



THE HONG KONG
POLYTECHNIC UNIVERSITY

香港理工大學

Pao Yue-kong Library

包玉剛圖書館

Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

By reading and using the thesis, the reader understands and agrees to the following terms:

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

IMPORTANT

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact lbsys@polyu.edu.hk providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

EXTENDED ELM-BASED ARCHITECTURES AND
TECHNIQUES FOR FAST LEARNING OF
FEATURE INTERACTION AND
INTERVALS FROM DATA

YINGJIE LI

Ph.D

The Hong Kong Polytechnic University

2016

The Hong Kong Polytechnic University

Department of Computing

**Extended ELM-based Architectures and
Techniques for Fast Learning of Feature
Interaction and Intervals from Data**

by

Yingjie LI

A Thesis Submitted in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy

July 2015

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signature)

_____ LI Yingjie

(Name of Student)

Abstract

This research focuses on the fast learning and extraction of knowledge from data. The particular technique that we adopt in this research is called extreme learning machine (ELM), which is a fast learning algorithm for single layer feed-forward network (SLFN). The ELM theories show that all the hidden nodes can be independent from training samples and do not need to be tuned. In this case, training a SLFN is simply equivalent to finding a least-square solution of a linear system, which can be achieved fast and accurately by using the generalized inverse technique. In this research, several extended ELM-based architectures and techniques are developed for fast learning from data. The contributions of this work can be summarized into three aspects: (i) ELM mapping and modeling, (ii) ELM architecture selection, and (iii) input data compression for ELM.

Focus on the ELM mapping and modeling aspect, a generalized framework named fuzzy ELM (FELM), is developed for fast learning of feature interaction from data. In order to solve the problem of high complexity in determining fuzzy measure, FELM extends the original ELM structure based on the subset selection concept of fuzzy measure. The main contribution is a new set selection algorithm, which transfers the input samples from the original feature space to a higher dimensional feature space for fuzzy measure representation. Then, the fuzzy measure can be obtained using the related fuzzy integral in this high dimensional feature space. The subset selection scheme in FELM is feasible for many

kinds of fuzzy integrals such as Choquet integral, Sugeno integral, Mean-based fuzzy integral and Order-based fuzzy integral. Compared with traditional genetic algorithm (GA) and particle swarm optimization (PSO) algorithm for determining fuzzy measure, FELM achieves faster learning speed and smaller testing error on both simulated data and real data from computer game.

Focus on the ELM architecture selection aspect, an architecture selection algorithm for ELM is developed. This algorithm uses the multi-criteria decision making (MCDM) model in selecting the optimal number of hidden neurons, it ranks the alternatives by measuring the closeness of their criteria. The major contribution is made by introducing a tolerance concept to evaluate a model's generalization capability in approximating unseen samples. Two trade-off criteria, training accuracy and Q-value which is estimated by the localized generalization error model (LGEM), are used. The training accuracy reflects the generalization ability of the model on training samples, and the Q-value estimated by LGEM reflects the generalization ability of the model on unseen samples. Compared with k-fold cross validation (CV) and LGEM, our method achieves better testing accuracy on most of the data datasets with shorter time.

Focus on the data compression aspect, a learning model named interval ELM is developed for large-scale data classification. Two contributions are made for selecting representative samples and removing data redundancy. The first is a newly developed discretization method based on uncertainty reduction inspired by the traditional decision tree (DT) induction algorithm. The second is a new concept named class label fuzzification, which is performed on the class labels of the compressed intervals. The fuzzified class labels can represent the dependency among different classes. Experimental comparison are conducted among basic ELM and Interval ELM with four different kinds of discretization methods. We have achieved a better and more promising result.

Publications

Journal Papers

1. **Yingjie Li**, Peter H.F. Ng, Simon C. K. Shiu, “A Fast Evaluation Method for RTS Game Strategy Using Fuzzy Extreme Learning Machine”, in *International Journal of Natural Computing*, pp 1-13, DOI :10.1007/s11047-015-9484-7, 2014
2. **Yingjie Li**, Ran Wang and Simon Chi-Keung Shiu, “Interval Extreme Learning Machine for Big Data based on Uncertainty Reduction”, in *Journal of Intelligent and Fuzzy System*, 28(2015) 2391C2403, DOI: 10.3233/IFS-141520, 2014.
3. **Yingjie Li**, Peter H.F. Ng and Simon Chi-Keung Shiu, “Extreme learning machine for determining signed efficiency measure from data”, in *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 21(supp02): 131-142, 2013.
4. **Yingjie Li**, Yan Li, Junhai Zhai and Simon Chi-Keung Shiu, “RTS game strategy evaluation using extreme learning machine”, in *Soft Computing*, vol. 16, no. 9, pp. 1627-1637, 2012.

Conference Papers

1. **Yingjie Li**, Peter H. F. Ng and Simon Chi-Keung Shiu. “Rapid Game Strategy Evaluation Using Fuzzy Extreme Learning Machine”, in proceeding of *5th International Conference on Pattern Recognition and Machine Intelligence*, Kolkata, India, pp.250-255, 2013.
2. Peter H. F. Ng, **Yingjie Li** and Simon Chi-Keung Shiu. “New Fuzzy Integral for the Unit Maneuver in RTS game”, in proceeding of *5th International Conference on Pattern Recognition and Machine Intelligence*, Kolkata, India, pp.256-261, 2013.
3. **Yingjie Li**, Peter H. F. Ng, Haibo Wang, Yan Li and Simon Chi-Keung Shiu. “Applying Fuzzy Integral for Performance Evaluation in Real Time Strategy Game”, in proceeding of *2012 International Conference on Machine Learning and Cybernetics*, Xian, China, Vol. 1, pp. 289-295, 2012.

Acknowledgements

It has been fascinating to study in the Hong Kong Polytechnic University over the past several years. Lucky for me, I had the opportunities to work with a number of wonderful researchers. This thesis would not be possible to be completed without their generous help. I would like to express my appreciation to all of them in this acknowledgement.

First, it is my pleasure to express my heartfelt thanks and deepest gratitude to my supervisor, Dr. Simon Chi Keung Shiu. I would like to thank him for his professional supervision and continuous support during my MPhil and PhD study period. Dr. Shiu is a kind and optimistic person. He always encourages me to become confident in overcoming difficulties. I appreciate his help, insight and patience in my research study and he could always guide me in a correct research direction. The experience as his student in these years is of great benefit to my life, both in terms of scientific knowledge as well as the proper way to elaborate the works to others.

Then, I wish to express my warm and most sincere gratitude to another prominent and distinguished person, Prof. Xizhao Wang. I would like to express my gratitude to him for his valuable advices during my PhD study. Prof. Wang always keeps the strictest attitude in doing research. He is extremely knowledgeable and offers me many useful suggestions about my research. His acuminous insight and professional guidance is of great help for my study.

Next, I want to thank all of my friends, my past and present colleagues Peter H. F. Ng, Haibo Wang, Yan Li and Pan Su. Thanks for their support, assistance, and advices in all research and administrative matters.

Last, but not the least, I wish to express my deepest appreciation to my parents for their unrequited love. Especially, I thank my wife, Dr. Ran Wang. I appreciate very much that she could spend her time to support me in paper writing and give me many useful comments and feedbacks. Without her, my life in Hong Kong can not be such enjoyable.

Table of Contents

Abstract	i
Publications	iii
Acknowledgements	v
Table of Contents	vii
List of Figures	viii
List of Tables	x
1 Introduction	1
1.1 Motivations and Objectives	1
1.2 Problems and Methodologies	4
1.3 Structure of the Thesis	6
2 Literature Review	9
2.1 Learning Theories of ELM	9
2.1.1 Interpolation Theorem	11
2.1.2 Universal Approximation Theorem	13
2.1.3 ELM Algorithm	17
2.2 Fuzzy Measure and Fuzzy Integral	19
2.2.1 Fuzzy Measure	20
2.2.2 λ -Fuzzy Measure	21
2.2.3 Choquet Integral	22
2.2.4 Sugeno Integral	23
2.3 Architecture Selection	24
2.4 Techniques for Handling Large-Scale Data	24
2.4.1 Discretization of Continuous-Valued Attributes	24
2.4.2 Center and Range	27

2.4.3	Feature Selection	28
3	ELM-based Fuzzy Measure Determination	31
3.1	Introduction	31
3.2	Determining Signed Efficiency Measure from Data	34
3.3	Set Selection Algorithm for Fuzzy Integrals	34
3.3.1	Fuzzy Integrals	36
3.4	Design of ELM	39
3.4.1	Characteristics of the FELM	43
3.5	Experimental Comparisons	44
3.5.1	Methods of Comparison	44
3.5.2	Description of Type-II Data	45
3.5.3	Performance Comparison on Type-II Data	47
3.5.4	Description of Warcraft III Data	50
3.5.5	Performance Comparison on Warcraft III Data	51
3.6	Conclusion	54
4	Architecture Selection for SLFNs with MCDM Model	57
4.1	Introduction	57
4.2	Localized Generalization Error Model	59
4.2.1	Generalization Error of Classifier	59
4.2.2	LGEM	60
4.2.3	Architecture Selection Based on LGEM for ELM	63
4.3	LGEM Based Architecture Selection with MCDM Model	64
4.3.1	Trade-off Between Training Accuracy and Q -Neighborhood	65
4.3.2	MCDM Model	66
4.3.3	Proposed Architecture Selection Algorithm	70
4.4	Experimental Comparisons	71
4.4.1	Methods of Comparison	71
4.4.2	Experimental Setting	71
4.4.3	Experimental Analysis	73
4.4.4	Trade-off Analysis	75
4.5	Conclusion	77
5	Interval ELM for Large-scale Data Based on Uncertainty Reduction	79
5.1	Introduction	79
5.2	Challenges in Learning from Large-Scale Data for ELM	81

5.3	Discretization of Conditional Attributes and Fuzzification of Decision Label	82
5.3.1	Uncertainty Measurement	82
5.3.2	Discretization of Conditional Attributes Based on Uncertainty Reduction	83
5.3.3	Fuzzification of Decision Label	84
5.4	Interval Extreme Learning Machine	88
5.5	Experimental Comparisons	90
5.5.1	Methods of Comparison	90
5.5.2	Experimental Design	91
5.5.3	Performance Comparison on Small UCI Benchmark Data	92
5.5.4	Performance Comparison on Large-Scale Data	94
5.5.5	Performance Comparison with SVM and KNN	95
5.6	Conclusions	98
6	Conclusions and Future Works	101
6.1	Evaluation on the Developed ELM Techniques	101
6.1.1	Fuzzy Extreme Learning Machine	101
6.1.2	LGEM based Architecture Selection for ELM with MCDM Model	103
6.1.3	Interval Extreme Learning Machine	104
6.2	Future Works	105
	Appendix A.	107
A.1	Derivation of the Stochastic Sensitivity Measure for Single-Layer Feedforward Network with Sigmoid Function	107
	Bibliography	111

List of Figures

1.1	The extended ELM-based techniques and architectures developed in this research	6
2.1	Generalized ELM architecture	10
2.2	Classification ability comparison between two-layers network and three layers network	16
3.1	Comparison among CI, Mean-based FI and Order-based FI.	39
3.2	ELM-based architecture for describing fuzzy integrals	41
3.3	Performance Comparison of Different Methods on Type-II data sets: C=CI, M=Mean-based FI, and O=Order-based FI	48
3.4	Performance Comparison of Different Methods on the Noisy data sets: C=CI, M=Mean-based FI, and O=Order-based FI	49
3.5	Performance Comparison of Different Methods on Warcraft III data sets: C=CI, M=Mean-based FI, and O=Order-based FI	53
4.1	Mean and standard deviation of training accuracy and Q -value of 20 ELM trials.	76
5.1	Learning structure of IELM: G_s is the heuristic calculated by CFS	89
5.2	Testing accuracy with different settings of parameters N^* and θ of the proposed method.	99
5.3	Percentage of the compressed data size to the original data size with different settings of parameters N^* and θ	100

List of Tables

3.1	Set selection of different fuzzy integrals	40
3.2	The Original Fuzzy Measure for Constructing Type-II Data	46
3.3	Detailed Information of Data Set 2	47
3.4	Nature of Testing Data Set	51
3.5	Fuzzy Measure Subsets ($n \leq 2$) in Data Set 1	52
4.1	Distances between relations.	69
4.2	Selected Datasets for Performance Comparison	72
4.3	Comparisons of Different Model Selection Methods: Average Number of Nodes and Testing Accuracy for 50 Trials	75
4.4	Comparisons of Different Model Selection Methods: Average Validation Time for 50 Trials	77
5.1	Selected Small Binary Data Sets for Performance Comparison . .	95
5.2	Optimal Parameter Settings of Different Methods on the Selected Small Data Sets	96
5.3	Compression Rate (%) of the large-scale data Set <i>Skin</i>	96
5.4	Performance Comparisons of Different Methods on the Selected Small Data Sets: Testing Accuracy (%) and Compression Rate (%)	96
5.5	Paired Wilcoxon's Signed Rank Tests of Testing Accuracies (p Values)	97
5.6	Comparative Results on the large-scale data Set <i>Skin</i> : Testing Accuracy (%) and Training Time (seconds)	97
5.7	Performance Comparison of the Proposed Method with SVM and KNN on the Selected Small Data Sets	97

Chapter 1

Introduction

In this chapter, we present the motivations, objectives, and methodologies of the proposed works, and give an overview of the thesis.

1.1 Motivations and Objectives

Machine learning is an important topic in artificial intelligence, which is regarded as the problem of learning knowledge models from data, for the purpose of future predictions on unseen data. In many real world applications, a simple, flexible and efficient learning technique is required for problem solving. Unfortunately, many existing popular learning techniques, such as genetic algorithm (GA), particle swarm optimization (PSO), and artificial neural network (ANN), are time consuming. The training process of these methods involves a large number of iterations in tuning the model parameters, i.e., learning rate, stopping criteria, and inertia weight, etc [36], which adds certain difficulties to their applications. Thus, the development of fast and efficient learning techniques are crucial for both research evolution and industrial problem solving, which is also the major motivation of this research.

Over the past decades, single hidden layer feedforward neural networks (SLFNs) have been thoroughly studied by researchers. Supported by the universal approximation theory, a SLFN with enough number of neurons is proved to be capable of approximating continuous functions on compact subsets of R^n (n is any positive integer), which adopts a weak assumption on the activation function [73]. This special property demonstrates the great potential of SLFN in knowledge extraction and pattern recognition. However, existing learning algorithms for SLFNs are usually suffered from the problems of slow convergence and local minimum. This is because the development of these algorithms are mostly based on the delta rule [80], which is a gradient descent learning rule for updating weights of different neurons. The most popular algorithm for training SLFNs is back-propagation (BP). The impact of BP algorithm on the neural network research is enormous, however, it is also widely realized that this algorithm is suffered from high complexity, due to the large number of training iterations.

In [24], Huang et al. proposed a new learning system for SLFNs with fast training speed, which is named extreme learning machine (ELM). The ELM theories indicate that SLFNs with piecewise continuous activation function can maintain the generalization capability when the weights of input neurons are randomly generated. Under this condition, the parameters of the output layer can be estimated by applying the generalized inverse technique. Compared with traditional BP algorithm, ELM is proved to be capable of achieving better generalization ability and faster learning speed in many applications. This research is to study ELM-based methods for fast learning, in the context of feature interaction and intervals from data. Moreover, an effective architecture selection technique is also

provided for reducing unimportant nodes in a neural network.

