5.2.

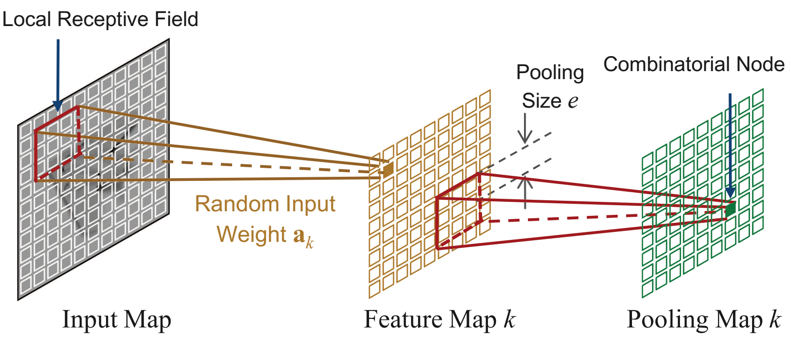


Figure 5.3: One feasible network of ELM-LRF

5.3.2 Random input weights

Fig. 5.4 displays the whole network of ELM-LRF with K maps. K diverse feature maps are generated with K different input weights and provide comprehensive representations of the input. The feature maps are composed of random convolutional nodes, where input weights on the same map are shared while different among different maps. The input weights are first generated randomly and then orthogonalized as follows:

(1) *Random generation*: Generate the initial weight matrix $\hat{\mathbf{A}}^{\text{init}}$ based on the standard Gaussian distribution randomly [1]. Assume that the input is of $d \times d$ and receptive field of $r \times r$, then the feature map should be of $(d-r+1) \times (d-r+1)$.

$$\begin{aligned} &\hat{\mathbf{A}}^{\text{init}} \in \mathbf{R}^{r^2 \times K} \\ &\hat{\mathbf{a}}_k^{\text{init}} \in \mathbf{R}^{r^2},\ k = 1, \cdots, K \end{aligned} \tag{5.1}$$

(2) *Orthogonalization*: Use singular value decomposition (SVD) method to orthogonalize the matrix $\hat{\mathbf{A}}^{\text{init}}$ into $\hat{\mathbf{A}}$. The columns of $\hat{\mathbf{A}}$, $\hat{\mathbf{a}}_k$'s, are the orthonormal basis of $\hat{\mathbf{A}}^{\text{init}}$ [2].

The effect of orthogonalization is to extract more comprehensive and discriminative

[1]We discover that the bias term is not necessary in the convolutional nodes.

[2]Orthogonalization cannot be performed over the column when $r^2 < K$. In this case, 1) $\hat{\mathbf{A}}^{\text{init}}$ is transposed; 2) orthogonalized over the column; 3) transposed back.

features than non-orthogonal ones. Thus, it further improves the generalization performance of the network. The works in [58, 63] have previously utilized orthogonal random weights and present superior performance.

$\hat{\mathbf{a}}_k \in \mathbf{R}^{r^2}$ is transformed into $\mathbf{a}_k \in \mathbf{R}^{r\times r}$ column-wisely. Thus, the convolutional hidden node (i,j) in the k-th feature map, $c_{i,j,k}$, is calculated as:

$$\begin{aligned} c_{i,j,k}(\mathbf{x}) &= \sum_{m=1}^{r}\sum_{n=1}^{r} x_{i+m-1,j+n-1}\cdot a_{m,n,k} \\ & i,j = 1,\cdots,(d-r+1) \end{aligned} \tag{5.2}$$

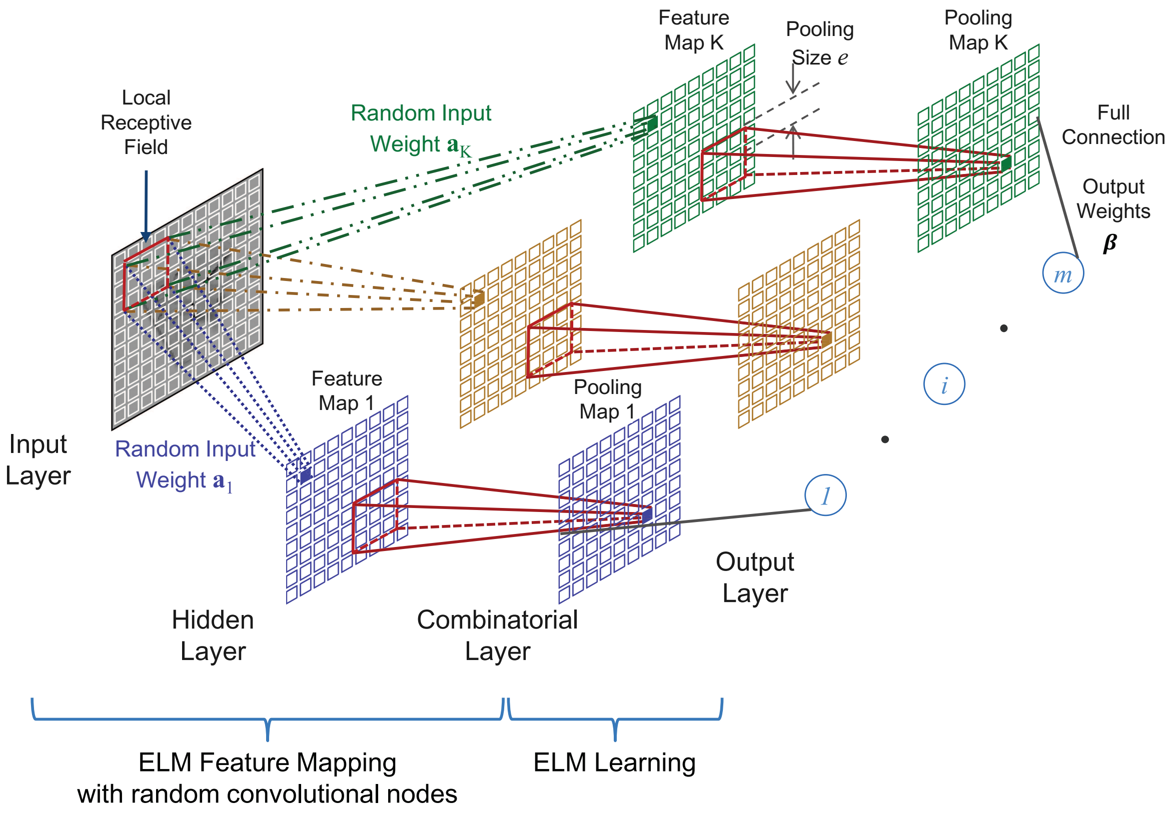


Figure 5.4: The network of ELM-LRF with K maps

Several CNN networks with certain structures also present surprisingly good performance with random, untrained input weights [90, 17, 55]. However, the performance in these works still cannot outperform fine-tuned ones. In contrast, the proposed ELM-LRF in this chapter, where the input weights to the feature maps are orthogonal random, can produce even higher accuracy than well-trained counterparts while achieving much faster learning speed. We will compare ELM-LRF and CNN in depth later.

5.3.3 Square/square-root pooling structure

We form the combinatorial node with square/square-root pooling structure. As shown in Fig. 5.4, pooling size e is the distance between the center and the edge of the pooling area. In addition, the size of the pooling map is $(d-r+1)\times(d-r+1)$, equivalent to the size of the feature map. $c_{i,j,k}$ and $h_{p,q,k}$ denote the convolutional node (i,j) in the feature map k and combinatorial node (p,q) in the pooling map k.

$$\begin{aligned} h_{p,q,k} = & \sqrt{\sum_{i=(p-e)}^{p+e} \sum_{j=(q-e)}^{q+e} c_{i,j,k}^2} \\ & p,q = 1,\cdots,(d-r+1) \\ & \text{if } (i,j) \text{ is out of bound: } c_{i,j,k} = 0 \end{aligned} \tag{5.3}$$

The network is provided with the key factors of successful image processing [14]: rectification nonlinearity and translational invariance, by the square and summation operations respectively. Moreover, it is proved that the square/square-root pooling structure after the convolution operation is frequency selective and translational invariant [90]. Consequently, the network of ELM-LRF implemented in this chapter will be exceptionally suitable for image processing and similar tasks.