In particular, this research is also motivated by the following open issues on ELM:

- **Fast and accurate representation of interacted data:** Feature interactions among different attributes are usually complicated and hard to explain. Traditional learning techniques are difficult to solve this highly non-additive measurement problem. Thus, the development of fast and effective technique for learning feature interaction from data has attracted considerable interests over the past decade. Developing ELM-based method for learning interacted data should be a worth studying topic.
- **Optimal architecture selection for SLFNs:** The number of hidden nodes is a critical factor in applying ELM to problem solving, which needs to be set by users. In theory, the expected performance of an estimator in predicting new observations should be good when its model structure “best fits” the data. However, how to automatically determine the number of hidden nodes for a given task is still an unsolved problem.
- **Fast classification method for large-scale data:** Choosing representative samples and removing data redundancy are two key issues in learning from large-scale data. Due to the fast learning speed, ELM has a great potential for solving these problems. However, how to reduce the high demand on time and space for matrix calculation in ELM is still a challenging issue.

1.2 Problems and Methodologies

In order to achieve the above objectives, several extended ELM-based architectures and techniques are proposed in this thesis. The research problems and their corresponding methodologies are listed as follows.

- **ELM-based fuzzy measure determination.** Due to the non-additive and non-linear characteristics, fuzzy measure and fuzzy integral are commonly used in describing the feature interactions among different attributes. In this part, a fast learning model named fuzzy ELM (FELM) is developed as a generalized framework for fuzzy measure determination. The model structure is developed based on the set selection concept of fuzzy integral, which transfers the fuzzy measure determination to the problem of finding the least-square solution of a linear system. A subset selection scheme is developed for different kinds of fuzzy integrals, in order to represent the relationship among different layers. Afterwards, the fuzzy measure are analytically determined according to the output weights of ELM. Compared to existing methods for fuzzy measure determination such as genetic algorithm (GA) [75] and particle swarm optimization (PSO) [77], FELM achieves a faster learning speed and a higher testing accuracy.
- **Architecture selection for SLFNs trained by ELM.** How to select an appropriate number of hidden nodes (network architecture) is an important problem for designing a neural network. This architecture selection process could be treated as a decision making problem, which evaluates and makes choices from a finite set of candidate architectures. Multi-criteria deci-

sion making (MCDM) model is a widely known branch of decision making, which evaluates the alternatives on the basis of multiple criteria. In this part, an architecture selection method with MCDM model is proposed. In this model, the optimal number of hidden nodes is selected by achieving a trade-off between the generalization ability on training samples and unseen samples. Two conflicting criteria, training accuracy and Q -value (i.e., the union size of input perturbations) estimated by the localized generalization error model (LGEM), are adopted in decision making. The most satisfactory architecture is sought by making compromise between these two criteria.

- **Large-scale data compression for fast classification using ELM.** Selecting representative samples and removing data redundancy are two key issues in learning from large-scale data. This part proposes a new model, named interval ELM (IELM), for classification on large-scale data with continuous-valued attributes. This model is constructed with two techniques, i.e., discretization of conditional attributes and fuzzification of output labels. First, each attribute is discretized into a number of intervals based on uncertainty reduction measurement. Then, the samples with the same interval value with regard to all the attributes are merged as one record, and a fuzzification process is performed on their class labels. Finally, the original data set can be reduced into a smaller one with fuzzy labels, which can represent the dependency among different classes. As a result, the IELM model is developed.

1.3 Structure of the Thesis

The structure of this thesis is summarized in Fig. 1.1.

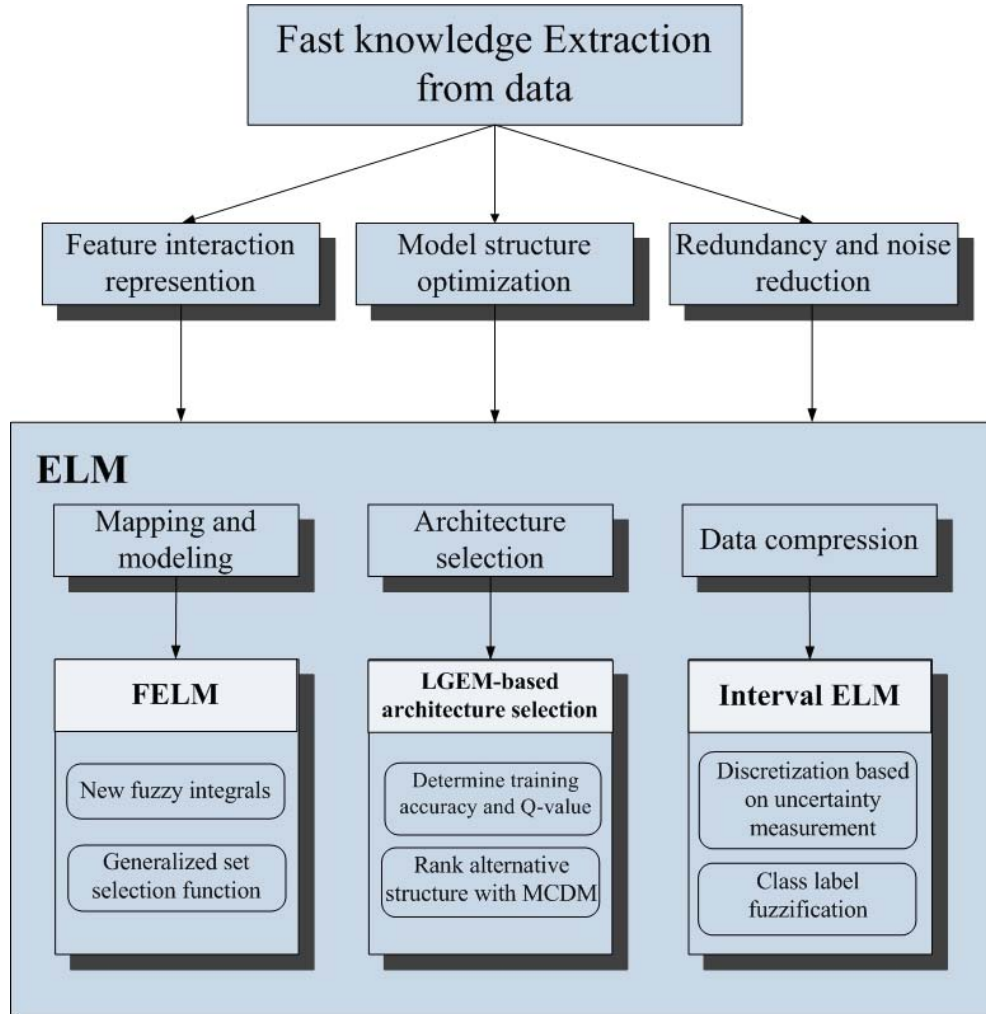


Figure 1.1: The extended ELM-based techniques and architectures developed in this research

The remainder of this thesis is organized as follows. Chapter 2 gives a detailed literature review, which provides preliminary knowledge on the proposed works. Chapter 3 develops the FELM model for fast determination of signed efficiency measure. Chapter 4 provides a LGEM based architecture selection method with

MCDM model. Chapter 5 presents the IELM for big data learning problem.

Conclusion and future works are given in Chapter 6.

Chapter 2

Literature Review

This chapter firstly provides a brief review on the learning theory of ELM, followed by some background knowledge on fuzzy measure and fuzzy integral. Secondly, the basic techniques used in architecture selection and data compression are also discussed.

2.1 Learning Theories of ELM

SLFN is a simple form of ANN, which consists of three layers, i.e., one input layer for information collection, one output layer for sending message to external environment, and one hidden layer for feature mapping [30]. As point out in [30], there are three commonly used approaches for training SLFNs: 1) standard optimization based method [62] (e.g., support vector network); 2) gradient descent based method (e.g., BP), and 3) least-square based method [50] (e.g., radial basis function (RBF)). It is commonly known that the above learning techniques suffer from several challenging issues, such as slow learning speed, large number of parameters, and poor computational scalability.

ELMs can be regarded as “generalized” SLFNs, which even does not need to

be neuron like [30]. Huang et al. [24] state that ELM have three major characteristics: 1) the weights of hidden neurons can be randomly assigned, which could be independent of the inputs; 2) the weights connecting the hidden layer and the output layer could be determined by the least-square method; and 3) the training error will be the smallest and the norm of the output weights obtained is also the smallest. A typical ELM model is shown in Fig. 2.1. The standard feedforward neural network used by ELM is efficient in regression, classification and clustering [30]. Moreover, the random hidden neurons need not be algebraic sum based.

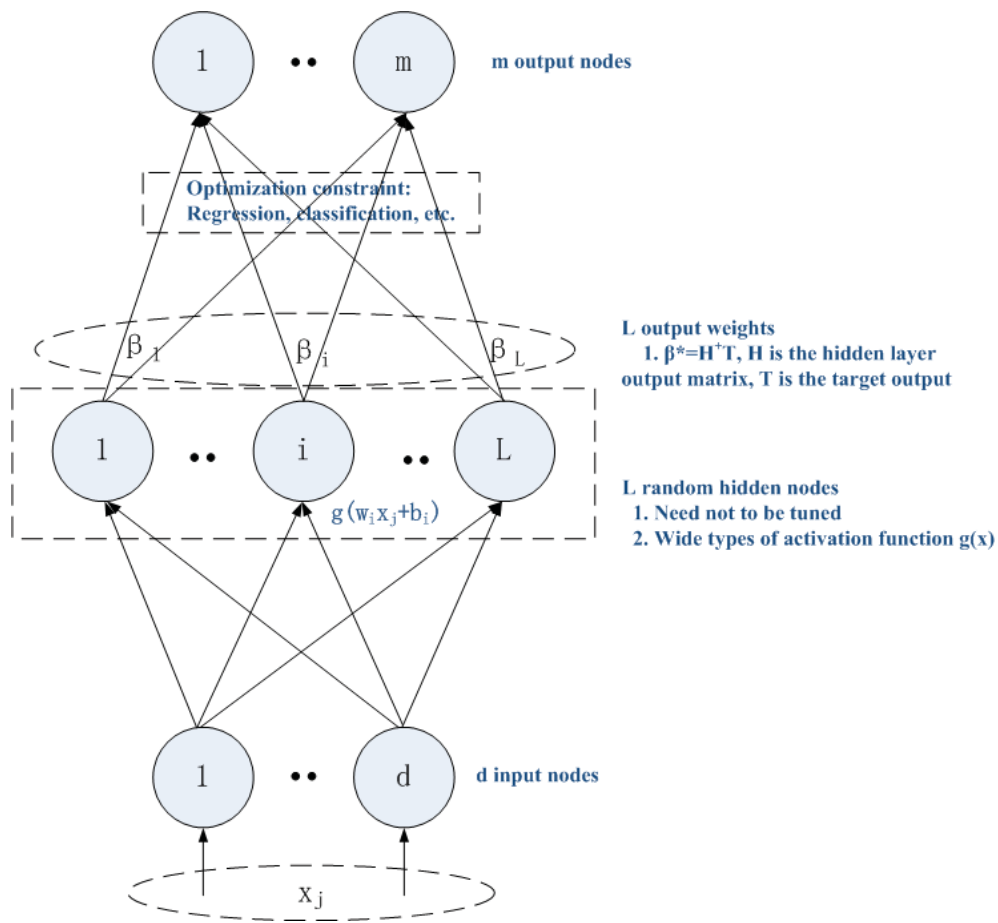


Figure 2.1: Generalized ELM architecture

The research on function approximation capabilities of feedforward neural networks has been concentrated on two aspects: universal approximation capability and interpolation capability. In the past two decades, researchers have investigated these two capabilities of standard multilayer feedforward neural networks thoroughly. Hornik [22] proved that a neural network can approximate continuous mapping over compact input data if its activation function is continuous, bounded and non-constant. Leshno [42] further proved that feedforward neural networks with non-polynomial activation function is capable to approximate continuous functions. Based on the above important theorems, Huang and Babri [25] proved that a SLFN with (at most) N hidden nodes can learn any N arbitrary distinct samples with zero error in case the activation function is piecewise continuous. The commonly used activation functions (such as sigmoid function, threshold function, and RBF) can satisfy this requirement. The interpolation capability and universal approximation capability of ELM [30] are detailed in the following section.

2.1.1 Interpolation Theorem

Given a training set \mathbb{X} that contains N distinct samples with n input features and m output classes, i.e., $\mathbb{X} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N$ where $\mathbf{x}_i = [x_{i1}, \dots, x_{in}]$, $\mathbf{t}_i = [t_{i1}, \dots, t_{im}]$, and $t_{ik} \in \{0, 1\}, k = 1, \dots, m$, the SLFN with L hidden nodes and activation function $g(\mathbf{x})$ is formulated as:

$$\sum_{j=1}^L \beta_j g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j) = \mathbf{o}_i, \quad i = 1, \dots, N, \quad (2.1)$$

where $\mathbf{w}_j = [w_{j1}, \dots, w_{jn}]^T$ is the weight vector connecting the input nodes and the j -th hidden node, b_j is the bias of the j -th hidden node, and β_j is the weight connecting the j -th hidden node and the output nodes, and \mathbf{o}_i is the network output of \mathbf{x}_i .

It is proved that the SLFN can approximate the training samples with zero error [30], thus we have

$$\sum_{j=1}^L \beta_j g(\mathbf{w}_j \cdot \mathbf{x}_i + b_j) = \mathbf{t}_i, \quad i = 1, \dots, N. \quad (2.2)$$

Eq. (2.2) can be rewritten into a matrix form, i.e.,

$$\mathbf{H} \cdot \beta = \mathbf{T}, \quad (2.3)$$

where \mathbf{H} is the hidden layer output matrix, i.e.,

$$= \begin{matrix} \mathbf{H}(\mathbf{w}_j, b_j, \mathbf{x}_i) \\ \left[\begin{array}{ccc} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \dots & g(\mathbf{w}_L \cdot \mathbf{x}_1 + b_M) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \dots & g(\mathbf{w}_L \cdot \mathbf{x}_N + b_M) \end{array} \right]_{N \times L}, \end{matrix} \quad (2.4)$$

$\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_N]_{N \times m}^T$ is the target output of the training samples, and $\beta = [\beta_1, \dots, \beta_L]_{L \times m}^T$ is the output weight.

In the hidden layer output matrix \mathbf{H} , each column indicates the output value of the corresponding hidden node [1, 31], which satisfy the following theorems.

Theorem 2.1.1. [30] Given a standard SLFN with N hidden nodes and activation function $g : \mathcal{R} \rightarrow \mathcal{R}$ that is infinitely differentiable in any interval, for N arbitrary distinct samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N \in \mathcal{R}^n \times \mathcal{R}^m$, for any $\{(\mathbf{w}_j, b_j)\}_{j=1}^L$ randomly generated from any interval of $\mathcal{R}^n \times \mathcal{R}$, according to any continuous probability

distribution, with probability one, we have $\|\mathbf{H}_{N \times N} \cdot \beta_{N \times m} - \mathbf{T}_{N \times m}\| = 0$

Theorem 2.1.2. [30] Given any small positive value $\varepsilon > 0$ and activation function $g : \mathcal{R} \rightarrow \mathcal{R}$ that is infinitely differentiable in any interval, there exists $L \leq N$ such that for N arbitrary distinct samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N \in \mathcal{R}^n \times \mathcal{R}^m$, for any $\{(\mathbf{w}_j, b_j)\}_{j=1}^L$ randomly generated from any interval of $\mathcal{R}^n \times \mathcal{R}$, according to any continuous probability distribution, with probability one, we have $\|\mathbf{H}_{N \times L} \cdot \beta_{L \times m} - \mathbf{T}_{N \times m}\| < \varepsilon$

Theorem 2.1.1 and Theorem 2.1.2 can be extended to any kind of nonlinear piecewise continuous function [30]. Such function including threshold function, even though it is not differentiable [30].

2.1.2 Universal Approximation Theorem

Definition 2.1.3. [30] A function $g(\mathbf{x}) : \mathcal{R} \rightarrow \mathcal{R}$ is considered to be *piecewise continuous* if it has only a finite number of discontinuities in any interval and its left and right limits are defined (not necessarily equal) at discontinuity.

Definition 2.1.4. [30] A node is named a *random node* if its parameters (\mathbf{w}_j, b_j) are randomly generated based on a continuous sampling distribution probability.

The adoption of random nodes is the typical feature of ELM. Moreover, all the parameters (\mathbf{w}_j, b_j) can be generated independently.

Definition 2.1.5. [30] Let $\mathcal{L}^2(\mathbb{X})$ be a space of function f on a compact subset \mathbb{X} in the n -dimensional Euclidean space \mathcal{R}^n , such that $\|f\|^2$ are integrable and $\int_{\mathbb{X}} |f(\mathbf{x})|^2 d\mathbf{x} < \infty$. Denote $\mathcal{L}^2(\mathbb{X})$ as \mathcal{L}^2 , for $u, v \in \mathcal{L}^2$ the inner product $\langle u, v \rangle$ is defined by $\langle u, v \rangle = \int_{\mathbb{X}} u(\mathbf{x})v(\mathbf{x})d\mathbf{x}$

In this case, $\mathcal{L}^2(\mathbb{X})$ in the form of Eq. (2.5) is used to measure the distance between the network function f_L and the target function f , where $\|\cdot\|$ represents the norm in $\mathcal{L}^2(\mathbb{X})$ space, i.e.,

$$\|f_L - f\| = \left[\int_{\mathbb{X}} |f_L(\mathbf{x}) - f(\mathbf{x})|^2 d\mathbf{x} \right]^{1/2} \quad (2.5)$$

Lemma 2.1.6. [42] Given $g : \mathcal{R} \rightarrow \mathcal{R}$, $\text{span}\{g(\mathbf{w} \cdot \mathbf{x} + b) : (\mathbf{w}, b) \in \mathcal{R}^n \times \mathcal{R}\}$ is dense in $\mathcal{R} = \mathcal{L}^p$ for every $p \in [1, \infty)$, if and only if g is not polynomial (almost everywhere).

Lemma 2.1.7. [30] Let $g : \mathcal{R}^n \rightarrow \mathcal{R}$ be an integrable bounded function such that g is continuous (almost everywhere) and $\int_{\mathcal{R}} g(\mathbf{x}) d\mathbf{x} \neq 0$. Then $\text{span}\{g(\frac{\mathbf{x}-\mathbf{w}}{b}) : (\mathbf{w}, b) \in \mathcal{R}^n \times \mathcal{R}^+\}$ is dense in \mathcal{L}^p for every $p \in [1, \infty)$.

Lemma 2.1.6 and Lemma 2.1.7 state that feedforward neural networks with additive or RBF nodes can approximate any continuous target function with appropriate parameters (\mathbf{w}_j, b_j) . Based on these results, Huang et al. [30] further proved that SLFNs which randomly assigned hidden nodes can still approximate arbitrary target function. The related theorems are listed as follows.

Theorem 2.1.8. [28] Given any bounded nonconstant piecewise continuous function $g : \mathcal{R} \rightarrow \mathcal{R}$ for additive nodes or any integrable piecewise continuous function $g : \mathcal{R} \rightarrow \mathcal{R}$ and $\int_{\mathcal{R}} g(\mathbf{x}) d\mathbf{x} \neq 0$ for RBF nodes, for any continuous target function f and any randomly generated function sequence g_L , $\lim_{L \rightarrow \infty} \|f - f_L\| = 0$ holds with probability one if

$$\beta_L^{(j)} = \frac{\langle e_{L-1}^{(j)}, g_L \rangle}{\|g_L\|^2}, k = 1, \dots, m, \quad (2.6)$$

where $\beta_L^{(j)}$ denotes the output weight connecting the L -th hidden node and the j -th output node, and $e_L^{(j)} \equiv f^{(j)} - f_L^{(j)}$ is the residual error function of the j -th output node with L hidden nodes where $j = 1, \dots, m$, $f \in \mathcal{L}^2(\mathbb{X})$ is the target function, $f_L = [f_L^{(1)}, \dots, f_L^{(m)}]^T$ is the output function where m is the number of output nodes.

Theorem 2.1.8 is proved to be functional on generalized SLFNs, such as sigmoid network, trigonometric network, RBF network, threshold network, fully complex neural network, etc [28]. Huang [30] further stated that the input parameters for the hidden nodes can be randomly assigned, which is the foundation of ELM.

Theorem 2.1.9. [26, 27] Given $g : \mathcal{R} \rightarrow \mathcal{R}$, $\text{span}\{g(\mathbf{w} \cdot \mathbf{x} + b) : (\mathbf{w}, b) \in \mathcal{R}^n \times \mathcal{R}\}$ is dense in \mathcal{L}^2 , for any continuous target function f and any function sequence g_L randomly generated based on any continuous sampling distribution, $\int_{\mathcal{R}} g(\mathbf{x}) d\mathbf{x} \neq 0$ holds with probability one if the output weights β_j are determined by ordinary least square to minimize $\|f(\mathbf{x}) - \sum_{j=1}^L \beta_j g(\mathbf{w} \cdot \mathbf{x} + b)\|$.

Theorem 2.1.9 proved that ELM can be a universal approximator, whose weights of the hidden layer can be randomly assigned and the output weights can be determined by ordinary least square method [30]. The only requirement is that the activation function g is nonconstant piecewise and $\text{span}\{g(\mathbf{w} \cdot \mathbf{x} + b) : (\mathbf{w}, b) \in \mathcal{R}^n \times \mathcal{R}\}$ is dense in \mathcal{L}^2 .

Theorem 2.1.10. [29] Given any bounded function $G(x)$ in R which has limits and $\lim_{x \rightarrow -\infty} G(x) \neq \lim_{x \rightarrow +\infty} G(x)$, then all linear combinations $\sum_{i=1}^L \beta_i G(a_i, b_i, x)$ are dense in $C(M)$, where M is a compact set of R^n , $\beta_i \in R^1$, $w_i \in R^n$ and $w_i \cdot x$

is the inner product of w_i and x . Then an SLFN with such activation function $G(x)$ and with enough hidden units can separate any arbitrary disjoint regions with any shapes.

Theorem 2.1.11. [29] An SLFN with any continuous bounded non-constant activation function $g(x)$ and with enough hidden units can separate any arbitrary disjoint regions with any shapes.

Theorem 2.1.10 and 2.1.11 state that ELMs can form decision regions of arbitrary shapes. In other word, if there exists sufficient hidden nodes, ELMs are able to approximate almost any complex decision boundary in a classification problem as Fig. 2.2.

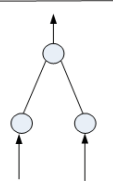
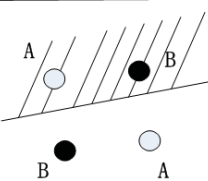
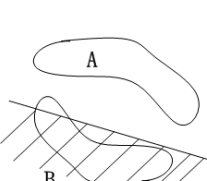
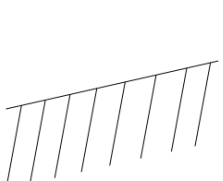
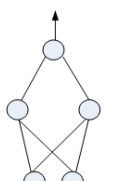
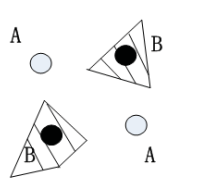
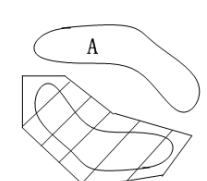
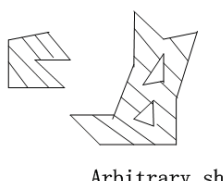
Network	XOR	Classification	Decision region
 <p>No-hidden</p>			 <p>Half-plane</p>
 <p>Single-hidden</p>			 <p>Arbitrary shape</p>

Figure 2.2: Classification ability comparison between two-layers network and three layers network

2.1.3 ELM Algorithm

Training an SLFN is equivalent to find a least-square solution $\hat{\beta}$ of the linear system $\mathbf{H} \cdot \beta = \mathbf{T}$:

$$\|\mathbf{H} \cdot \hat{\beta} - \mathbf{T}\| = \min_{\beta} \|\mathbf{H} \cdot \beta - \mathbf{T}\| \quad (2.7)$$

According to Theorem 2.1.1, if the number of hidden nodes is equal to the number of distinct training samples, the standard SLFNs are proved to be capable of approximating these N samples with zero error. However, the number of hidden nodes is usually much smaller than the number of training samples. In this case, the smallest norm least-square solution of the above linear system can be calculated directly with the *Moore-Penrose generalized inverse* of matrix \mathbf{H} [59, 65]. Thus, ELM can be described as Algorithm 2.1.

Algorithm 2.1 Extreme Learning Machine

Input:Training set $\mathbb{X} = \{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^N$;Activation function $g(\mathbf{x})$;Number of hidden node L .

- 1: Randomly assign input weight \mathbf{w}_j and bias b_j where $j = 1, \dots, L$;
- 2: Calculate the hidden layer output matrix \mathbf{H} by Eq. (2.4);
- 3: Calculate the output weight β by

$$\beta = \mathbf{H}^\dagger \mathbf{T}$$

where \mathbf{H}^\dagger is the Moore-Penrose generalized inverse of \mathbf{H} .

Output:Input weights \mathbf{w}_j and biases b_j , output weight β .

There are various methods to calculate the *Moore-Penrose generalized inverse* of a matrix, such as orthogonal projection method, iterative method and singular value decomposition (SVD). In practice, SVD could be applied in all cases no matter $\mathbf{H}^T \mathbf{H}$ is singular or not.

As concluded in [24], ELM have the following important properties:

- *Smallest training error.* The output of ELM is refers to the solution of

$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$, which is one of the least-squares solutions of a general linear system:

$$\|\mathbf{H}\hat{\beta} - \mathbf{T}\| = \|\mathbf{H}\mathbf{H}^\dagger \mathbf{T} - \mathbf{T}\| = \min_{\beta} \|\mathbf{H}\beta - \mathbf{T}\|. \quad (2.8)$$

- *Smallest norm of weights.* Based on the theory of *Moore-Penrose generalized inverse*, the solution of $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$ is considered to be the minimum norm least-squares solutions of $\mathbf{H}\beta = \mathbf{T}$:

$$\|\hat{\beta}\| = \|\mathbf{H}^\dagger \mathbf{T}\| \leq \|\beta\|, \forall \beta \in \{\beta : \|\mathbf{H}\beta - \mathbf{T}\| \leq \|\mathbf{H}z - \mathbf{T}\|, \forall z \in \mathcal{R}^{L \times N}\}. \quad (2.9)$$

- *Unique solution.* The solution calculated by $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$ is unique, which can be regarded as the optimal solution of the linear system.

The essence of ELM can be summarized as [30]:

- The input weights and biases can be randomly assigned.
- Both the training error $\|\mathbf{H} \cdot \beta - \mathbf{T}\|$ and the norm of weights $\|\beta\|$ of SLFNs are minimized [1].
- The activation function has to satisfy the bounded non-constant piecewise continuous condition (Theorem 2.1.8 and Theorem 2.1.9).

2.2 Fuzzy Measure and Fuzzy Integral

Fuzzy measure is a non-negative and monotonic set function, which can be regarded as a generalization of the classical probability measure by removing additivity

property [53]. It is able to describe three kinds of interactions: positive, negative and additive, which are defined as super-additive, sub-additive and additive measures. This versatility of fuzzy measure leads to numerous related works from both theoretical and practical aspects. For example, fuzzy measure has been successfully applied to multicentric decision making and evaluation of cooperative games. In the former case, fuzzy measure [16] introduces vetoes and favors in models based on the representation ability of dependent data. In cooperative games domain, fuzzy measure represents the strength of player coalitions [51] and effectiveness of unit combination strategies [54]. Other fields related to fuzzy measure include information fusion [2], feature selection [23] and classification based on non-additive and non-linear characteristic [43]. Furthermore, fuzzy integrals are developed with respect to non-additive set functions. Due to the non-linearity, fuzzy integral is proved to be a suitable way of integrating information. For instance, Choquet integral (CI) is a powerful nonlinear aggregation function which has been successfully used in information fusion and data mining [32]. In this section, we will introduce the concepts of most commonly used fuzzy measures and fuzzy integrals.

2.2.1 Fuzzy Measure

Definition 2.2.1. Let (X, F) be a measurable space. A fuzzy measure is a real-valued set function $\mu : F \rightarrow (-\infty, +\infty)$ by satisfying the following criteria:

$$(C1) \mu(\emptyset) = 0;$$

$$(C2) \mu(A) \geq 0 \text{ for every } A \in F;$$

$$(C3) \mu(A) \geq \mu(B) \text{ whenever } A \in F, B \in F, A \subseteq B.$$

Set function μ is called efficiency measure [79] if it only satisfies conditions (C1) and (C2). Furthermore, μ is a signed efficiency measure if it only satisfies condition (C1). Both of them are called non-additive fuzzy measures [79]. Any fuzzy measure satisfies condition (C3) can be regarded as monotonic fuzzy measure.

In [53], Murofushi and Sugeno provided an interesting example to explain the difference between monotonic fuzzy measure and non-monotonic fuzzy measure.

Let X be the set of workers, $A \subset X$ and $B \subset X$ are two different subsets of X . Each group can work in various ways (together or separately). Let $\mu(A)$ and $\mu(B)$ be the numbers of products made by groups A and B in an hour, respectively. In this situation, the productivity of the coupled groups A and B may have several different results.

Suppose we add a monotonic condition to this problem, “every group works in the most efficient way”. In this case, the more workers, the higher the productivity. The most efficient way of working is to kick out the “troublemakers” in each group and make $\mu(A \cup B) \geq \mu(A)$ and $\mu(A \cup B) \geq \mu(B)$.

Non-monotonic condition is to remove the monotonicity assumption. Suppose that the two groups interact with each other, there would be three different situations: 1) if they work separately, their productivity would be $\mu(A \cup B) = \mu(A) + \mu(B)$, which is an additive case; 2) good cooperation of members from A and B yields the productivity of $\mu(A \cup B) \geq \mu(A) + \mu(B)$, which is a super-additive case; 3) bad cooperation of members from A and B yields a productivity of $\mu(A \cup B) \leq \mu(A) + \mu(B)$, which is a sub-additive case.

2.2.2 λ -Fuzzy Measure

When $|X| = n$, there will be 2^n subsets in total. We have to define n singleton sets and the rest subsets can be defined by additive property. In fuzzy measure, we need to define 2^n coefficients, which will be a difficult task when n is large. In order to simplify this case, researchers introduced an additive property called λ law [53].

Theorem 2.2.2. Let $X = \{x_1, x_2, \dots, x_n\}$ be a finite set and $\lambda \in (-1, \infty)$. A normalized set function g_λ defined on 2^X to $[0, 1]$ is called a λ -fuzzy measure on X if for every pair of disjoint subsets A and B of X :

$$g_\lambda(A \cup B) = g_\lambda(A) + g_\lambda(B) + \lambda g_\lambda(A)g_\lambda(B). \quad (2.10)$$

The value of g at a singleton set x_i is called a density and is denoted by $g_i = g(x_i)$. In addition, λ satisfies the following condition.

Theorem 2.2.3. The λ in holomorphic g_λ fuzzy measure is defined by the following equation:

$$\prod_{i=1}^n (1 + \lambda g_i) = 1 + \lambda, \quad (2.11)$$

1. when $\sum_{i=1}^n g_i < 1$, $\lambda > 0$, g_λ is super-additive;
2. when $\sum_{i=1}^n g_i = 1$, $\lambda = 0$, g_λ is additive and become a classical measure;
3. when $\sum_{i=1}^n g_i > 1$, $\lambda > 0$, g_λ is sub-additive.

2.2.3 Choquet Integral

Choquet integral is an expansion of ordinary integral (Lebesgue Integral) and can be regarded as the most natural fuzzy integral.