5.3.4 Closed-form solution of the output weights

As observed in Fig. 5.4, the output layer is in full connection with the pooling layer. The vector of output weights $\boldsymbol{\beta}$ is calculated analytically and deterministically with regularized least squares method [71].

Suppose we have the input sample $\mathbf{x}$, the combinatorial node $h_{p,q,k}$ could be easily calculated by solving (5.2) and (5.3) sequentially.

$$\begin{aligned} c_{i,j,k}(\mathbf{x}) &= \sum_{m=1}^{r}\sum_{n=1}^{r} x_{i+m-1,j+n-1}\cdot a_{m,n,k} \\ h_{p,q,k} &= \sqrt{\sum_{i=(p-e)}^{p+e}\sum_{j=(q-e)}^{q+e} c_{i,j,k}^2} \\ & p,q = 1,\cdots,(d-r+1) \\ & \text{if } (i,j) \text{ is out of bound: } c_{i,j,k}=0 \end{aligned} \tag{5.4}$$

Putting all combinatorial nodes in the pooling layer into a row vector and concatenating the rows of N input samples together, the combinatorial layer matrix $\mathbf{H}\in\mathbf{R}^{N\times K\cdot(d-r+1)^2}$ is constructed.

(1) if $N \leq K\cdot(d-r+1)^2$

$$\boldsymbol{\beta} = \mathbf{H}^T\left(\frac{\mathbf{I}}{C}+\mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \tag{5.5}$$

(2) if $N > K\cdot(d-r+1)^2$

$$\boldsymbol{\beta} = \left(\frac{\mathbf{I}}{C}+\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \tag{5.6}$$

5.3.5 Algorithm

The algorithm is summarized in Algorithm 4.

Algorithm 4: The algorithm of ELM-LRF

```
1: Random convolutional nodes: Input $d \times d$; receptive field $r \times r$; feature map $(d-r+1) \times (d-r+1)$.
   1) Generate $\hat{\mathbf{A}}^{\text{init}} \in \mathbf{R}^{r^2 \times K}$ randomly based on standard Gaussian distribution.
   2) Orthogonalize $\hat{\mathbf{A}}^{\text{init}}$ with SVD method [32] and get $\hat{\mathbf{A}} \in \mathbf{R}^{r^2 \times K}$. Reshape each column of $\hat{\mathbf{A}}$, $\hat{\mathbf{a}}_k$, into $\mathbf{a}_k \in \mathbf{R}^{r \times r}, k = 1, \cdots, K$.
   3) Generate $K$ feature maps by convolving $\mathbf{a}_k, k = 1, \cdots, K$ with the input.
2: Square/square-root pooling: Pooling size $e$ is the distance between the center and the edge of the pooling area (Fig. 5.4).
   1) Calculate the squares of all nodes in $K$ feature maps.
   2) For each pooling area, sum up the squared values within it.
   3) Calculate the value of each pooling node by performing square-root operation on the corresponding summation.
3: Regularized least squares solution:
   1) Concatenate all nodes in the pooling maps into a row and put all rows of $N$ training samples together. And we obtain the output matrix of the pooling layer $\mathbf{H} \in \mathbf{R}^{N \times K \cdot (d-r+1)^2}$.
   2) Calculate the output weight $\boldsymbol{\beta}$:
```

$$\boldsymbol{\beta} = \begin{cases} \mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T \right)^{-1} \mathbf{T}, & \text{if } N \leq K \cdot (d-r+1)^2 \\ \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H} \right)^{-1} \mathbf{H}^T \mathbf{T}, & \text{if } N > K \cdot (d-r+1)^2 \end{cases} \tag{5.7}$$

5.4 Discussions

5.4.1 Universal approximation and classification capability

The network of ELM-LRF can be considered as a specific type of the general ELM:

(1) *Local receptive fields*: The sparse connections bounded by the local receptive fields can be regarded as ordinary connection with weight 0. Generally the probability distribution used to generate the connections is still piecewise continuous. Therefore, the universal approximation and classification capabilities of ELM are preserved for ELM-LRF, enabling ELM-LRF to learn sophisticated forms of input.

(2) *Combinatorial nodes*: Hidden node in ELM can be a sub-network, consisting of different nodes in linear or nonlinear manners. Thus, the square/square-root pooling structure is adopted to formulate the sub-network directly, introducing translational and rotational invariance.

5.4.2 Relationship with CNN

Apparently, the proposed ELM-LRF is closely related to CNN. They both handle the raw input directly and apply local connections to force the network to model spatial correlations in natural images and languages. Additionally, high-level features are implicitly learned or generated, on which the learning is subsequently performed. However, several distinctions differentiate the proposed ELM-LRF and CNN:

(1) *Local receptive fields*: The local receptive fields provided by ELM-LRF are more flexible, which can be sampled randomly based on different types of probability distributions. However, CNN adopts convolutional hidden nodes only. Except for the random convolutional nodes chosen in this chapter, other types are also applicable

for ELM-LRF, such as the ones generated by random sampling method in [73, 74]. ELM-LRF deserves further exploration for other proper local receptive fields and learning methods to construct the combinatorial nodes.

(2) *Training method*: The training of CNN depends on BP algorithm, while ELM-LRF provides a simple deterministic solution. CNN is degenerated by the trivial issues of BP, such as local minima, slow convergence rate, etc. Moreover, it is computation intensive as BP algorithm involves numerous gradient descent steps. On the contrary, ELM-LRF generates the input weights randomly and calculates the output weights analytically. Only the output weights require computations, making ELM-LRF deterministic and highly efficient.

5.5 Experiments

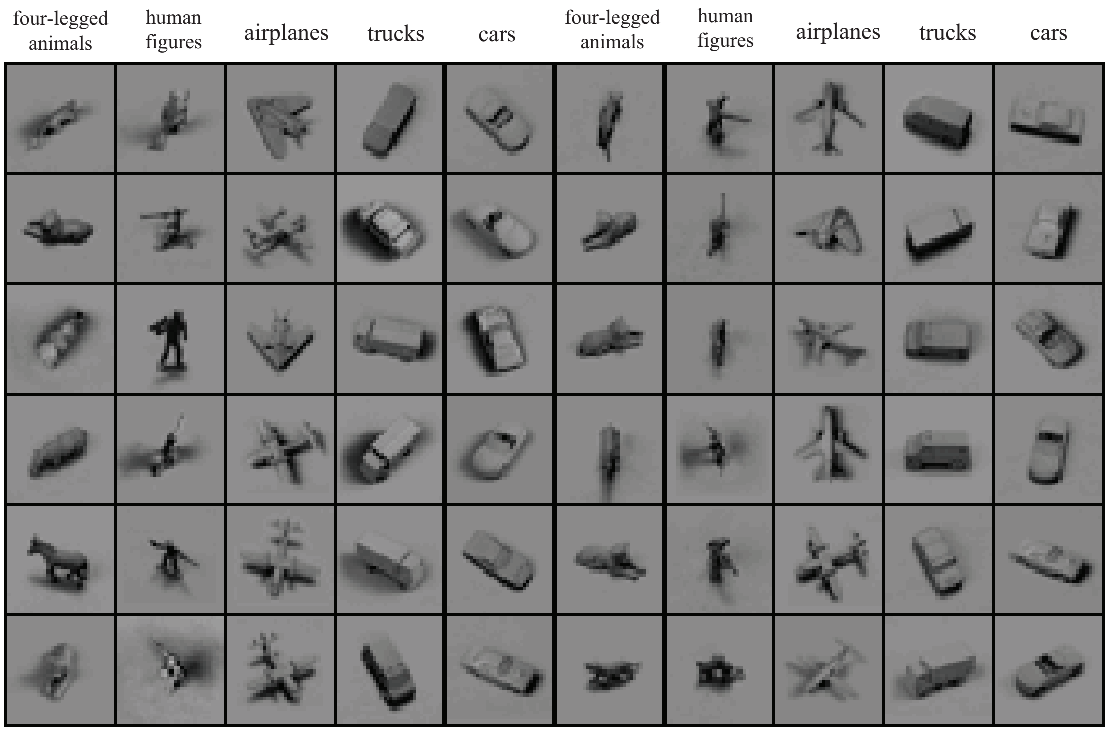


Figure 5.5: 60 samples of the NORB dataset

In this section, we compare the proposed ELM-LRF with state-of-the-art algorithms,

Table 5.1: Parameter specification

# training data	# testing data	Input dimensions	Receptive field	# feature maps	Pooling size	C
24300	24300	$32 \times 32 \times 2$	4×4	48	3	0.01

deep neural networks, etc., on the NORB object recognition dataset [65], which is a common benchmark. NORB dataset consists of 24,300 training images and 24,300 testing images, both of which belong to 5 generic categories. Each image has two channels as it is stereo type. High variations exist among all categories, such as 3D poses, lighting conditions, scales, etc. Fig. 5.5 shows 60 samples of the NORB dataset. The only pre-processing involved is to downsize the images into 32×32.

All experiments are conducted on MATLAB 2013a, Intel Xeon E5-2650, 2GHz CPU, 256GB RAM. Several parameters need to be chosen *a priori* in ELM-LRF: 1) the size of the receptive field $\{4 \times 4, 6 \times 6\}$; 2) the number of feature maps $\{24, 36, 48, 60\}$; the pooling size $\{1, 2, 3, 4\}$; 4) the value of C $\{0.01, 0.1, 1, 10, 100\}$. We hold a validation set out from the training data and choose the optimal parameters based on the validation accuracy. Table 5.1 summarizes the parameters we choose.

5.5.1 Test error

Table 5.2 reports the test errors of the proposed ELM-LRF and related works in the literature [3]. As easily observed, ELM-LRF outperforms other fine-tuned, computation intensive algorithms. The proposed ELM-LRF achieve 2.74% test error, the *best accuracy* in the literature with a significant gap.

5.5.2 Training time

For the comparison of training time, we reproduce other algorithms on our experimental platform. Table 5.3 lists the detailed training time of several algorithms. ELM-LRF

[3]We cite the accuracy of other methods from the literature directly to make the comparison fair.

Table 5.2: Test error rates of different algorithms

Algorithms	**Test error rates**
ELM-LRF	**2.74%**
ELM-LRF (no orthogonalization)	4.01%
Random weights [90]	4.8%
K-means + soft activation [1][14]	2.8%
Tiled CNN [63]	3.9%
CNN [65]	6.6%
DBN [77]	6.5%

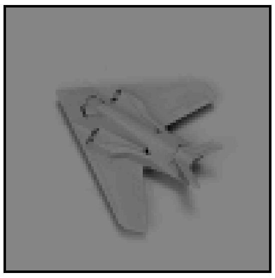
[1] Current state-of-the-art result.

Table 5.3: Training time of different algorithms

Algorithms	**Training time (s)**	**Speedup times [1]**
ELM-LRF	394.16	217.47
ELM-LRF (no orthogonalization)	391.89	218.73
Random weights [2]	1764.28	48.58
K-means + soft activation [3]	6920.47	12.39
Tiled CNN [4]	15104.55	5.67
CNN [5]	53378.16	1.61
DBN [6]	85717.14	1

[1] DBN is used as the standard to calculate the speedup times.
[2] The current most efficient CNN solution. The training time reported in [90] is 0.1h. Reasons for such difference in training time are: i) we have considered convolution and pooling operations as part of training, and the training time reported includes the time spent on convolution and pooling which were not considered in [90]; ii) experiments are run in different experimental platforms.
[3] Use the same parameters as in [14] with 4000 features.
[4] Use the same architecture as ELM-LRF and set the maximum number of iterations in the pretraining to 20.
[5] The architecture is provided in [65] and we use 50 epochs.
[6] The architecture is: 2048 (input)-500-2000-500-5 (output) and 500 epochs since the convergence of training error is slow when dealing with NORB dataset.

accelerates the learning speed up to *200 times* than other algorithms. Moreover, even if we use the random weights method [90], which is the current most efficient method, as the benchmark, the proposed ELM-LRF still realizes 4.48 times speedup and reduces the test error.


(a) The original image (left one)

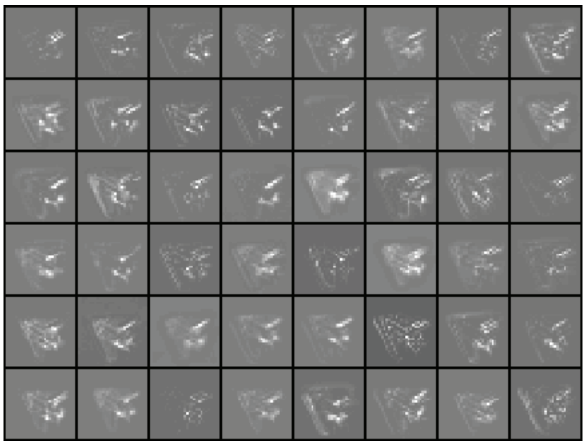

(b) The 48 feature maps

Figure 5.6: The original image (left) and corresponding feature maps

5.5.3 Feature maps

To show what features (representations) are obtained at the higher level of ELM-LRF network, we display the feature maps in the hidden layer. For the input stereo images, only the left one is displayed since the difference between left and right ones is almost unobservable. As seen from Fig. 5.6, all 48 feature maps share similar outlines, since they represent the same input (an airplane). Nevertheless, these maps highlight different parts, producing diversified and comprehensive representations. Jointly, the 48 feature maps generate different representations for the original image and make the subsequent classification accurate and efficient.

5.5.4 Orthogonalization of random input weights

In this section, we explore the improvement that the orthogonalization operation can introduce. We take the center convolutional node in each feature map as the example. The value distributions of the 48 center nodes generated by the orthogonal and non-orthogonal weights are compared. And one sample is chosen from each category and displayed in Fig. 5.7. It can be easily seen that the distribution of orthogonal weights is more broad and even. Similar patterns are discovered for convolutional nodes at other positions in the feature maps. Consequently, the effect of orthogonalization is to make the images easier to be separated and classified.

As listed in Table 5.2, ELM-LRF still produces a 4.01% test error, which decreases 38% compared with conventional deep learning methods, even without orthogonalization. In addition, ELM-LRF (no orthogonalization) provides exceptionally fast solution, making it quite suitable for image processing and similar tasks.

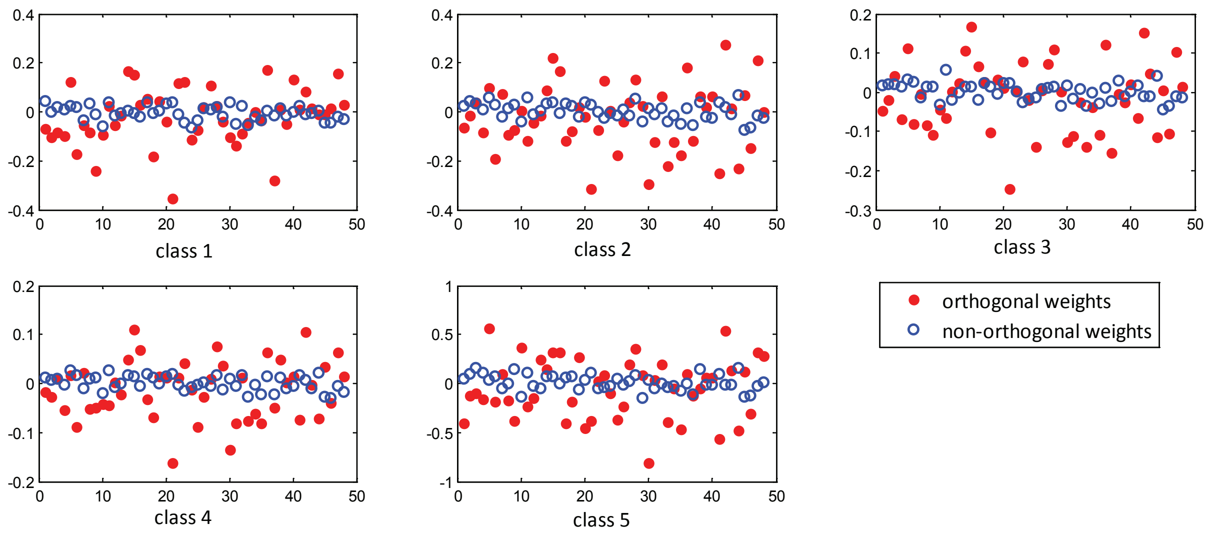


Figure 5.7: The value distributions of the center convolutional nodes in the 48 feature maps with orthogonal and non-orthogonal random weights