Theorem 2.2.4. Suppose $X = \{x_1, x_2, \dots, x_n\}$ where $x_1 \leq x_2 \leq \dots \leq x_n$, given a fuzzy measure $\mu(X) : P(X) \rightarrow [0, 1]$ (where $P(X)$ is the power set of X and $\mu(\emptyset) = 0$) and a function $f(x)$, Choquet integral is defined as

$${}^{(c)} \int f(x) \cdot \mu(X) = \sum_{i=1}^n (x_i - x_{i-1}) \cdot \mu(x|f(x) \geq x_i), \quad (2.12)$$

where $x_0 = 0$.

Choquet integral has the following properties:

1. ${}^{(c)} \int 1_A d\mu = \mu(A)$;
2. If μ is a fuzzy measure and $f \leq g$ then ${}^{(c)} \int f d\mu \leq {}^{(c)} \int g d\mu$;
3. ${}^{(c)} \int (af + b) d\mu = a \cdot {}^{(c)} \int f d\mu + b \cdot \mu(X)$;
4. ${}^{(c)} \int (-f) d\mu = -({}^{(c)} \int f d\bar{\mu})$;
5. ${}^{(c)} \int (-f) d\mu = -({}^{(c)} \int f d\mu)$;
6. ${}^{(c)} \int f d\mu = {}^{(c)} \int f^+ d\mu - {}^{(c)} \int f^- d\mu$;
7. ${}^{(c)} \int f d(a \cdot \mu) = a({}^{(c)} \int f d\mu)$;
8. ${}^{(c)} \int f d\mu \leq {}^{(c)} \int f d\nu$;
9. ${}^{(c)} \int f d\mu = {}^{(c)} \int g d\mu$.

2.2.4 Sugeno Integral

Sugeno integral is a special kind of fuzzy integral which is only for the function $h : X \rightarrow [0, 1]$ and normalized fuzzy measures.

Theorem 2.2.5. [53] Suppose μ is a normalized fuzzy measure on X and f is a function on X within the range a_1, a_2, \dots, a_n , where $0 \leq a_1 \leq a_2 \leq \dots \leq a_n \leq 1$. The Sugeno integral is defined as

$$({}_s) \int f(x) \cdot \mu(X) = \bigvee_{i=1}^n [a_i \wedge \mu(\{x | f(x) \geq a_i\})] \quad (2.13)$$

Sugeno integral has the following properties:

1. $(c) \int 1_A d\mu = \mu(A)$;
2. If $f \leq g$ then $({}_s) \int f d\mu \leq ({}_s) \int g d\mu$;
3. $({}_s) \int (a \vee f) \cdot \mu = a \vee (({}_s) \int f \cdot \mu)$
4. $|({}_s) \int f \cdot \mu - (c) \int f d\mu| \leq \frac{1}{4}$
5. If μ is a fuzzy measure in range $[0, 1]$, then $({}_s) \int f \cdot \mu = (c) \int f d\mu$
6. If $\mu \leq \nu$, then $({}_s) \int f \cdot \mu \leq ({}_s) \int f \cdot \nu$
7. $({}_s) \int f \cdot (\mu \vee \nu) = (({}_s) \int f \cdot \mu) \vee (({}_s) \int f \cdot \nu)$
8. If N is a null set and $f(x) = g(x)$ for all $x \notin N$, then $({}_s) \int f \cdot \mu = ({}_s) \int g \cdot \mu$

2.3 Architecture Selection

The most commonly used technique for architecture selection is cross validation (CV), which evaluates the results from a statistical point of view [19]. The main idea of CV is to test the model architectures with different data partitions, estimate the generalization capability, and find out the optimal architecture. Commonly used CV methods are listed as follows:

1. **Leave-p-out CV (LpO CV):** This method adopts p observations as the validation set and the rest observations as the training set. In order to go through all combinations of validation sets and training sets, C_p^n (i.e., n is the number of observations in data set) iterations need to be conducted. Thus, when n is very large, LpO CV may become impractical.
2. **K-fold CV:** This method involves k -fold partition of the given data set, where k is an integer ≥ 2 . A model is constructed based on $k - 1$ subsets (named the training set), and is validated on the remaining subset (named the validation set or testing set). This process is performed k times using different subsets, and the average result is recorded. The k -fold CV is experimentation-driven and is easy to implement, but the time complexity is high.

2.4 Techniques for Handling Large-Scale Data

2.4.1 Discretization of Continuous-Valued Attributes

Discrete and continuous are two classical ordinal data types with orders among the values. Generally, the number of discrete values for an attribute is finite and

sometimes even only a few, while the number of continuous values can be infinitely many. This property of discrete value makes it easier to use and comprehend in data analysis. For example, when a decision tree is induced, the continuous attribute may make the tree reach a *pure* state quickly but with a bad performance (all the instances in a leaf node belong to a specific class) [37], while a discrete attribute may clearly divide a node into several branches. There are many other advantages of using discrete values. For instance, it is mentioned in [64, 69] that discrete attributes have a closer knowledge-level representation than continuous ones. Thus, in many problems, there is a need to make discretizations on continuous-valued attributes. Given a continuous-valued attribute, two important terms should be introduced firstly, i.e., cut-point and arity [48].

- **Cut-point:** This term, also known as split-point, refers to a real value within the range of the given attribute that divides the range into two intervals, one is less than or equal to the cut-point and the other is greater than the cut-point [39]. For instance, a continuous interval $[a, b]$ is split into $[a, c]$ and $(c, b]$, where $c \in (a, b)$ is a cut-point.
- **Arity:** This term represents the number of intervals divided for the attribute. Suppose the arity is \mathbf{d} , then the number of cut-points for this attribute is $\mathbf{d} - 1$. There always exist a trade-off between the arity and its impact on the accuracy [48].

The evaluation of discretization is a complex issue that largely depends on the problem to be solved. Based on [39], three important dimensions are introduced: (1) simplicity: with enough representation ability, the fewer the cut-points, the

better the discretization result; (2) consistency: the inconsistency after discretization should not be much higher than that before discretization (two instances are considered as inconsistent when they have the same attribute values and different class labels); (3) accuracy: a good discretization should be able to reduce the data volume and maintain or even improve the accuracy. Obviously, in order to achieve these objectives, the most important step is the selecting part, i.e., how to evaluate the cut-points and select the best ones.

A typical discretization process on the attribute broadly consists three steps:

- **Sorting:** Sort the values of the given attribute in either descending or ascending order.
- **Selecting:** Get all the available cut-points and select the $\mathbf{d} - 1$ ones using a certain evaluation method.
- **Splitting:** Split the attribute into \mathbf{d} intervals according to the $\mathbf{d} - 1$ selected cut-points.

Two commonly used discretization methods [39] are listed as follows:

- **Equal-width discretization:** This is a simple and commonly used method for discretization. First, the minimum and maximum values of the related attributes are determined. Then the value of different attributes are divided into the predefined number of equal width discrete intervals. However, if the observations is not evenly distributed, then important information may be lost after discretization.

- **Equal-frequency discretization:** Different from equal-width method, the intervals generated with equal-frequency method contain the same number of observations. First, the minimum and maximum values of discretized attributes are determined. Then, all values in the current attribute are sorted in ascending order and divided into a numbers of intervals.

2.4.2 Center and Range

In [11], Diday introduced the concept of symbolic data, which includes intervals, lists, histograms and so on. It is known that, classical data type with single-value data point is easy to represent and analyze. However, symbolic data is usually difficult to analyze by traditional learning techniques. Focus on interval-valued data, there is a need to improve the existing learning technique for further data processing. In what follows, two existing popular methods for interval data are briefly introduced. Other methods could also be found from the literatures [15, 38, 47].

- **Center method (CM):** This method can be regarded as a classical technique to model interval-valued data [3]. The main idea is to build up a regression model using center points of the observed intervals. The model coefficients are applied to the lower and upper bounds of the regressors to predict bounds for the dependent features. In practice, CM could avoid the disturbance of the variation in intervals and is applicable in real-world scenarios.
- **Center and range method (CRM):** Later, Linma and de Carvalho [46] argued that the prediction on interval data will be more accurate when

both the center and range information are used. In CRM, two independent regressors are trained on the center and range values, and the lower and upper bounds are calculated by combining them.

2.4.3 Feature Selection

Feature selection, also called variable elimination, is proved to be effective in reducing computational complexity, understanding data structure and improving the predictor performance [9]. Generally, the feature selection methods can be classified into two categories: filter and wrapper. Filter methods act as preprocessing to rank the features and then selecting the highly ranked ones. The filter methods are computationally light and do not rely on the learning algorithms [9]. Forman developed twelve feature selection metrics for a text classification problem [14]. Besides, a new ranking criterion based on class densities for binary data is introduced in [34], and a ranking principle based on Gram-Schmidt orthogonalization is proposed in [70]. Wrapper methods use the predictor performance to evaluate the variable subsets. A number of search algorithms were developed to find a suboptimal subset of features which gives the highest performance. For instance, a mathematic programming method is presented to minimize a concave function on a polyhedral set [5], and an extra term is used to penalize the size of a subset in order to find the optimal feature subset [4].

The correlation-based feature selection (CFS) [18] is one of the most commonly used methods. This method evaluates the feature subsets by considering the usefulness of individual features for predicting the class labels along with the degree of redundancy among them. It claims that a good feature subset should

contain features highly correlated with the class and uncorrelated with each other.

As a famous multivariate filter, CFS formulates the heuristic as Eq. (2.14):

$$G_s = \frac{\hat{n}\overline{r_{ci}}}{\sqrt{\hat{n} + \hat{n}(\hat{n} + 1)\overline{r_{ii'}}}}, \quad (2.14)$$

where \hat{n} is the number of features in the subset, $\overline{r_{ci}}$ is the mean feature correlation with the class, $\overline{r_{ii'}}$ is the average feature intercolumniation.

Chapter 3

ELM-based Fuzzy Measure Determination

In this chapter, we focus on ELM mapping and modeling of fuzzy measure, and develop the fuzzy ELM (FELM) model for fast fuzzy measure determination. More specifically, we first introduce the concept of set selection, then list several newly developed fuzzy integrals. Afterwards, we propose the detailed structure of FELM. Finally, we conduct some experimental comparisons among GA, PSO, and FELM for fuzzy measure determination. Furthermore, an application of FELM on real time strategy (RTS) game evaluation is also conducted. Five real data sets extracted from games are used to test the effectiveness of FELM.

3.1 Introduction

Due to the highly non-additive and non-linear characteristics of fuzzy measure and fuzzy integral, they are successfully applied in describing the importance of each individual attributes as well as the feature interaction among them [77]. When using fuzzy integral for solving real world problems, the corresponding fuzzy measures should be known in advance. Thus, the determination of fuzzy

measure is the fundamental task in using these methods. However, there are three main difficulties of applying fuzzy measure and fuzzy integral to represent interacting features:

1. **High complexity of fuzzy measure.** Fuzzy measures are defined on the power set of all predictive features. In order to define a fuzzy measure, $2^n - 1$ coefficients need to be determined. This high computational complexity made the task of determining fuzzy measures difficult.
2. **Complex subset selection rule in fuzzy integrals.** A fuzzy integral can be regarded as an integration tool by selecting different fuzzy measure subsets based on certain learning principle. However, the relations among different subsets are usually nonlinear and hard to be represented by traditional polynomial function. There is still a lack of generalized formulation to represent set selection in different fuzzy integrals.
3. **Difficult to select appropriate fuzzy integral.** In general, the interaction among different features are unknown. Thus, it is difficult to choose a suitable fuzzy integral for integrating the importance of these interactions. A fast and generalized fuzzy measure determination method could help user to understand better fuzzy measures and select the appropriate fuzzy model.

Several attempts have been made to reduce the complexity by imposing additional constraints on measures such as k-additive fuzzy measure [16, 51] and *Sugeno* - λ fuzzy measure [71]. However, both of them sacrifice certain degree

of representation ability. Wang et al. [75] proposed a genetic algorithm (GA) based method for fuzzy measure determination. However, the chromosomes are long since there are too many unknown parameters to determine. Later, Grabisch et al. [17] proposed a gradient descent (GD) algorithm to determine the fuzzy measure by solving a quadratic problem, and Wang et al. [73] proposed a neural network (NN) based method for the case that the objective function is not differentiable, e.g., the nonlinear multi-regressions of Choquet integral. These methods have fast convergence but are easy to stuck at local minimum. Thus, it is necessary to develop a new computational technique for determining fuzzy measures with a fast learning speed and a high accuracy.

As discussed in last chapter, ELM has good learning ability, low computational complexity, and is ease for implementation. Recent studies revealed that ELM can be well applied to the areas of pattern searching [8], classification [81], image quality assessment [72], fuzzy rule learning [35], and fuzzy integral determination [76]. How to use FELM to determine signed efficiency measure is explained in this chapter.

The rest of this chapter is organized as follows. Section 3.2 presents the problem statement of determining signed efficiency measure from data. Then a detail explanation of the set selection algorithm in our model is provided in section 3.3. We present our ELM based methodology to determine signed efficiency measures from data in section 3.4. This is followed by some experimental comparisons among GA, PSO and FELM. The result is given in section 3.5. Finally, we conclude this chapter in section 3.6.

3.2 Determining Signed Efficiency Measure from Data

As stated in [73], the fuzzy integral $\int f d\mu$ on $X = \{x_1, x_2, \dots, x_n\}$ can be regarded as a multi-input single output system. The input is a vector of values $f : (f_1, f_2, \dots, f_n)$, and the output is $y = \int f d\mu$. The matrix of input-output data with sample size m is shown as follows:

$$\begin{array}{cccccc}
 x_1 & x_2 & \cdots & x_n & y \\
 f_{11} & f_{12} & \cdots & f_{1n} & y_1 \\
 f_{21} & f_{22} & \cdots & f_{2n} & y_2 \\
 \vdots & \vdots & \ddots & \vdots & \vdots \\
 f_{m1} & f_{m2} & \cdots & f_{mn} & y_m
 \end{array}$$

where f_{ij} is the i -th attribute of source x_j , y_i is the i -th object value.

Now, we want to find a fuzzy measure μ on a measurable space $(X, 2^X)$ such that $y = \int f^{(i)} d\mu, \forall i = 1, 2, \dots, m$, where $f^{(i)}(x_j) = f_{ij}, j = 1, 2, \dots, n, i = 1, 2, \dots, m$. Thus, the problem is transformed to a constrained optimization problem as $e = \sqrt{\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2}$, where \hat{y}_i . A result of $e = 0$ means that a precise solution is found.

3.3 Set Selection Algorithm for Fuzzy Integrals

The set selection in a fuzzy integral is to identify the related fuzzy measure subset for each information source. However, it is complicated and difficult to represent this process as a function. In [73], Wang et al. presented an neural network based learning system for fuzzy measure determination with regard to Choquet integral, and a nonlinear function is provided to represent the set selection process. The value of fuzzy integral is expressed as a linear function, which explains the

relationship between each individual information source and the corresponding fuzzy measure subset.

Actually, the set selection can be considered as finding the corresponding weight for each fuzzy measure subset. In this chapter, each δ_i represents the weight of corresponding fuzzy measure subset, where $i = 1, 2, \dots, N$, N is the number of fuzzy measure subset. The calculation of δ_i contains two steps. The first step is to generate all the subsets of information sources K_i , as well as their corresponding complementary $\overline{K_i}$. The second step is to calculate the value of δ_i for each fuzzy measure subset by some operation between its subset K_i and complementary $\overline{K_i}$.