5.6 Conclusion

In conclusion, this chapter explores the randomness of ELM in two aspects: random connection and random weights. The network of ELM-LRF, which is in sparse and local connections, explicitly models the local correlations within the input. Thus, it is exceptionally suitable for image analysis, speech recognition, natural language processing, etc. Orthogonalization is followed after random generation of the input weights, forcing the network to obtain more thorough representations for the input. The output weights are calculated analytically, providing a deterministic solution.

In the experiments, detailed comparisons are conducted between the proposed ELM-LRF and related works on NORB dataset, a common benchmark. ELM-LRF reduces the test error to 2.7%, the *best accuracy* in the literature, while accelerating the learning speed up to *200 times*.

Chapter 6

Generic Object Recognition with Local Receptive Fields Based Extreme Learning Machine

Generic object recognition, also called object categorization, is to classify an individual object to a generic category. It faces serious issue of intra-class variabilities, such as different poses, lightings, scales, contracts, etc. Conventional methods are problem dependent and require various pre-processing operations. In this chapter, we suggest to use local receptive field based extreme learning machine (ELM-LRF) as a general framework for generic object recognition. ELM-LRF requires no pre-processing steps as it deals with the original input directly. Furthermore, the network is simple and efficient as most connection weights are generated randomly. We extensively compare the performance of ELM-LRF with state-of-the-art algorithms on several datasets, NORB, ETH-80, COIL and ALOI. It is comparable with the best result on ETH-80 and sets the new records for NORB, COIL and ALOI.

6.1 Introduction

Generic object recognition is to classify an unknown object to a generic category and also known as object categorization [83, 25]. High level of intra-class variabilities, such as different instances of the same category, diverse poses, lightings, scales, etc., pose big problems for this task. Many different methods have been proposed to handle it:

(1) *Shape-based methods*: In [66, 72], explicit shape models are built up, on which the subsequent recognition is performed. In these works, other attributes, like color, texture, etc., are neglected. Later, parts-based shape models are constructed and produce satisfactory performance in some applications [54, 35].

(2) *Appearance-based methods*: Objects of the same category may display various appearance. In these cases, low-level information, such as texture and color histogram, are helpful. The images are highly correlated and will be processed by PCA or other compression methods to generate compact representations as the model images. Hence, the classification criteria will be the similarity between the object itself and the model images [97, 68, 109].

(3) *Local features-based methods*: These works identify some points of interest and extract local features around the identified points. Various types of local features have been used, such as HOG features [18], SIFT features [70], scale invariant descriptors (SIDs) [61], etc.

However, these methods require large amount of human involvement, making it tedious to use. Even worse, these methods are task dependent since the local features or shape models need to be changed if facing new tasks [10, 6]. Consequently, we propose to use the ELM-LRF described in Chapter 5 as a general framework for this task.

6.2 ELM-LRF for Generic Object Recognition

ELM-LRF is operated on the raw images directly. It randomly generates the input weights and analytically calculates the output weights, providing a deterministic solution. Furthermore, unlike convolutional neural networks (CNNs), it does not require BP algorithm to iteratively tune the connection weights. Consequently, the requirements for computational ability and huge training set are both greatly reduced as the input weights need no tuning.

ELM-LRF provides several superiorities over other methods as the general framework for generic object recognition:

(1) *General applicability*: It does not utilize any task-specific information, such as local features, global shapes, etc., thus is applicable for different applications.

(2) *Simple usage*: The only human involvement is to select several parameters based on the validation accuracy, enabling it simple to use.

(3) *High efficiency*: Most connection weights (input weights) are simply generated randomly, making it highly efficient.

We construct the network of ELM-LRF by formulating convolutional nodes and combinatorial nodes as shown in Fig. 6.1. A car image from ETH-80 dataset is used as an example. There are 3 maps (RGB image) in the input layer and K maps in the feature and pooling layers. These maps provide complete representations for the original input.

6.3 Experiments

In this section, we examine the performance of ELM-LRF on several generic object recognition datasets, NORB [65], ETH-80 [66], COIL [78] and ALOI [31]. The exper-

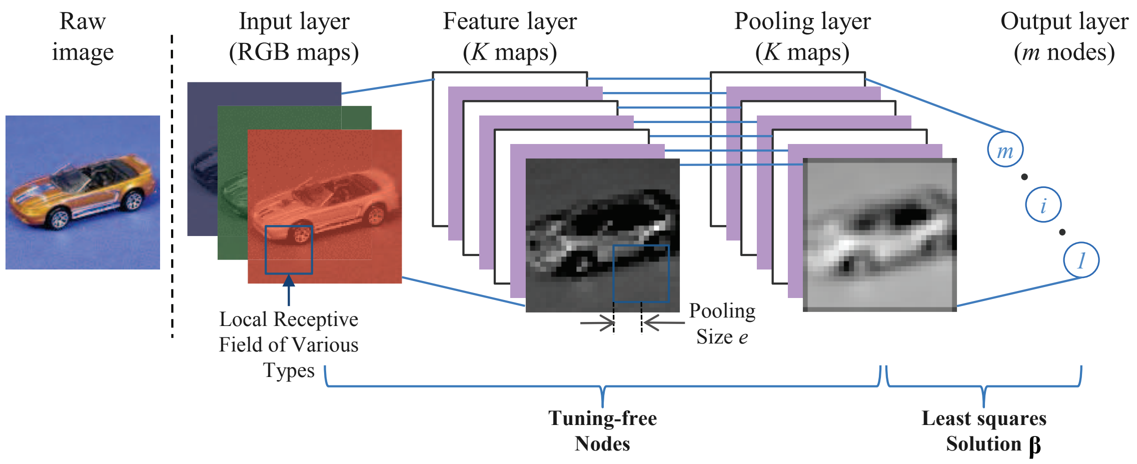


Figure 6.1: The network of ELM-LRF with convolutional hidden nodes

imental platform is MATLAB 2013a, on a Windows Server 2012, Intel Xeon E5-2650, 2GHz CPU, 256G RAM.

6.3.1 Datasets descriptions

These datasets have plenty of variations, including lightings, poses, scales, positions in the image, etc. The only pre-processing is to downsize the original images into 32×32. There are 2 input maps (stereo image) for the NORB dataset and 3 input maps (RGB image) for other datasets.

NORB consists of 5 generic categories, with 24,300 for training and 24,300 for testing. ETH-80 consists of 8 generic categories. Each category has 10 different objects, with 41 viewing angles. For each category, the 10 objects are separated evenly into training and testing sets as in [59]. COIL is consisted of 100 objects with 72 viewing angles (5° increment). The testing set contains images with 20° increment, resulting in 18 views. The training set contains the remaining images, 54 views. ALOI consists of 1,000 objects under three types of variations: illumination colors, illumination directions and viewpoints. And these three types respectively correspond to experiment 1, 2 & 3. Experiment 4 consists of all the data of different variations. For all experiments, 25%

Table 6.1: Datasets descriptions

Dataset	# of categories	# of training data	# of testing data	# of input channels
NORB	5	24300	24300	2
ETH-80	8	1640	1640	3
COIL	100	1800	5400	3
ALOI (Exp1)	1000	3000	9000	3
ALOI (Exp2)	1000	6000	18000	3
ALOI (Exp3)	1000	18000	54000	3
ALOI (Exp4)	1000	27000	81000	3

samples is chosen for training and the other 75% for testing.

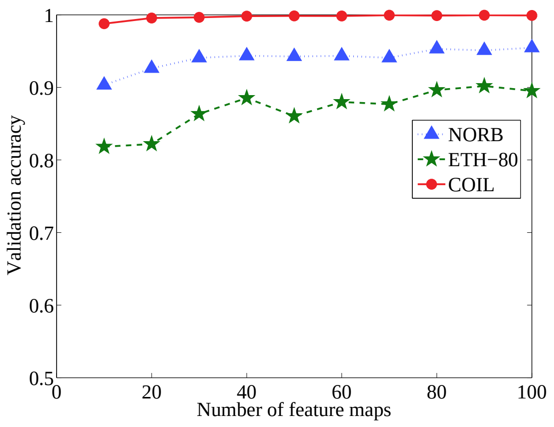
6.3.2 Parameter selections

We select the optimal parameters based on the accuracy on the hold-out validation set. For each dataset, 20% samples of the training set are reserved for validation and the parameters are searched over a grid. Here are the parameters to be chosen: 1) the number of feature maps; 2) the size of receptive field; 3) the pooling size; 4) the value of C.

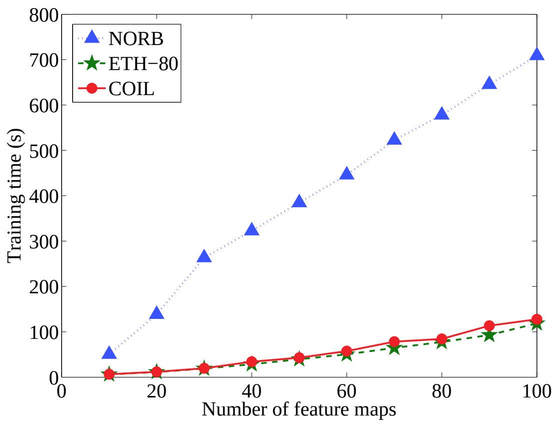
Influence of the number of feature maps K

Feature maps generate diversified, comprehensive representations for the raw images. Thus, the more feature maps, generally the more comprehensive representations. After the number passes a threshold, further increase of K will not improve the performance of ELM-LRF. On the contrary, it may even degrade the performance as it may easily cause overfitting.

There are 4 parameters to be tuned, requiring plenty of human involvement and computations. In this section, we evaluate the performance by varying the values of K from 10 to 100 and aims to find a value that is suitable, though not optimal, for most problems to lighten the computational burden. Naive values are used for other 3 parameters when we explore the values of K: receptive field 4×4, pooling size 5 and $C = 0.01$.



(a) Validation accuracy with varying feature maps



(b) Training time with varying feature maps

Figure 6.2: The validation accuracy and training time with varying feature maps

Table 6.2: Parameter specifications

Dataset	Receptive field	Pooling size	C	# Feature maps (fixed)
NORB	4×4	3	0.01	50
ETH-80	3×3	6	1	50
COIL	7×7	5	1	50
ALOI (all 4 experiments)	4×4	6	1	50

Observing from Fig. 6.2(a), the validation accuracy generally increases with the increment of feature maps, signifying that it has not reached the threshold. However, the training time increases significantly with the increment of feature maps as illustrated in Fig. 6.2(b). We make a compromise at here and fix $K = 50$ feature maps for later experiments.

Parameter specifications

After fixing $K = 50$, other 3 parameters are chosen with grid search based on the validation accuracy. 5 values for the receptive field: 3×3, 4×4, 5×5, 6×6; 5 values for the pooling size: 3, 4, 5, 6, 7; and 3 values for C: 0.01, 1, 100.

In order to reduce the computational burden, parameters of the 4 experiments of ALOI with different variations are selected for ALOI (Exp3) only. Other 3 experiments simply use the same parameters directly. However, it should be reminded that the parameters for

Table 6.3: Test error rates and training time on the NORB dataset

Methods	**Test error rates**	**Training time (s)**
ELM-LRF	**2.76%**	**400.78**
Random weights [90]	4.8%	1764.28
K-means + soft activation [14]	2.8%	6920.47
Tiled CNN [63]	3.9%	15104.55
CNN [65]	6.6%	53378.16
DBN [77]	6.5%	85717.14

other 3 experiments are not necessarily optimal. Table 6.2 lists the selected parameters for all these datasets.

6.3.3 Performance on NORB

Table 6.3 summarizes the test errors and training time of several methods on the NORB dataset [1]. ELM produces the *best accuracy*, while reduces the training time nearly *200 times* compared with deep belief network (DBN) and CNN.

6.3.4 Performance on ETH-80

Fig. 6.3 displays some samples of the ETH dataset. It contains different objects belonging to 8 generic categories with high level of intra-class variations. Table 6.4 lists the error rates of ELM-LRF and related methods. It is observed that ELM-LRF produces error rate on par with state-of-the-art result achieved by DCC method [59]. Furthermore, ELM-LRF is remarkably efficient with 48.64 seconds for training and 15.35 seconds for testing.

[1]The error rates of other methods are cited from the literature directly. However, the training time are recorded on our experimental platform to provide fair comparisons.