First, we explain how to generate all the subsets of a given information source. Based on the definition of signed efficiency measure, the combinations of a set of information sources include all the nonempty subsets. Given a set $f = \{f_1, f_2, f_3\}$ with three elements, it could have $2^3 - 1 = 7$ nonempty subsets, which can be represented using 3-digit binary code from 001 to 111 as follows:

$$\begin{array}{ll}
 \beta_N = \mu(\{x_1, x_2, x_3\}), & a = \{1, 2, 3\}, \\
 \beta_1 = \mu(\{x_1\}), & 001 = \{1\}, \\
 \beta_2 = \mu(\{x_2\}), & 010 = \{2\}, \\
 \beta_3 = \mu(\{x_1, x_2\}), & 011 = \{1, 2\}, \\
 \beta_4 = \mu(\{x_3\}), & 100 = \{3\}, \\
 \beta_5 = \mu(\{x_1, x_3\}), & 101 = \{1, 3\}, \\
 \beta_6 = \mu(\{x_2, x_3\}), & 110 = \{2, 3\}, \\
 \beta_7 = \mu(\{x_1, x_2, x_3\}), & 111 = \{1, 2, 3\}.
 \end{array}
 \rightarrow$$

The subsets are generated according to the binary codes of their indices. For example, the index of the first subset is 1, by transforming it to the binary form, we get the binary code 001. Further assume that the first, second and third coding digits are corresponding to elements x_3 , x_2 and x_1 . In this case, each

digit is connected to a particular element in the set, an element is included if the corresponding digit is 1 and not included if it is 0. Subsequently, the first subset is composed of x_1 . Using this representation, the nonempty subsets can be generated by going through all the possible variations, each step output the subset with the elements whose related digits are not zero. To determine the decimal representation of a binary number, we can take the sum of the products of the binary digits and the powers of 2 that they represent. For example, $(1011)_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 11$. Based on above consideration, Wang et. al developed a set selection algorithm to represent this process mathematically [73]. In each step, whether a digit is zero or not can be judged according to the digit weight described in Eq. (3.1), i.e.,

$$\begin{aligned} K_j &= \{k : \frac{j}{2^k} - \lfloor \frac{j}{2^k} \rfloor \geq 0.5, \quad 1 \leq k \leq n\}, j = 1, 2, \dots, N, \\ \overline{K}_j &= \{1, 2, \dots, n\} - K_j, \end{aligned} \quad (3.1)$$

where $\lfloor \frac{j}{2^k} \rfloor$ denotes the integer part of a nonnegative real number $\frac{j}{2^k}$. Then, the set selection in different fuzzy integrals can be easily solve by selecting suitable K_j and \overline{K}_j .

3.3.1 Fuzzy Integrals

Although choquet integral has been widely used for feature interaction representation in many real world applications, there is still a need for new fuzzy integrals with different set selection principles. We developed two new fuzzy integrals, Mean-based Fuzzy integral (Mean-based FI) and Order-based Fuzzy Integral (Order-based FI) are proposed in this part.

Mean-based FI, as given in Definition 3.3.1, is designed by involving all the

fuzzy measures with regard to the current information source. Based on its design principle, the effectiveness of an ensemble should consider all possible combinations that involve the current information source. Since we have no prior knowledge on how each individual interacts with each other, all the combinations are expected to have similar probability to contribute to the ensemble. According to this consideration, a simple averaging mechanism is applied. Mean based FI could give a better accuracy than CI, but suffers from the high time complexity. In this research, we overcome this problem by using it with ELM-based learning.

Definition 3.3.1. Given a fuzzy measure μ on x . The discrete Mean-based Fuzzy Integral of a function $f : X \rightarrow \mathfrak{R}$ can be defined as:

$${}^{(m)} \int f(x) \cdot \mu(X) = \sum_{j=1}^n x_j \cdot \left(\sum_{k=1}^{m_j} \frac{\mu(S_{jl})}{|S_{jl}|} \right), \quad (3.2)$$

where $X = \{x_1, x_2, \dots, x_n\}$, x_j ($j = 1, \dots, n$) is the j -th element in X , m_j is the number of subsets of X that contain x_j , S_{jl} is the l -th subset for x_j , and $|S_{jl}|$ is the number of elements in S_{jl} .

Order-based FI, as given in Definition 3.3.2, is developed by considering the production sequence in many real world applications. In general, it is a reasonable assumption that the resource needed of producing an advanced product should be higher than that of a less advanced one (i.e., the resource weighting for producing a tank should be higher than that for producing a bike). Usually, the production of an advanced product is an important objective in industrial development, which should be considered having more interaction with other byproducts or components. Order-based FI focuses on the information source with the highest

proportion first, and calculates their interactions with all the others. There are two differences between the set selection schemes for CI and Order-based FI. First, the one for Order-based FI does not apply subtraction between two information sources. Second, Order-based FI realizes the set selection process by considering the fact that information source with the highest proportion should dominate the ensemble. In fact, the scheme for Order-based FI is an inverse version of the one for CI.

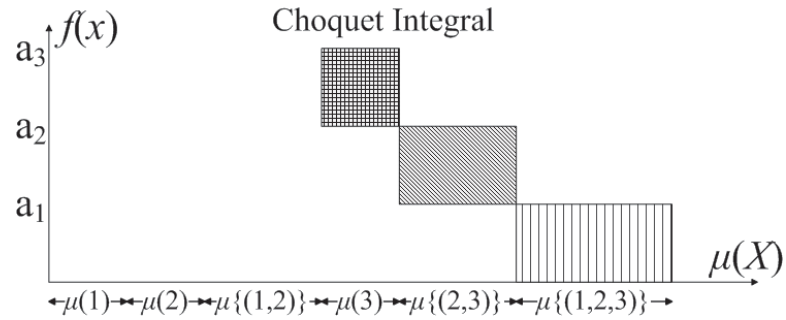
Definition 3.3.2. Given a fuzzy measure μ on x . The discrete Order-based Fuzzy Integral of a function $f : X \rightarrow \mathfrak{R}$ can be defined as:

$$({}_o) \int f(x) \cdot \mu(X) = \sum_{j=1}^n a_j \cdot \mu(x | \{0 < f(x) \leq a_j\}), \quad (3.3)$$

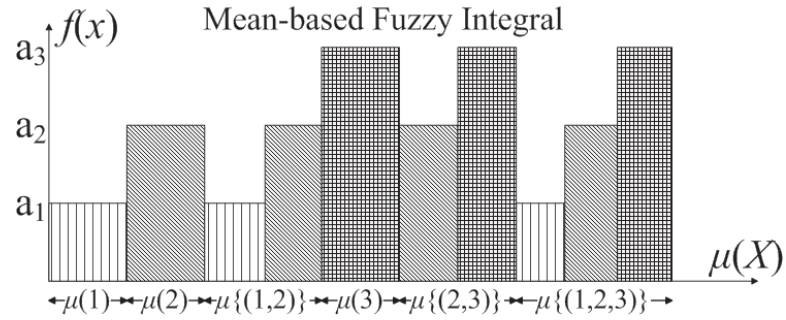
where $X = \{x_1, x_2, \dots, x_n\}$, x_j ($j = 1, \dots, n$) is the j -th element in X , $a_1 \geq \dots \geq a_{n-1} \geq a_n$ is the descending order of $f(x_1), f(x_2), \dots, f(x)_n$.

An illustrative example indicating the differences among CI, Mean-based FI, and Order-based FI is shown in Figure. 3.1. From a geometry point of view, the interactions among different feature can be represented by area. CI has a subtraction on information source and it just considers specific fuzzy measure subsets. Thus, CI's area is the smallest of the three. Order-based FI considers a inverse set selection principle of CI but has no subtraction on information source. It covers a larger area than CI and may has overlap when using efficiency measure. Mean-based FI adopts an average strategy in set selection which covers all the fuzzy measure subsets in calculation. This made it having the largest area of the three.

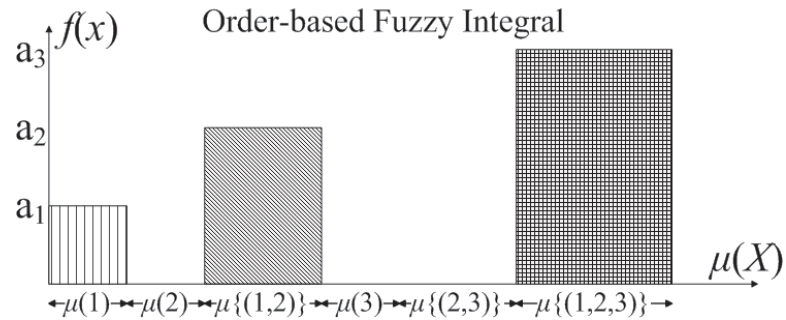
With the power set generated by Eq. 3.3, the set selection scheme for fuzzy integrals (i.e., Sugeno integral, Mean-based fuzzy integral and Order-based fuzzy integral) are listed in Table 3.1.



(a) Choquet Integral



(b) Mean-based Fuzzy Integral



(c) Order-based Fuzzy Integral

Figure 3.1: Comparison among CI, Mean-based FI and Order-based FI.

Table 3.1: Set selection of different fuzzy integrals

Choquet integral	<p>Original definition: $(c) \int f(x) \cdot \mu(X) = \sum_{j=1}^n (a_j - a_{j-1}) \cdot \mu(x f(x) \geq a_j)$ $a_1 \leq \dots \leq a_{n-1} \leq a_n$ is the ascending order of $f(x_1), f(x_2), \dots, f(x_n)$</p> <p>Set selection: $\delta_{ij} = \begin{cases} \min_{k \in K_j} f^{(i)}(x_k) - \max_{k \in K_j} f^{(i)}(x_k), & \min_{k \in K_j} f^{(i)}(x_k) > \max_{k \in \bar{K}_j} f^{(i)}(x_k) \\ 0, & \min_{k \in K_j} f^{(i)}(x_k) \leq \max_{k \in \bar{K}_j} f^{(i)}(x_k) \end{cases}$</p>
Sugeno integral	<p>Original definition: $(s) \int f(x) \cdot \mu(X) = \bigvee_{j=1}^n [a_j \wedge \mu(\{x f(x) \geq a_j\})]$ where $0 \leq a_1 \leq a_2 \leq \dots \leq a_n \leq 1$</p> <p>Set selection: $\delta_{ij} = \begin{cases} \min_{k \in K_j} f^{(i)}(x_k), & \min_{k \in K_j} f^{(i)}(x_k) > \max_{k \in \bar{K}_j} f^{(i)}(x_k) \\ 0, & \min_{k \in K_j} f^{(i)}(x_k) \leq \max_{k \in \bar{K}_j} f^{(i)}(x_k) \end{cases}$</p>
Mean-based FI	<p>Original definition: $(m) \int f(x) \cdot \mu(X) = \sum_{j=1}^n x_j \cdot \left(\sum_{l=1}^{m_j} \frac{\mu(S_{jl})}{ S_{jl} } \right)$ m_j is the number of subsets of X that contain x_j, and S_{jl} is the number of elements in S_{jl}, S_{jl} is the l-th subset for x_j</p> <p>Set selection: $\delta_{ij} = \text{avg}_{k \in K_j} f^{(i)}(x_k)$</p>
Order-based FI	<p>Original definition: $(o) \int f(x) \cdot \mu(X) = \sum_{j=1}^n a_j \cdot \mu(x \{0 < f(x) \leq a_j\})$ $a_1 \geq \dots \geq a_{n-1} \geq a_n$ is the descending order of $f(x_1), f(x_2), \dots, f(x_n)$</p> <p>Set selection: $\delta_{ij} = \begin{cases} \max_{k \in K_j} f^{(i)}(x_k), & \max_{k \in K_j} f^{(i)}(x_k) < \min_{k \in \bar{K}_j} f^{(i)}(x_k) \\ 0, & \max_{k \in K_j} f^{(i)}(x_k) \geq \min_{k \in \bar{K}_j} f^{(i)}(x_k) \end{cases}$</p>

Note that suppose $X = \{x_1, x_2, \dots, x_n\}$, $\mu(X) : P(X) \rightarrow [0, 1]$ is a fuzzy measure (where $P(X)$ is the power set of X and $\mu(\emptyset) = 0$), $f(x)$ is a function.

3.4 Design of ELM

Suppose the values of information sources and the integrals are set as $f(x)$ and d respectively, then with unknown fuzzy measure value, we could represent them by the following linear equations:

$$\begin{bmatrix} g(f_1, \{x_1\}) & g(f_1, \{x_2\}) & g(f_1, \{x_1, x_2\}) & \dots & g(f_1, X) \\ g(f_2, \{x_1\}) & g(f_2, \{x_2\}) & g(f_2, \{x_1, x_2\}) & \dots & g(f_2, X) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g(f_m, \{x_1\}) & g(f_m, \{x_2\}) & g(f_m, \{x_1, x_2\}) & \dots & g(f_m, X) \end{bmatrix} \cdot \begin{bmatrix} \mu(\{x_1\}) \\ \mu(\{x_2\}) \\ \mu(\{x_1, x_2\}) \\ \vdots \\ \mu(X) \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_m \end{bmatrix}.$$

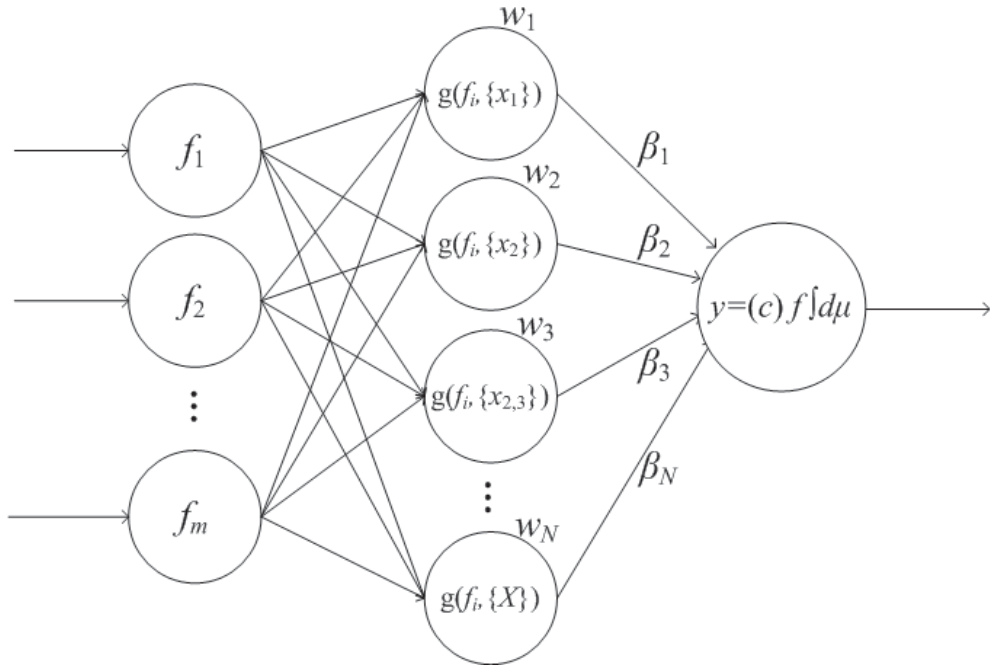


Figure 3.2: ELM-based architecture for describing fuzzy integrals

The $g(f_i, A_j)$ is a function which value is directly related to δ_{ij} , where $f_i = [f_{i1}, f_{i2}, \dots, f_{in}]$ represents the i sample, $i = 1, 2, \dots, m$, m is the number of samples, n is the number of feature. $A = \mu^{(1)}(x_1), \mu^{(2)}(x_2), \mu^{(3)}(x_1, x_2), \dots, \mu^{(2^n-1)}(X)$ represents all the fuzzy measure subsets related to the information source.

Based on the above linear equations, we design an ELM model as shown in Fig. 3.2 to describe different fuzzy integrals. The learning process of our ELM algorithm can be separated into two stages, i.e., set selection stage and learning stage. The complete learning processes are shown as follows:

Stage 1: Set selection

In this stage, the input data is converted a suitable form for ELM based on the selection algorithm developed by [73], as shown in Eq. (3.1). Then, the set selection mechanisms described in Table. 3.1, are used respectively. As a result, the $m \times n$ sample matrix can be converted to a $m \times 2^n - 1$ input matrix as follows:

$$g = \begin{bmatrix} g(f_1, \{x_1\}) & g(f_1, \{x_2\}) & g(f_1, \{x_1x_2\}) & \dots & g(f_1, X) \\ g(f_2, \{x_1\}) & g(f_2, \{x_2\}) & g(f_2, \{x_1x_2\}) & \dots & g(f_2, X) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g(f_m, \{x_1\}) & g(f_m, \{x_2\}) & g(f_m, \{x_1x_2\}) & \dots & g(f_m, X) \end{bmatrix}_{m \times (2^n - 1)} .$$

Stage 2: Learning the fuzzy measure

Step 1: Randomly assign input weight $w_i = \{w_{i1}, w_{i2}, \dots, w_{ij}\}$, where $i = 1, \dots, m$ and $j = 1, \dots, 2^n - 1$;

Step 2: Calculate the hidden layer output matrix H using Eq. (3.4):

$$H = \begin{bmatrix} w_1 \cdot g(f_1, \{x_1\}) & w_2 \cdot g(f_1, \{x_2\}) & w_3 \cdot g(f_1, \{x_1, x_2\}) & \dots & w_N \cdot g(f_1, \{x_A\}) \\ w_1 \cdot g(f_2, \{x_1\}) & w_2 \cdot g(f_2, \{x_2\}) & w_3 \cdot g(f_2, \{x_1, x_2\}) & \dots & w_N \cdot g(f_2, \{x_A\}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_1 \cdot g(f_m, \{x_1\}) & w_2 \cdot g(f_m, \{x_2\}) & w_3 \cdot g(f_m, \{x_1, x_2\}) & \dots & w_N \cdot g(f_m, \{x_A\}) \end{bmatrix}; \tag{3.4}$$

Step 3: Calculate the output weight vector $\beta = H^\dagger T$;

Step 4: The learned fuzzy measure value could be calculated using Eq. (3.5):

$$\begin{bmatrix} \mu(\{x_1\}) \\ \mu(\{x_2\}) \\ \mu(\{x_1, x_2\}) \\ \vdots \\ \mu(X) \end{bmatrix} = \begin{bmatrix} w_1 \cdot \beta_1 \\ w_2 \cdot \beta_2 \\ w_3 \cdot \beta_3 \\ \vdots \\ w_{2^n-1} \cdot \beta_{2^n-1} \end{bmatrix}. \quad (3.5)$$

3.4.1 Characteristics of the FELM

The determination of fuzzy measure by using FELM can be considered as solving a minimum-norm least square error problem on a general linear system $H\beta = T$.

The characteristics of FELM are as follows:

- Universal approximation capability of FELM:** Theoretically, the universal approximation capability of ANN is built on the possibility of increasable and sufficient hidden nodes. FELM cannot fulfill this condition for adopting a hidden layer with fixed number of hidden nodes. Thus, it cannot be regarded as a universal approximator. However, the fuzzy measure determined by FELM is relying on the output weight vector of FELM, which is the minimum-norm least square error solutions of a linear system. In the ideal case, suppose there are N arbitrary distinct and linear independent samples (x_i, y_i) , where $x_i \in R^n$ and $y_i \in R$, if $N = 2^n - 1$ and the hidden layer output matrix H of the SLFN is invertible, then $\|H\beta - T\| = 0$.
- Feature mapping in FELM:** The hidden layer in ANN is serving as a mapping tool for input samples. The addition of hidden neurons map the input data to a higher dimension, which allows the ANN approximating the target function with more coefficients and resulting a better training accuracy. The reason of why FELM maps the input data to a higher dimension is

because the original feature space cannot cover the range of fuzzy measures. From the function approximation point of view, the $2^n - 1$ dimension space may not be the best one to approximate the target function. However, this is the only choice to determine fuzzy measure.

- **Sign of the output weight in SLFN:** It is noteworthy that the output weight vectors of FELM could be negative and can not guarantee of monotonic, which do not satisfy the definition of monotonic fuzzy measure and efficiency measure (as section 2.2.1). Thus, we just focus on signed efficiency measure determination in this research. Moreover, Sugeno integral is defined for the functions included in $[0, 1]$ and the normalized fuzzy measures [53]. Thus, it cannot be applied to the determination of sign efficiency measure from data.

3.5 Experimental Comparisons

In this section, we conduct some experimental comparisons to show the feasibility and effectiveness of our proposed FELM model.

3.5.1 Methods of Comparison

Three learning methods are listed in this section for performance comparison. Each one is used to learn fuzzy measure values by employing different FIs.

Genetic Algorithm (GA): In GA, each chromosome consists $2^n - 1$ fuzzy measures with n attributes. The fitness function is based on the average differences between the real scores and the estimated scores. Roulette wheel selection and one-point crossover are applied for chromosomes evolution. Mutation prob-

ability is set to 0.01 and the population of each generation is set to 50. Detail settings of GA could be found from [45].

Particle Swarm Optimization (PSO): [77] proposed several methods by using PSO to determine fuzzy measure values. Other than the basic PSO, they introduced a new method named generalized PSO (GPSO) by adopting a nonlinear decreasing strategy of inertia weight, sigmoid function and velocity mutation.

Fuzzy Extreme Learning Machine (FELM): Our proposed method is realized.

All the simulations of our proposed methods are performed under MATLAB, and conducted on a computer with an Intel Pentium 4 2.3GHz CPU and 2GB Ram. In each test, 70% cases are selected for training and the rest 30% are used for testing.

3.5.2 Description of Type-II Data

According to the literature, [77] classified the training data into three different types, which are Type-I Data, Type-II Data and Type-III Data. Type-I Data is given by the experts. Considering its high time complexity, this should not be a good way to collect a large dataset. Type-III Data allows users to identify fuzzy measures using a smaller dataset without loss of any information. However, its collection procedure is difficult to understand. Type-II Data can be collected based on some strategies with the predefined fuzzy measure values. The generation process is simple and the collection of large scale data is easy. Therefore, in this experiment, we focus on comparing the performances of different learning algorithms on Type-II data, with the following construction strategies.

Table 3.2: The Original Fuzzy Measure for Constructing Type-II Data

Measure	Value	Measure	Value
$\mu(x_1)$	0.040930607	$\mu(x_2, x_4)$	-0.668111735
$\mu(x_2)$	-0.508966883	$\mu(x_3, x_4)$	-0.725450167
$\mu(x_3)$	0.273797321	$\mu(x_1, x_2, x_3)$	0.094048031
$\mu(x_4)$	-0.848041926	$\mu(x_1, x_2, x_4)$	0.602738412
$\mu(x_1, x_2)$	0.873263465	$\mu(x_1, x_3, x_4)$	-0.226069563
$\mu(x_1, x_3)$	0.672351011	$\mu(x_2, x_3, x_4)$	0.246414918
$\mu(x_1, x_4)$	-0.045030226	$\mu(x_1, x_2, x_3, x_4)$	0.963704525
$\mu(x_2, x_3)$	0.260854265		

Strategy I:

1. Generate a fuzzy measure as shown in Table 3.2.
2. Construct the information sources following the uniform distribution, which are between $[0, 1]$.
3. Calculate the fuzzy integral values according the related fuzzy measures and information sources.

Strategy II:

1. Generate a fuzzy measure as shown in Table 3.2.
2. Construct the information sources with a generator which randomly selects value from set $0, 0.5, 1$.
3. Calculate the fuzzy integral values as output.

We construct three datasets based on strategy I and another three datasets based on strategy II with different sizes. The sizes of the six datasets are 32, 120, 300, 152, 452, and 1200, respectively. For convenience, we only list the detailed information for dataset 2 as in Table 3.3.

Table 3.3: Detailed Information of Data Set 2

i	f_{i1}	f_{i2}	f_{i3}	f_{i4}	y_i
1	9.14E-01	2.19E-01	4.62E-01	5.74E-01	9.54E-02
2	4.95E-01	1.99E-02	1.64E-01	7.85E-01	-1.59E-01
3	3.98E-01	2.13E-01	9.47E-01	6.68E-01	8.98E-02
4	6.37E-03	4.47E-01	1.91E-01	6.34E-01	-8.50E-02
5	6.64E-01	2.51E-01	5.06E-01	1.66E-01	3.84E-01
6	7.43E-01	7.50E-01	8.88E-01	2.40E-01	9.55E-02
7	3.81E-01	8.79E-01	8.30E-01	3.62E-01	6.26E-01
8	8.87E-01	5.97E-01	5.15E-01	7.70E-01	3.93E-01

Furthermore, two methods presented by [77] are adopted to add perturbations in order to generate noisy data. More specifically, a set of random variables normally distributed in interval $[-0.5\rho, 0.5\rho]$ are generated, where ρ represents the diverse strength, and these random perturbations are added to the data generated by strategy I; besides, Gaussian noise with variance σ^2 is directly added to the data generated by strategy II. Based on these methods, eight noisy datasets with the same size of 100 are generated. Three datasets are generated by strategy I, where the value of ρ is set as 0, 0.0001 and 0.001 respectively. The other five datasets are generated by strategy II, where the value of σ^2 is set as 0, 0.00096, 0.00125, 0.00625 and 0.0125 respectively. The information sources are set according to Table 3.2, and the related FIs are used to calculate the output values. These datasets are used to test whether the actual outputs of our proposed FELM algorithm is capable to maintain a certain level of accuracy when noise exists.

3.5.3 Performance Comparison on Type-II Data

From Figures 3.3 and 3.4, we have the following observations.

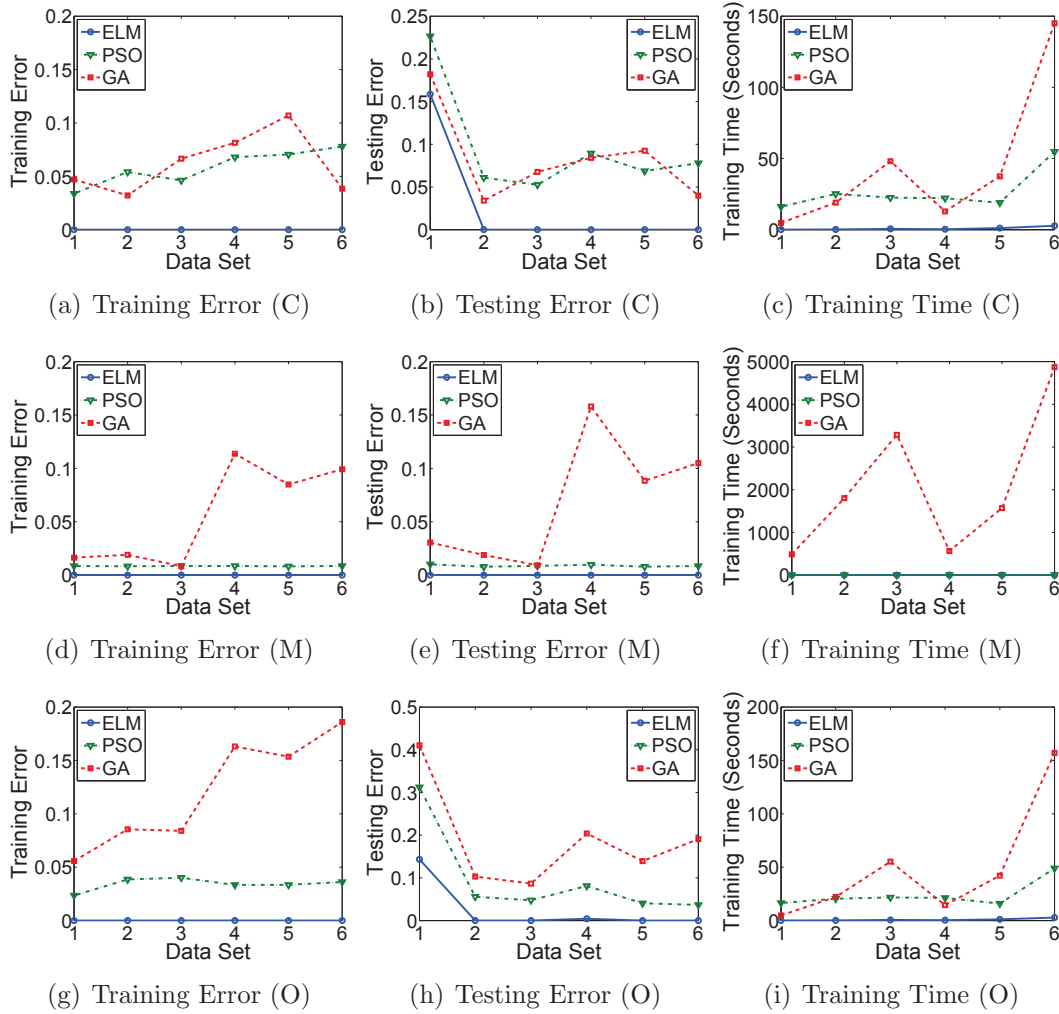


Figure 3.3: Performance Comparison of Different Methods on Type-II data sets: C=CI, M=Mean-based FI, and O=Order-based FI

- FELM demonstrates an obvious advantage in training speed. Its training time is about 200 times faster than GA and PSO. Besides, it seems that GA spends a lot of time in training Mean-based FI. This is because GA needs to traverse all the related fuzzy measure subsets and take average in each generation, which is a time consuming process. The ELM model with set selection algorithm is able to overcome the high complexity problem of random search-based learning techniques in fuzzy measure determination.

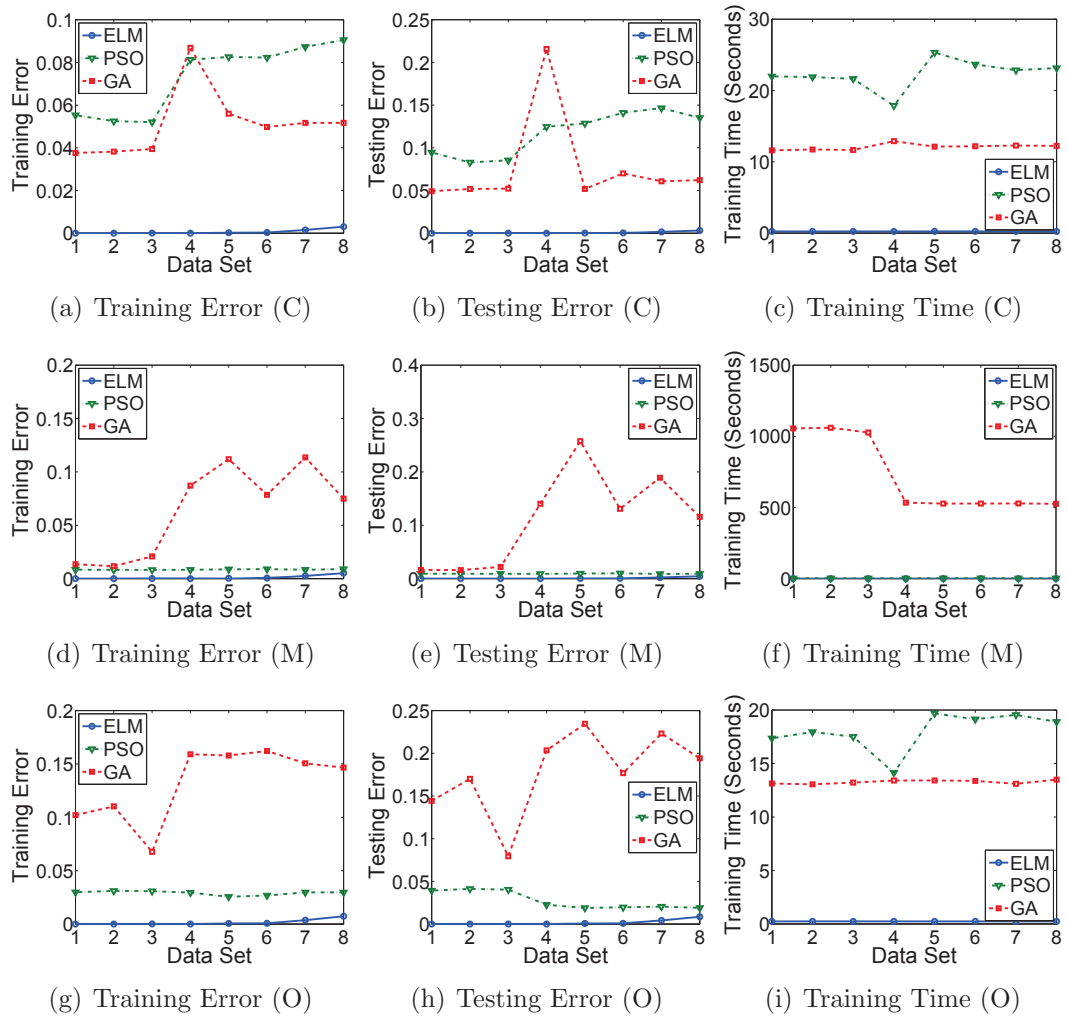


Figure 3.4: Performance Comparison of Different Methods on the Noisy data sets: C=CI, M=Mean-based FI, and O=Order-based FI

- It can be seen from Figure 3.3 that the training and testing errors of FELM are very close to zero. As there are no perturbation in data sets 1 to 6, FELM can guarantee the success of finding the minimum norm least-square solution of SLFNs. On the other hand, the presence of local minima may decrease the accuracy of GA and PSO.
- Although some literatures state that ELM model may be sensitive to the perturbations in data, we found no strong evidence on this statement in our

experiments. FELM still achieve the best training and testing accuracies on all 8 noisy data sets with fast speed. PSO is in general more stable with noisy data and also has an acceptable training speed, while the performance of GA is bad with a very slow training speed. The perturbations in data and the complex set selection of FIs may make it stuck to the local minima.