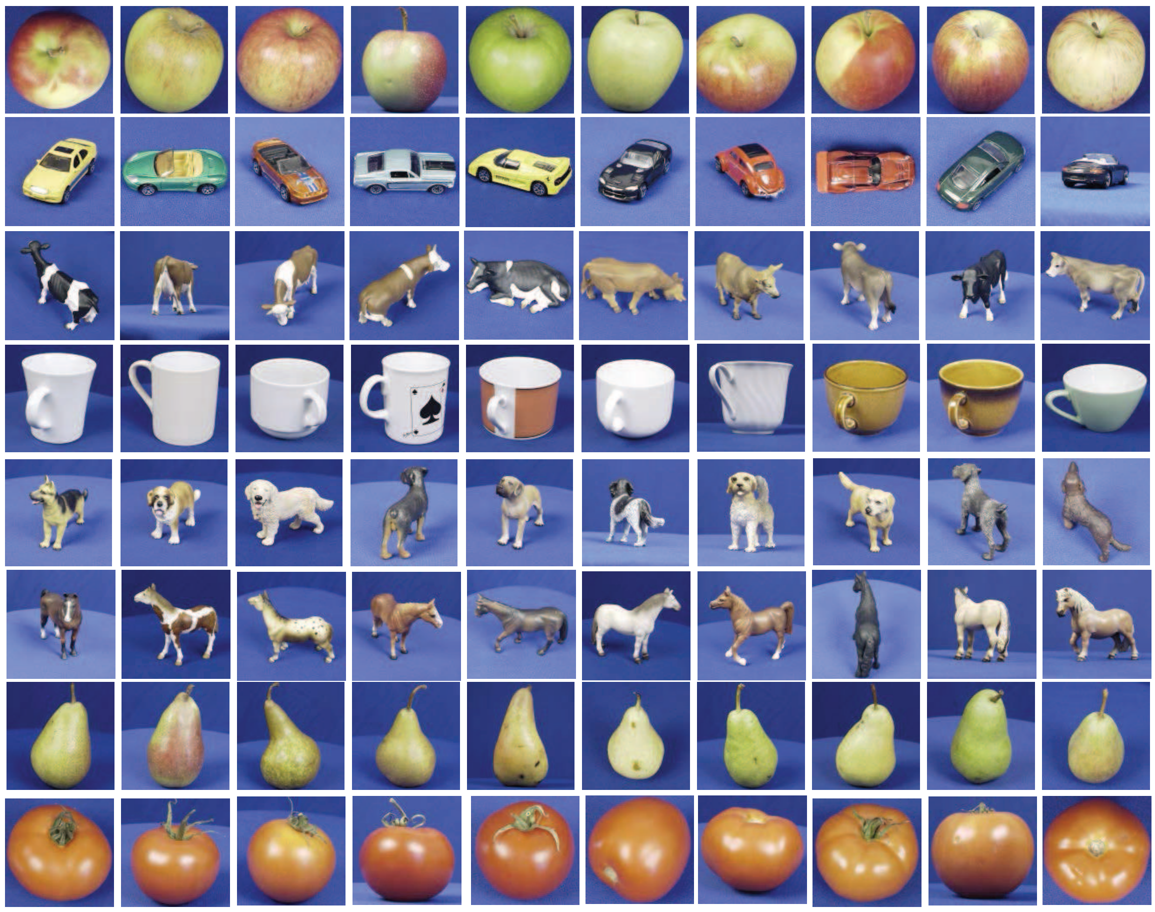


Figure 6.3: Samples of ETH dataset

Table 6.4: Test error rates on the ETH-80 dataset

Methods	**Test error rates**
ELM-LRF	10.0%
Discriminant Analysis of Canonical Correlations (DCC) [59]	**8.3%**
Orthogonal Subspace Method (OSM) [59]	9.5%
Constrained Mutual Subspace Method (CMSM) [79]	10.3%
kNN-LDA [7]	24.8%
kNN-PCA	23.8%

6.3.5 Performance on COIL

Some samples of COIL dataset are shown in Fig. 6.4, which has 100 categories. ELM-LRF outperforms the *best accuracy* on the COIL dataset. Furthermore, some works also investigate the performance of CNN on COIL [76]. ELM-LRF outperforms not only the standard CNN, but also the CNNs with additional information. ELM-LRF presents lower test error by a big gap even compared with CNN plus unlabeled test images or COIL-like images for pre-training. We believe that the relatively too few training samples cannot train CNN properly, making it inferior to ELM-LRF.

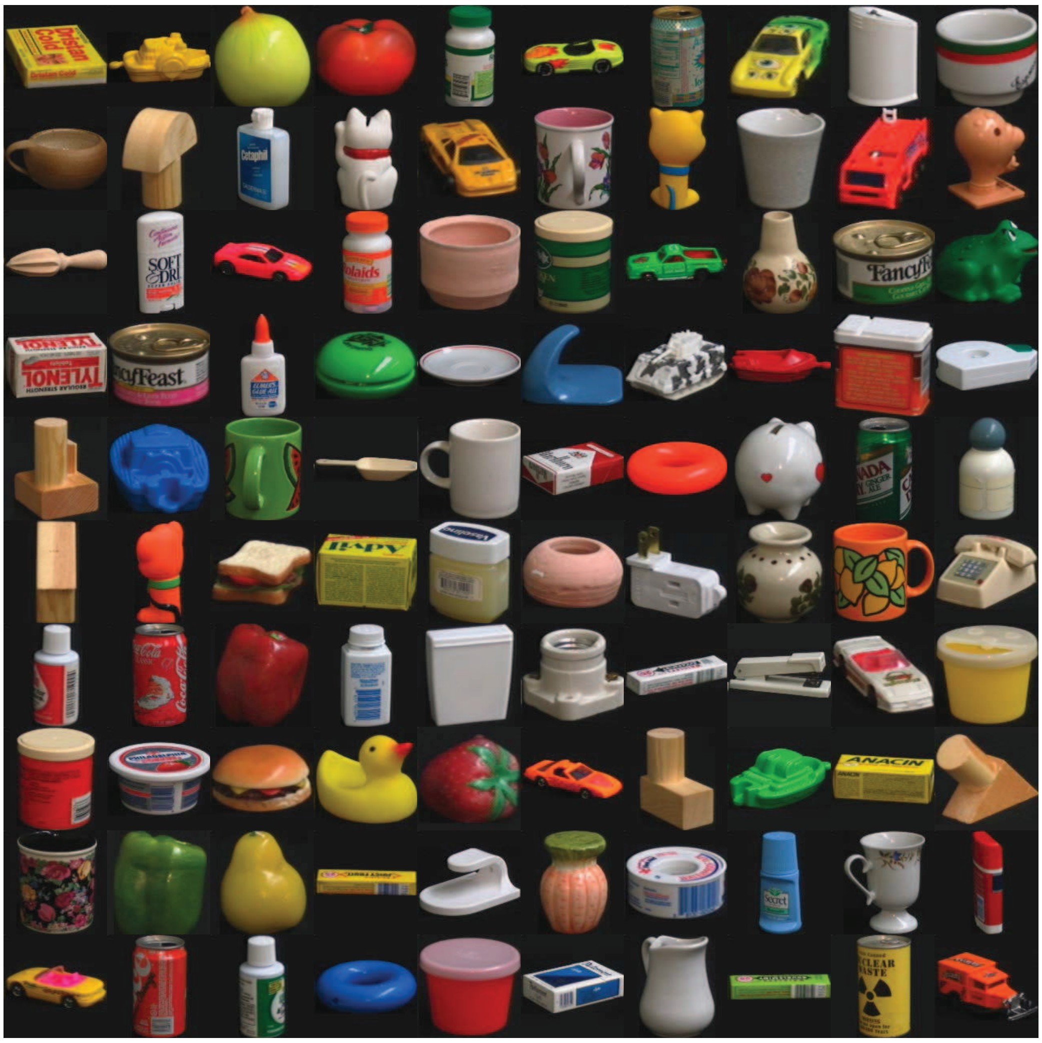


Figure 6.4: Samples of COIL dataset

Table 6.5: Test error rates on the COIL dataset

Methods	**Test error rates**
ELM-LRF	**0.02%**
Local Affine Frames (LAFs) [82] [1]	0.1%
Linear SVM [106]	8.7%
Spin-Glass Markov Random Field (MRF) [72]	3.2%
Standard CNN [76]	28.51%
CNN+*video (test images of COIL)* [76] [2]	7.75%
CNN+*video (COIL-like images)* [76] [3]	20.23%

[1] The current state-of-the-art result.
[2] Use the unlabeled test images as additional learning source. It is a semi-supervised method together with the labeled training images.
[3] Use COIL-like images as additional learning source.

Table 6.6: Test error rates on the ALOI dataset

Methods	**Test error rates**			
	Exp1	Exp2	Exp3	Exp4
ELM-LRF	**0%**	**0.24%**	**0.49%**	**0.80%**
SalBayes [21]	35.21%	24.50%	10.29%	16.17%
SIFT [70]	10.59%	28.53%	29.05%	27.32%
HMAX [91]	0.96%	16.87%	19.24%	16.58%

6.3.6 Performance on ALOI

ALOI dataset includes 1000 generic categories and is even more diversified than previous datasets. Additionally, several types of variations exist in the datasets of ALOI: illumination colors, illumination directions and viewpoints. We conduct experiments 1, 2, 3 and 4 to investigate the performance of ELM-LRF with regard to different variations separately. Table 6.6 shows the detailed performance comparison and ELM-LRF achieves the *best accuracy* for all these four experiments with big improvements.

6.3.7 High efficiency of ELM-LRF

In this section, we discuss the efficiency of the general framework of ELM-LRF. Table 6.7 summarizes the training and testing time for all datasets. ELM-LRF needs less than 0.03 seconds per image for training and about 0.01 seconds for testing. Conse-

Table 6.7: Training and testing time on different datasets

Dataset	**Training stage**		**Testing stage**	
	Total training time	Per image	Total testing time	Per image
NORB	400.78	0.0165	113.7	0.0047
ETH-80	48.64	0.0297	15.35	0.0094
COIL	33.23	0.0185	34.18	0.0063
ALOI (Exp1)	68.05	0.0227	96.11	0.0107
ALOI (Exp2)	116.41	0.0194	192.81	0.0107
ALOI (Exp3)	328.55	0.0183	548.47	0.0102
ALOI (Exp4)	579.19	0.0215	819.91	0.0101

quently, ELM-LRF can be straightforwardly extended to real-time applications because it is capable of dealing with around 100 images per second.