3.5.4 Description of Warcraft III Data

In our previous research, [44] have successfully applied fuzzy measure and Fuzzy Integral (FI) to describe the uncertain information in the evaluation of computer game unit combination strategy in RTS game. Unit refers to some game characters, resources and playable elements, such as soldiers, horses, machines, castles and the like. A strategy in game is referred to as arranging the army with appropriate unit combination, which is able to gain massive destroy power against opponent enemy. The definition of a *strategy case* is given as

$$\textit{Strategy Case} = \{\textit{Goal}, \textit{Situation}, \textit{Score}\},$$

where “Goal” is the creation of suitable unit combination with certain proportion, e.g. 10% peasants, 30% footman and 60% rifleman; “Situation” is defined as what circumstance the player is dealing with; and “Score” is the evaluation point provided by the game system. Usually, strategy cases are clustered into several groups based on different races and unit combinations, and “Score” is used in learning the fuzzy measures.

In this experiment, Warcraft III is used as our testing platform, which is a famous RTS games with over 7 million copies sold. We collected 2,649 game files

Table 3.4: Nature of Testing Data Set

Data Set	1	2	3	4	5
Player race	Undead	Undead	Orc	Orc	Elf
Enemy unit	Fm, P	Dr, Fm, P	Pr, So, Sb, P	Fm, P	Fm, P
No. of Case	162	65	1004	549	869
No. of Combination	23	16	60	31	31

of professional one-versus-one competitions from the internet, with three types of data: (1) player unit production statistics, (2) enemy unit type, and (3) performance scores. The player unit production statistics involves the number of different unit types, which are used as the values of information sources. Performance scores is considered as the real fuzzy integral value. Five datasets based on different enemy unit types are collected as shown in Table 3.4.

3.5.5 Performance Comparison on Warcraft III Data

The training time, training error, and testing error are shown in Figure 3.5. From the results, we have the following observations.

- FELM achieves a faster learning speed on most of the data sets except data set 3. Its training time is in the range of 1 to 30 seconds. In this case, FELM can meet the requirement for real-time evaluation of game strategies.
- In general, the training and testing errors of FELM, GA and PSO are similar. Focusing on data set 3, we could see that GA and PSO get a bad result with CI. Data set 3 not only has the largest number of instances (i.e. 1004 instances) but also has the biggest variety of unit types (i.e., 11 kinds of unit types). We can concluded that having many unit types increase the difficulty in fuzzy measure determination. Although FELM is

Table 3.5: Fuzzy Measure Subsets ($n \leq 2$) in Data Set 1

Fuzzy Measure	CI	Mean-based FI	Order-based FI
{1}	-2.5510E-10	-5.1177E-02	1.9448E-08
{2}	4.8580E-01	2.5587E-02	-4.1983E-09
{3}	6.3657E-01	-1.7308E-02	-7.3512E-10
{4}	5.4350E-01	2.7944E-02	2.8518E-10
{5}	3.5784E-01	-1.4135E-02	2.6719E-09
{6}	8.9122E-01	3.0005E-02	6.3940E-10
{7}	1.3612E+00	1.9998E-03	-1.5170E-09
{8}	4.3797E-01	5.2121E-02	-7.6184E-10
{1,2}	-9.6410E-11	-5.7313E-04	-1.1408E-12
{1,3}	1.2293E+00	4.2360E-02	-2.7748E-09
{2,3}	1.5141E+00	8.4990E-03	2.4204E-09
{1,4}	4.8388E-10	4.5402E-02	1.1167E-09
{2,4}	-6.3387E-11	9.6199E-03	-5.2911E+00
{3,4}	6.0379E-11	4.0877E-02	5.5367E-10
{1,5}	-2.9677E-10	1.7695E-02	1.5606E-10
{2,5}	1.0689E+00	2.1717E-04	-2.0453E-09
{3,5}	-1.0553E-09	-2.5216E-02	2.3160E-09
{1,6}	1.0691E+00	1.6874E-02	-2.8561E-09
{2,6}	5.6947E-11	-6.5893E-03	-4.8743E-10
{3,6}	-5.2444E+00	1.4251E-02	2.5410E-09
{4,6}	4.4615E-11	-9.6180E-03	-1.3406E-10
{1,7}	-4.0740E-02	2.7849E-02	3.7575E-11
{4,7}	3.9682E+00	-1.6872E-05	1.0694E-10
{1,8}	-2.1809E-10	2.4679E-02	-5.6514E-11
{2,8}	-1.6716E-10	-2.8243E-03	1.8370E+00
{4,8}	2.1075E+00	2.6293E-02	-5.5740E-11

Note: The fuzzy measure of which combination contains positive interaction (i.e., $\mu\{x_1, x_2\} \geq \mu\{x_1\} + \mu\{x_2\}$) is in bold face.

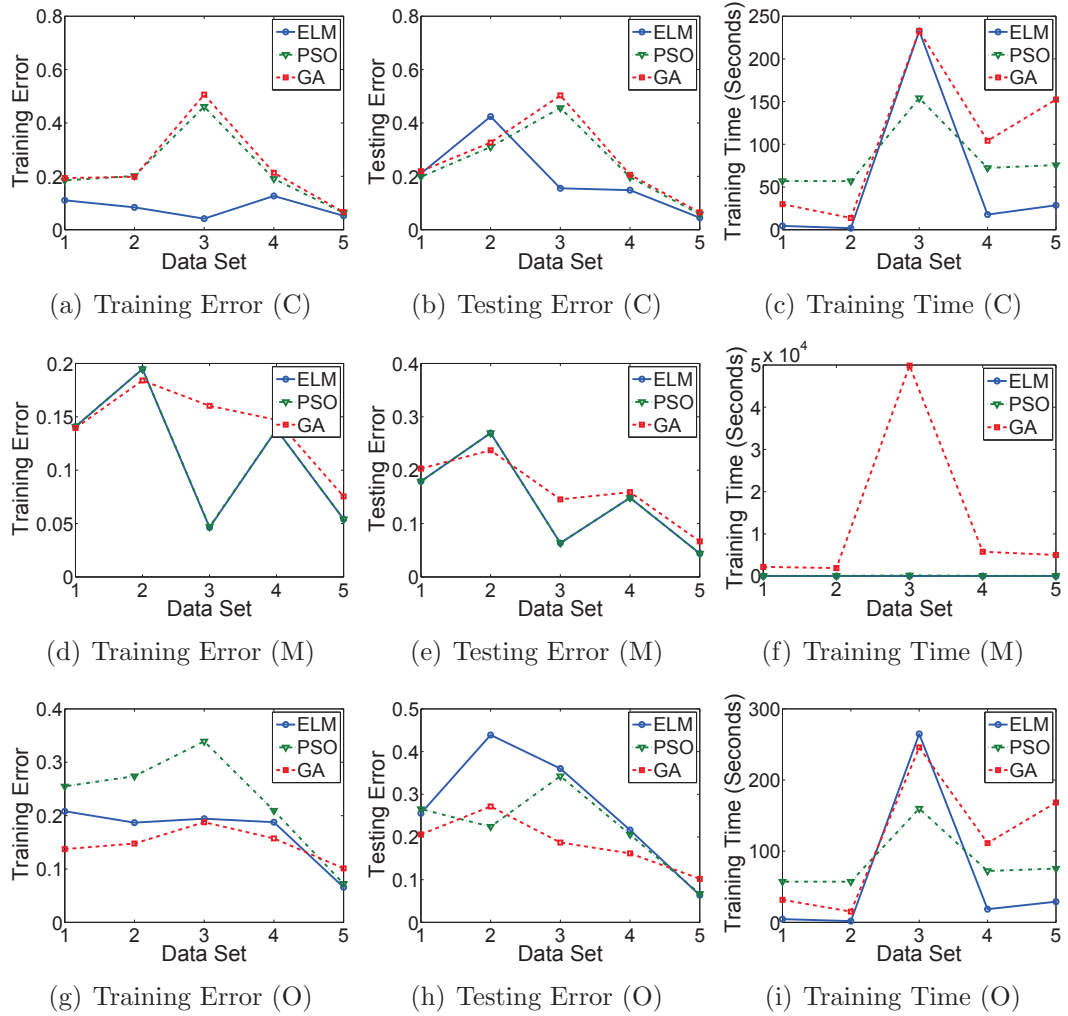


Figure 3.5: Performance Comparison of Different Methods on Warcraft III data sets: C=CI, M=Mean-based FI, and O=Order-based FI

able to maintain the performance on this data sets, the training time is not satisfactory. This is because the over many feature in samples highly increase the dimension of fuzzy measure. In this case, the time complexity in calculating the pseudo-inverse of the hidden layer output matrix is highly increased. How to deal with the massive data with high dimensionality in ELM training is still a challenge problem.

- In our previous research, we found that Mean-based FI could achieve a higher accuracy than CI and Order-based FI but is time consuming using GA. The newly designed set selection scheme for Mean-based FI in FELM significantly reduces the complexity. The experimental result shows that the combination of Mean-based FI and ELM not only reaches the best accuracy but also achieves the fastest learning speed.
- In order to demonstrate the usefulness of fuzzy measure in feature interaction description, we list some learned fuzzy measure subsets of data set 1 in Table 3.5. For convenience, we only list the subsets with no more than 2 elements. By using fuzzy measure, we could easily find out the relation among different unit types. For example, the fuzzy measure values for unit types 2 and 3 are $4.8580E - 01$ and $6.3657E - 01$ respectively, while the fuzzy measure value for their combination is $1.5141E + 00$, which is larger than the summation of their individual values. In this case, we could conclude that unit types 2 and 3 have a positive interaction.

3.6 Conclusion

In this chapter, we have presented a fast learning method for fuzzy measure determination named FELM, and have tested using some data sets including the unit combination strategy evaluation in RTS game. Different from GA and PSO, FELM has several special features.

- It has a very fast learning speed, which has been shown in the experiment. Since the efficiencies of the classical learning techniques, such as GA and

PSO, are usually low, FELM appears to be a suitable choice for real-time evaluation of applications of such as unit combination strategies in RTS game.

- Many classical learning algorithms usually have the local minima and over fitting problems. Overcoming these problems may further increase the time complexity of the model. FELM can avoid such problems effectively with a simple learning structure, which requires no parameter to be tuned.
- The proposed FELM performs better than GA and PSO in most cases. Besides, there is no strong evidence to indicate that FELM is sensitive to noisy data.

Chapter 4

Architecture Selection for SLFNs with MCDM Model

This chapter presents a novel architecture selection method for ELM. It is able to estimate the generalization ability of the selected architecture with input data perturbations. Some fundamental concepts, such as multi-criteria decision making (MCDM) and localized generalized error model (LGEM) will be introduced, and a LGEM-based architecture selection method with MCDM model is developed.

4.1 Introduction

As discussed in [40,84], the number of hidden nodes will affect the generalization ability of SLFNs. Thus, architecture selection is a necessary step before training the final model. The commonly used cross validation (CV) method is effective but it is time consuming because of too many iteration in estimating the best model structure. Moreover, it focuses on estimating the expected generalization error instead of its bound [84], which means that it cannot guarantee a good generalization ability of the selected classifier. On the other hand, error bound

models attempt to find an upper bound of the error between the target function and the trained model in the entire input space [49, 55, 57, 83]. The advantage of using these models is that the estimated error bound is effective for all unseen samples.

In [84], Yeung et al. proposed a localized generalization error model (LGEM) for architecture selection for radial basis function neural networks (RBFNNs). It bounds the generalization error for unseen samples located within a predefined neighborhood of the training samples. Later, Wang et al. [81] proposed an extended LGEM for ELM with sigmoid activation function. The error bound generated by LGEM is an upper bound of the mean square error (MSE) for unseen samples within a certain area. Theoretically, this bound is better than the error bound that is determined without considering the statistical characteristics of the training samples.

In this part, the architecture selection process is considered as a decision making problem, which evaluates a set of alternative architectures and selects the best. MCDM [67] is a widely known branch of decision making, which evaluates the alternatives on the basis of multiple criteria. Typical tasks of MCDM include sorting the alternatives into a preference preorder and selecting the best alternative from a set of candidates. With some predefined assumptions, the decision maker is required to indicate the priority among several alternatives with respect to each criteria [74]. MCDM has been widely used in various domains, such as knowledge discovery, preference modeling, and multi-objective optimization [10, 12, 13].

In this chapter, an LGEM based architecture selection method for ELM with

MCDM system is proposed. Two criteria, i.e. the training accuracy of an classifier and the estimated Q -value from LGEM, are selected to evaluate the ELM architecture. By investigating the priority of architectures, a preference preorder for each criteria could be determined. Then, the dominated indices and dominating indices of an architecture in the preference preorders are computed to reflect its informativeness. This strategy is likely to provide accurate priority information, since the preference preorders are evaluated uniformly and independently. Finally, the architecture with the maximum value of informativeness is selected.

The rest of this chapter is organized as follows: Section 4.2 lists some background knowledge on LGEM and MCDM system. In Section 4.3, the LGEM based architecture selection with MCDM system for SLFNs trained with ELM is derived in detail. In Section 4.4, extensive experiments are conducted to show the effectiveness of the proposed model. Section 4.5 concludes this chapter.

4.2 Localized Generalization Error Model

In this section, we will present some background knowledge generalization error and LGEM.

4.2.1 Generalization Error of Classifier

Classification problem can be regarded as building a classifier f_θ to simulate the unknown input-output mapping function f , where θ is a set of parameters selected from a specific domain [84]. Given an input space \mathcal{T} , the generalization error for unseen samples in \mathcal{T} is defined as:

$$R_{true} = \int_{\mathcal{T}} (f_{\theta}(\mathbf{x}) - f(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x}, \quad (4.1)$$

where \mathbf{x} is the input sample, and $p(\mathbf{x})$ is the true unknown probability density function of \mathbf{x} .

Given a training set \mathbb{X} with N distinct samples, i.e., $\mathbb{X} = \{(\mathbf{x}_b, f(\mathbf{x}_b))\}_{b=1}^N$, the empirical error R_{emp} of classifier f_{θ} over \mathbb{X} can be defined as:

$$R_{emp} = \frac{1}{N} \sum_{b=1}^N (f_{\theta}(\mathbf{x}_b) - f(\mathbf{x}_b))^2. \quad (4.2)$$

The objective is to find a f_{θ} that can correctly classify unseen samples and minimize the generalization error. Since the target output and the distribution of the unseen samples are unknown, the generalization error cannot be calculated directly.