6.4 Conclusion

In this chapter, we suggest ELM-LRF to be used as a general framework for generic object recognition. Compared with traditional methods, ELM-LRF provides several advantages: 1) tasks non-specific since it does not use any task-specific information; 2) easy to use as it requires no pre-processing steps; 3) highly efficient since most connection weights are simply generated randomly. Moreover, different from the newly-emerging CNN, which requires BP method to iteratively tune numerous parameters, ELM-LRF provides a simple and deterministic solution. Thus, compared with CNN, ELM-LRF: 1) largely reduces the computational requirement; 2) is more suitable for applications that does not contain enormous training samples. The experiments on several generic object recognition datasets, NORB, ETH-80, COIL and ALOI, well show the superior accuracy and high efficiency of ELM-LRF.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

In this thesis, we fully investigate the extreme learning machine (ELM) with sparse connections. Unlike previous implementations of ELM, where the hidden nodes are in full connections with the input ones, we focus on the sparse connections. On one hand, the sparse connections reduce the number of connection weights, thus the storage space and addition/multiplication operations in the testing phase. On the other hand, the sparse connections explicitly model the local structures within the input, making it especially suitable for locally correlated applications, such as image analysis, speech recognition, natural language processing, etc.

In Chapter 3 and 4, the sparse ELM is proposed as an alternate for the unified ELM to handle classification and regression problems respectively. The sparse ELM produces comparable accuracy with the unified ELM while beating the commonly used support vector machine (SVM). Additionally, it constructs a sparse network, largely reducing the storage space and testing time. More importantly, the computational complexity of sparse ELM is of lower magnitude than the unified ELM. Thus, it is considerably preferred when dealing with large-scale applications.

In Chapter 5, we explore the properties of local receptive fields and propose the ELM-LRF, which naturally implements the local receptive fields in ELM network. The connections in the ELM-LRF are sparse and bounded by corresponding local receptive fields. ELM-LRF explicitly models the local structures within the input and is remarkably suitable for image processing and similar tasks, where local correlations exist. It presents the *best accuracy* on the NORB dataset, a common benchmark for object recognition, and shortens the training time up to *200 times*.

In Chapter 6, the proposed ELM-LRF is studied as a general framework for generic object recognition. It is operated on the raw input directly and generally applicable for requiring no task-specific information. In addition, the network is simple and deterministic, offering it superiorities over traditional methods and recent CNN networks. ELM-LRF produces high accuracy with exceptional efficiency on several benchmark datasets, NORB, ETH-80, COIL and ALOI.

7.2 Future Works

In the future, we would further investigate the ELM with sparse connections to deal with applications of growing scale, local correlations, etc. In general, there are several topics deserving more research:

(1) *Parallel implementation*: For the sparse ELM, we have developed training algorithms based on iterative computation. Therefore, the training phase can be paralleled by partitioning the training data into several blocks to be handled by different processors.

(2) *Different types of local receptive fields*: As discussed in Chapter 5, various types of probability functions can be utilized to sample the local receptive fields. Thus, we plan to search for other proper local receptive fields that can well represent the local structures within the input data.

(3) *Deep network*: It is promising to build up a deeper network so that more meaningful and complete representations of the original inputs can be produced. The key issue is proper representational learning, such as auto-encoder, manifold learning, convolution, pooling, etc.

Publication list

Journal Papers:

1. **Bai Zuo**, Guang-Bin Huang, Danwei Wang, Han Wang, M. Brandon Westover, “Sparse Extreme Learning Machine for Classification,” *IEEE Transactions on Cybernetics*, vol. 44, no. 10, pp. 1858-1870, 2014

2. Guang-Bin Huang, **Bai Zuo**, Liyanaarachchi Lekamalage Chamara Kasun, Chi Man Vong, “Local Receptive Fields Based Extreme Learning Machine,” *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 18-29, 2015

3. Wan-Yu Deng, Yew-Soon Ong, **Bai Zuo**, Guang-Bin Huang, Qing-Hua Zheng, “A Fast SVD-Hidden-Nodes Based Extreme Learning Machine for Large-Scale Data Analytic,” *Neural Networks*, major revision

Conference Papers:

1. **Bai Zuo**, Liyanaarachchi Lekamalage Chamara Kasun, Guang-Bin Huang “Generic Object Recognition with Local Receptive Fields Based Extreme Learning Machine,” *INNS Big Data 2015, “Soft Computing Methods in Large-scale Content-based Image Retrieval”*, San Francisco, USA, 8-10 August 2015

2. **Bai Zuo**, Guang-Bin Huang, Danwei Wang “Sparse Extreme Learning Machine for Regression”, *The International Conference on Extreme Learning Machine*, Hangzhou, China, 15-17 December 2015, accepted

Bibliography

[1] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999.

[2] Z. Bai, G.-B. Huang, L. L. C. Kasun, C. M. Vong, and C. L. P. Chen. Local receptive fields based extreme learning machine. *IEEE Computational Intelligence Magazine (accepted)*, 10, 2015.

[3] Z. Bai, G.-B. Huang, D. Wang, H. Wang, and M. B. Westover. Sparse extreme learning machine for classification. *IEEE Transactions on Cybernetics*, 44(10):1858–1870, 2014.

[4] O. Barak, M. Rigotti, and S. Fusi. The sparseness of mixed selectivity neurons controls the generalization-discrimination trade-off. *The Journal of Neuroscience*, 33(9):3844–3856, 2013.

[5] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[6] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[7] M. Bressan and J. Vitria. Nonparametric discriminant analysis and nearest neighbor classification. *Pattern Recognition Letters*, 24(15):2743–2749, 2003.

[8] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. SVM and kernel methods matlab toolbox. Perception Systmes et Information, INSA de Rouen, Rouen, France, 2005.

[9] L. J. Cao, S. S. Keerthi, C.-J. Ong, J. Zhang, U. Periyathamby, X. J. Fu, and H. Lee. Parallel sequential minimal optimization for the training of support vector machines. *Neural Networks, IEEE Transactions on*, 17(4):1039–1049, 2006.

[10] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma. PCANet: A Simple Deep Learning Baseline for Image Classification? *ArXiv e-prints*, Apr. 2014.

[11] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[12] W.-L. Chao. Machine learning tutorial.

[13] P.-H. Chen, C.-J. Lin, and B. Schölkopf. A tutorial on ν-support vector machines. *Applied Stochastic Models in Business and Industry*, 21(2):111–136, 2005.

[14] A. Coates, A. Y. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[15] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

[16] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Interscience, 1953.

[17] D. Cox and N. Pinto. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *IEEE International Conference on Automatic Face & Gesture Recognition and Workshops*, pages 8–15. IEEE, 2011.

[18] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[19] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[20] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In M. Mozer, J. Jordan, and T. Petscbe, editors, *Neural Information Processing Systems 9*, pages 155–161, MIT Press, 1997.

[21] L. Elazary and L. Itti. A bayesian model for efficient visual search and recognition. *Vision research*, 50(14):1338–1352, 2010.

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[23] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

[24] 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.

[25] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–264–II–271 vol.2, June 2003.

[26] G. W. Flake and S. Lawrence. Efficient svm regression training with smo. *Machine Learning*, 46(1-3):271–290, 2002.

[27] R. Fletcher. *Practical Methods of Optimization: Volume 2 Constrained Optimization*. John Wiley & Sons, 1981.

[28] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[29] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[30] G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *International Conference on Knowledge Discovery and Data Mining*, pages 77–86, San Francisco, California, USA, 2001.

[31] J.-M. Geusebroek, G. J. Burghouts, and A. W. Smeulders. The amsterdam library of object images. *International Journal of Computer Vision*, 61(1):103–112, 2005.

[32] G. H. Golub and C. Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[33] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[34] S. S. Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall Englewood Cliffs, NJ, 2007.

[35] T. Heimann and H.-P. Meinzer. Statistical shape models for 3d medical image segmentation: A review. *Medical Image Analysis*, 13(4):543 – 563, 2009.

[36] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[37] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[38] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[39] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification, 2003.

[40] 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, 2002.

[41] G. Huang, G.-B. Huang, S. Song, and K. You. Trends in extreme learning machines: A review. *Neural Networks*, 2014.

[42] G. Huang, S. Song, J. N. Gupta, and C. Wu. Semi-supervised and unsupervised extreme learning machines. *IEEE Transactions on Cybernetics*, 44(12):2405–2417, 2014.

[43] G.-B. Huang. An insight into extreme learning machines: Random neurons, random features and kernels. *Cognitive Computation*, 6(3):376–390, 2014.

[44] G.-B. Huang and L. Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70:3056–3062, 2007.

[45] G.-B. Huang and L. Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71:3460–3468, 2008.

[46] 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.

[47] G.-B. Huang, Y.-Q. Chen, H. Babri, et al. Classification ability of single hidden layer feedforward neural networks. *Neural Networks, IEEE Transactions on*, 11(3):799–801, 2000.

[48] G.-B. Huang, X. Ding, and H. Zhou. Optimization method based extreme learning machine for classification. *Neurocomputing*, 74:155–163, 2010.

[49] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 42(2):513–529, 2012.

[50] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: A new learning scheme of feedforward neural networks. In *International Joint Conference on Neural Networks*, volume 2, pages 985–990, 2004.

[51] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70:489–501, 2006.

[52] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

[53] A. K. Jain, J. Mao, and K. Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[54] A. Jaklic, A. Leonardis, and F. Solina. *Segmentation and recovery of superquadrics*, volume 20. Springer, 2000.

[55] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *International Conference on Computer Vision*, pages 2146–2153. IEEE, 2009.

[56] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *arXiv preprint cs/9605103*, 1996.

[57] P. Kara and R. C. Reid. Efficacy of retinal spikes in driving cortical responses. *The Journal of Neuroscience*, 23(24):8547–8557, 2003.

[58] L. L. C. Kasun, H. Zhou, G.-B. Huang, and C. M. Vong. Representational learning with extreme learning machine for big data. *IEEE Intelligent Systems*, 28(6):31–34, 2013.

[59] T.-K. Kim, J. Kittler, and R. Cipolla. Discriminative learning and recognition of image set classes using canonical correlations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1005–1018, June 2007.

[60] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[61] I. Kokkinos and A. Yuille. Scale invariance without scale selection. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[63] Q. V. Le, J. Ngiam, Z. Chen, D. Chia, P. W. Koh, and A. Y. Ng. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1279–1287, 2010.

[64] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[65] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *International Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–97–104, 2004.

[66] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–409–15 vol.2, June 2003.

[67] C. K. L. Lekamalage, T. Liu, Y. Yang, Z. Lin, and G.-B. Huang. Extreme learning machine for clustering. In *Proceedings of ELM-2014 Volume 1*, pages 435–444. Springer, 2015.

[68] A. Leonardis and H. Bischof. Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99–118, 2000.

[69] 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, nov. 2006.

[70] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[71] D. W. Marquaridt. Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, 12(3):591–612, 1970.

[72] J. Matas and S. Obdrzalek. *Object recognition methods based on transformation covariant features*. na, 2004.

[73] M. D. McDonnell, M. D. Tissera, A. van Schaik, and J. Tapson. Fast, simple and accurate handwritten digit classification using extreme learning machines with shaped input-weights. *ArXiv e-prints*, Dec. 2014.

[74] M. D. McDonnell and T. Vladusich. Enhanced Image Classification With a Fast-Learning Shallow Convolutional Neural Network. *ArXiv e-prints*, Mar. 2015.

[75] R. Minhas, A. A. Mohammed, and Q. M. J. Wu. Incremental learning in human action recognition based on snippets. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(11):1529–1541, 2012.

[76] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM, 2009.

[77] V. Nair and G. E. Hinton. 3D object recognition with deep belief nets. In *Advances in Neural Information Processing Systems*, pages 1339–1347, 2009.

[78] S. A. Nene, S. K. Nayar, H. Murase, et al. Columbia object image library (coil-20). Technical report.

[79] M. Nishiyama, O. Yamaguchi, and K. Fukui. Face recognition with the multiple constrained mutual subspace method. In *Audio-and Video-Based Biometric Person Authentication*, pages 71–80. Springer, 2005.

[80] A. Nizar, Z. Dong, and Y. Wang. Power utility nontechnical loss analysis with extreme learning machine method. *IEEE Transactions on Power Systems*, 23(3):946–955, 2008.

[81] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer verlag, 1999.

[82] Š. Obdržálek and J. Matas. Local affine frames for image retrieval. In *Image and Video Retrieval*, pages 318–327. Springer, 2002.

[83] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(3):416–431, March 2006.

[84] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276–285. IEEE, 1997.

[85] M. Pal, A. E. Maxwell, and T. A. Warner. Kernel-based extreme learning machine for remote-sensing image classification. *Remote Sensing Letters*, 4:853–862, 2013.

[86] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[87] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Microsoft Research Technical Report MSR-TR-98-14*, 1998.

[88] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagation errors. *Nature*, 323:533–536, 1986.

[89] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

[90] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng. On random weights and unsupervised feature learning. In *International Conference on Machine Learning*, pages 1089–1096, 2011.

[91] T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 994–1000. IEEE, 2005.

[92] A. Smola, N. Murata, B. Schölkopf, and K.-R. Müller. Asymptotically optimal choice of ε-loss for support vector machines. In *ICANN 98*, pages 105–110. Springer, 1998.

[93] A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[94] D. L. Sosulski, M. L. Bloom, T. Cutforth, R. Axel, and S. R. Datta. Distinct representations of olfactory information in different cortical centres. *Nature*, 472(7342):213–216, 2011.

[95] I. Steinwart, D. Hush, and C. Scovel. Training SVMs without offset. *The Journal of Machine Learning Research*, 12:141–202, Feb. 2011.

[96] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[97] D. Swets and J. Weng. Using discriminant eigenfeatures for image retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(8):831–836, Aug 1996.

[98] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[99] A. Uzilov, J. Keegan, and D. Mathews. Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, 7(1):173, 2006.

[100] V. Vapnik. *The nature of statistical learning theory*. Springer, 1999.

[101] N. Wang, M. Er, and M. Han. Parsimonious extreme learning machine using recursive orthogonal least squares, 2014.

[102] Wikipedia. Computational complexity of mathematical operations – wikipedia, the free encyclopedia, 2013.

[103] Wikipedia. Machine learning — wikipedia, the free encyclopedia, 2015. [Online; accessed 12-January-2015].

[104] V. V. Williams. Breaking the coppersmith-winograd barrier. *Unpublished manuscript, Nov*, 2011.

[105] R. Xu, D. Wunsch, et al. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[106] M.-H. Yang, D. Roth, and N. Ahuja. Learning to recognize 3d objects with snow. In *Computer Vision-ECCV 2000*, pages 439–454. Springer, 2000.

[107] Y. Yang, Y. Wang, and X. Yuan. Bidirectional extreme learning machine for regression problem and its learning effectiveness. *IEEE Transactions on Neural Networks and Learning Systems*, 23(9):1498–1505, 2012.

[108] Z.-H. You, Y.-K. Lei, L. Zhu, J. Xia, and B. Wang. Prediction of protein-protein interactions from amino acid sequences with ensemble extreme learning machines and principal component analysis. *BMC Bioinformatics*, 14(Suppl 8):S10, 2013.

[109] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.

[110] Y. Zhang and P. Zhang. Optimization of nonlinear process based on sequential extreme learning machine. *Chemical Engineering Science*, 66(20):4702–4710, 2011.

[111] W. Zheng, Y. Qian, and H. Lu. Text categorization based on regularization extreme learning machine. *Neural Computing and Applications*, 22:447–456, 2013.

[112] Y. Zhou, J. Peng, and C. L. P. Chen. Extreme learning machine with composite kernels for hyperspectral image classification. *(in press) IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2014.