4.2.2 LGEM

The LGEM was proposed by Yeung et al. [84], and is used to measure the generalization error for unseen samples within a predefined neighborhood (i.e., Q -neighborhood) of the training data.

For every training sample $\mathbf{x}_b \in \mathbb{X}$, suppose there exists a set of samples with each sample \mathbf{x} fulfills $0 < |\Delta x_i| < Q, \forall i = 1, \dots, n$, where n is the number of input features, $\Delta \mathbf{x} = [\Delta x_1, \dots, \Delta x_n]^T = \mathbf{x} - \mathbf{x}_b$, and Q is a given value. In classification problem, each unseen sample has the same chance to appear since one usually has no prior knowledge about the distribution of the true input space. Thus, $\Delta \mathbf{x}$ can be considered as the input perturbation randomly chosen from a uniform distribution with zero mean. The Q -neighborhood of training sample \mathbf{x}_b is defined

as:

$$S_Q(\mathbf{x}_b) = \{\mathbf{x} | \mathbf{x} = \mathbf{x}_b + \Delta\mathbf{x}; 0 < |\Delta x_i| \leq Q, \forall i = 1, \dots, n\}. \quad (4.3)$$

All the samples in $S_Q(\mathbf{x}_b)$ are considered as unseen samples except \mathbf{x}_b . Let S_Q be the union of all $S_Q(\mathbf{x}_b)$, which is named Q -union. For $0 \leq Q_1 \leq \dots \leq Q_k \leq \infty$, the following relationship holds:

$$\mathbb{X} \subseteq S_{Q_1} \subseteq \dots \subseteq S_{Q_k} \subseteq \mathcal{T}. \quad (4.4)$$

It is noteworthy that the Q -neighborhood could be a hypercube or other shapes.

In LGEM, the stochastic sensitivity measure (ST-SM) is used to measure the output perturbations (Δy) of the classifier with unseen samples. In general, the unseen samples could be selected from various distributions. Moreover, the unseen samples located far away from the training samples have no effect on the learning model. Thus, we ignore the generalization for unseen samples that are out of S_Q . According to Eq. (4.1), the localized generalization error is defined as:

$$R_{SM}(Q) = \int_{S_Q} (f_\theta(\mathbf{x}) - f(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x}. \quad (4.5)$$

By the Hoeffding's inequality [21], with a probability of $1 - \eta$, we have:

$$\begin{aligned} R_{SM}(Q) &= \int_{S_Q} (f_\theta(\mathbf{x}) - f(\mathbf{x}))^2 p(\mathbf{x}) d\mathbf{x} \\ &\leq \left(\sqrt{E_{S_Q}((\Delta y)^2)} + \sqrt{R_{emp}} + A \right)^2 + \varepsilon \\ &= R_{SM}^*(Q), \end{aligned} \quad (4.6)$$

where $\Delta y = f_\theta(\mathbf{x}) - f_\theta(\mathbf{x}_b)$, $\varepsilon = B\sqrt{\ln \eta / (-2N)}$, $R_{emp} = \frac{1}{N} \sum_{b=1}^N (f_\theta(\mathbf{x}_b) - f(\mathbf{x}_b))^2$, $E_{S_Q}((\Delta y)^2)$ denotes the stochastic sensitivity measure (SSM), A , B and η are the

difference between the maximum and minimum value of the target outputs, the possible maximum value of the MSE and the confidence of the bound, respectively. With a given data set, A and B can be fixed. In this case, the upper bound of $R_{SM}(Q)$ is defined as $R_{SM}^*(Q)$.

Based on the definitions in [56,78], the SSM measures the output perturbation of the classifier when the input value changes. With no prior knowledge about the distribution of unseen data, Wang et al. [81] assume that the training samples are all independent. Moreover, the input perturbation of the i -th input feature is a random variable with a uniform distribution having the mean $\mu_{\Delta x_i} = 0$ and the variance $\sigma_{\Delta x_i}^2 = (2Q)^2/12 = Q^2/3$. According to the basic ELM structure, SLFNs with sigmoid activation function could be described as

$$\begin{aligned} f_{\theta}(\mathbf{x}) &= \sum_{j=1}^M \beta_j g(\mathbf{w}_j \cdot \mathbf{x} + b_j) \\ &= \sum_{j=1}^M \frac{\beta_j}{1 + \exp(-(\mathbf{w}_j \cdot \mathbf{x} + b_j))}, \end{aligned} \quad (4.7)$$

where M is the number of hidden nodes.

According to Taylor's series expansion and central limit theorem, Wang et al. [81] has the derivation of the SSM for SLFNs with sigmoid activation function as:

$$E_{S_Q}((\Delta y)^2) = \frac{Q^2}{3} \sum_{j=1}^M \beta_j^2 \sum_{k=1}^n w_{jk}^2, \quad (4.8)$$

where Q is a given real value, β_j is the output weight for the j -th hidden node, and w_{jk} is the weight connecting the j -th hidden node and the k -th input node.

The SSM evaluates the sensitivity of the network output to the change of input. Theoretically, a classifier yielding good generalization capability should

be able to minimize both the training error and the SSM or achieve a good balance between them [81]. Here, Eq. (4.8) measures the output fluctuations of the classifier. A classifier that has high output fluctuation yields high SSM because its output value changes dramatically when the input value changes. Thus, we should try to minimize it when constructing a neural network.

4.2.3 Architecture Selection Based on LGEM for ELM

In order to compare the performance of two classifiers, we can either compare their Q values by fixing $R_{SM}^*(Q)$, or compare their $R_{SM}^*(Q)$ values by fixing Q [84]. Considering the first method, suppose two classifiers f_1 and f_2 have the same value of $R_{SM}^*(Q)$, i.e., $R_{SM}^*(Q_1) = R_{SM}^*(Q_2) = a$. Obviously, the one with larger Q value has a better generalization performance, since it covers more unseen samples with the same error bound. In order to get the best network structure, we can select the largest Q that satisfies $R_{SM}^*(Q) = a$

Based on the above statements, Wang et al. [81] formulate the architecture selection problem as an optimization problem:

$$\max_Q R_{SM}^*(Q) \leq a. \quad (4.9)$$

Based on Eqs. (4.6) and (4.8), the R_{SM}^* for SLFNs with sigmoid activation function is described as:

$$R_{SM}^* \approx \left(\sqrt{\frac{Q^2}{3} \sum_{j=1}^M \beta_j^2 \sum_{k=1}^n w_{jk}^2} + \sqrt{R_{emp}} + A \right)^2 + \varepsilon. \quad (4.10)$$

Let $R_{SM}^*(Q) = a$, we have

Algorithm 4.1 Architecture Selection Based on LGEM for ELM (ASLGEM-ELM)

Input:

- Training set \mathbb{X} ;
- A set of available model parameters $\mathbb{M} = \{M_1, \dots, M_n\}$.
- 1: **for** each available model parameter $M_i (i = 1, \dots, n)$ **do**
- 2: Train an ELM model based on \mathbb{X} ;
- 3: Get the Q -value using Eq. (4.11), denoted as Q_i ;
- 4: Calculate $h(M_i, Q_i)$ based on Eq. (4.12);
- 5: **end for**
- 6: Select the model parameter M^* with the maximum value of $h(M_i, Q_i)$, i.e.
 $M^* = \operatorname{argmax}_{M_i \in \mathbb{M}} h(M_i, Q_i)$;

Output:

Selected model parameter M^* .

$$Q^2 \sum_{j=1}^M \beta_j^2 \sum_{k=1}^n w_{jk}^2 - 3(\sqrt{a - \varepsilon} - \sqrt{R_{emp} - A})^2 = 0. \quad (4.11)$$

Eq. (4.11) is a quadratic equation for Q which has two solutions. Let Q^* be the positive real solution of the quadratic equation, then we have:

$$h(M, Q^*) = \begin{cases} 0, & R_{emp} \geq a \\ Q^*, & \text{else} \end{cases}. \quad (4.12)$$

Finally, the architecture selection algorithm based on LGEM for ELM (ASLGEM-ELM) is described as Algorithm 4.1.

4.3 LGEM Based Architecture Selection with MCDM Model

In this section, we will present a LGEM-based architecture selection model with MCDM model.

4.3.1 Trade-off Between Training Accuracy and Q -Neighborhood

In practice, an ideal classifier is supposed to yield a good generalization ability on both training samples and unseen samples [84]. A higher training accuracy represents a better performance on the training samples, and a larger Q -value represents a better generalization performance on unseen samples. Unfortunately, there is a conflict between these two criteria. A higher training accuracy will lead to a smaller Q -value. In order to select a better network structure, it is necessary to achieve a balance between these two terms. Thus, a MCDM model is used to achieve a balance between Q -neighbourhood and training accuracy to perform architecture selection for ELM. The reasons are listed as follows.

1. **Importance of training accuracy:** LGEM adopts the training error as a key component to model the generalization ability of an classifier. The main consideration is that a classifier could be constructed by minimizing the classification error over the training set. Training accuracy is an important criteria to estimate the effectiveness of a classifier. It directly affects the empirical error of a classifier.
2. **Tradeoff between training accuracy and Q -value:** A larger Q -value means that more unseen samples have been included into the localized generalization error bound calculation. However, this may result in a decreasing of the training accuracy in practice.

4.3.2 MCDM Model

MCDM is a branch of decision makings. It evaluates a finite number of alternatives based on two or more conflicting criteria. Generally, there are two important phases in a MCDM model: [82]: (1) information input and construction; (2) aggregation and exploitation.

Suppose there are n distinct alternatives M_1, \dots, M_n and m criteria Cr_1, \dots, Cr_m , where $Cr_k(M_i)$ is the level of achievement of $M_i (i = 1, \dots, n)$ with regard to $Cr_k (k = 1, \dots, m)$. In order to select the best one, we have to evaluate the informativeness of these alternatives. In the first phase, the problem can be mathematically modeled as a decision matrix [7, 20], i.e.,

$$\begin{bmatrix} Cr_1(M_1) & Cr_2(M_1) & \dots & Cr_m(M_1) \\ Cr_1(M_2) & Cr_2(M_2) & \dots & Cr_m(M_2) \\ \vdots & \vdots & \ddots & \vdots \\ Cr_1(M_n) & Cr_2(M_n) & \dots & Cr_m(M_n) \end{bmatrix}.$$

In the second phase, the system will aggregate the elements in the decision matrix, and exploit the preference relations of the alternatives. For preference modeling, the elements of the decision matrix are usually represented by ordinal numbers. As an examples, four relations are defined for a pair of distinct alternatives M_i, M_j :

1. $M_i \succ M_j$: M_i is preferred to M_j ;
2. $M_i \prec M_j$: M_j is preferred to M_i ;
3. $M_i ? M_j$: M_i is incomparable to M_j ;
4. $M_i \approx M_j$: M_i is indifferent to M_j .

Then, the system is required to determine that which of the four relations holds for any M_i, M_j with regard to each criterion Cr_k . After the relations between any two distinct alternatives have been exploited, all the alternatives could be arranged into some preference preorders as defined in Definition 4.3.1.

Definition 4.3.1. (preference preorder) Given a MCDM problem with n distinct alternatives M_1, \dots, M_n and m criteria Cr_1, \dots, Cr_m , where the level of achievement of $M_i (i = 1, \dots, n)$ with regard to $Cr_k (k = 1, \dots, m)$ is denoted by $Cr_k(M_i)$. If for Cr_k there exists $Cr_k(M_1^*) \succ \approx Cr_k(M_2^*) \succ \approx \dots \succ \approx Cr_k(M_n^*)$, where $M_1^* \neq M_2^* \neq \dots \neq M_n^* \in \{M_1, M_2, \dots, M_n\}$, then the order $M_1^*, M_2^*, \dots, M_n^*$ is called a preference preorder of M_1, M_2, \dots, M_n with regard to Cr_k , denoted by \mathcal{P}_k .

Obviously, the decision maker tends to select the anterior alternative in \mathcal{P}_k with regard to Cr_k . In the following, we further introduce some definitions based on preference preorders.

Definition 4.3.2. (k -th dominated index and k -th dominating index) Given that \mathcal{P}_k is a preference preorder of alternatives M_1, \dots, M_n with regard to $Cr_k (k = 1, \dots, m)$, then the k -th dominated index and k -th dominating index of $M_i (i = 1, \dots, n)$, denoted by $\psi_k^\succ(M_i)$ and $\psi_k^\prec(M_i)$, are respectively defined as

$$\psi_k^\succ(M_i) = \sum_{j=1, \dots, n, j \neq i} d(\succ, R_{ij}^{(k)}), \quad (4.13)$$

and

$$\psi_k^\prec(M_i) = \sum_{j=1, \dots, n, j \neq i} d(\prec, R_{ij}^{(k)}). \quad (4.14)$$

where $d(\cdot, \cdot)$ is a distance metric, $R_{ij}^{(k)} \in \{\succ, \prec, \approx, ?\}$ is the relation of M_i and M_j with regard to Cr_k .

Definition 4.3.3. (dominated index and dominating index) Given that $\mathcal{P}_1, \dots, \mathcal{P}_m$ are the preference preorders of alternatives M_1, \dots, M_n with regard to Cr_1, \dots, Cr_m , then the dominated index and dominating index of $M_i (i = 1, \dots, n)$, denoted by $\psi^\succ(M_i)$ and $\psi^\prec(M_i)$, are respectively defined as

$$\phi^\succ(M_i) = \sum_{k=1}^m w_k \psi_k^\succ(M_i), \quad (4.15)$$

and

$$\phi^\prec(M_i) = \sum_{k=1}^m w_k \psi_k^\prec(M_i), \quad (4.16)$$

where w_k is the weight of Cr_k .

The dominated index and dominating index of M_i respectively reflect the degrees of M_i being dominated by others and dominating others in $\mathcal{P}_1, \dots, \mathcal{P}_m$, respectively. In a MCDM problem, an alternative with lower dominated index and higher dominating index is preferred.

It is noteworthy that in order to make Eqs. (4.13)~(4.16) concrete, we have to define the distance metric $d(R, R')$, where $R, R' \in \{\succ, \prec, ?, \approx\}$. As demonstrated in [33, 61], there are several conditions for $d(R, R')$ to be a real-valued metric measure:

1. $d(\succ, ?) = d(\prec, ?)$ and $d(\succ, \approx) = d(\prec, \approx)$.
2. $d(\succ, \approx) + d(\approx, \prec) = d(\succ, \prec)$.

3. $d(\succ, ?) \geq d(\approx, ?)$.
4. $d(\approx, ?) \geq d(\approx, \succ)$.
5. $d(\succ, \prec) = \max\{d(R, R') \mid R, R' \in \{\succ, \prec, \approx, ?\}\}$.
6. $d(R, R') > 0$, if $R \neq R'$ and $d(R, R') = 0$ if $R = R'$.
7. $d(\succ, \prec) - d(\succ, ?) = d(\succ, ?) - d(\approx, ?) = d(\approx, ?) - d(\approx, \succ)$.

Finally, the distances are derived as in Table 4.1, where a is a positive real number.

Table 4.1: Distances between relations.

$d(\cdot, \cdot)$	$M_i \succ M_p$	$M_i \prec M_p$	$M_i ? M_p$	$M_i \approx M_p$
$M_i \succ M_p$	0	$2d$	$(5/3)d$	d
$M_i \prec M_p$	$2d$	0	$(5/3)d$	d
$M_i ? M_p$	$(5/3)d$	$(5/3)d$	0	$(4/3)d$
$M_i \approx M_p$	d	d	$(4/3)d$	0

According to the distance metrics in Table 4.1 and Definitions 4.3.2~4.3.3, the dominated index and dominating index of each alternative could be calculated easily. Since the criteria used in this work are real-valued criteria, the preference preorders determined by these criteria are complete preorders (the relation of ? does not exist). In this case, two issues need to be resolved for each criterion [74]:

1. How to define the relation of \approx .
2. How to design the weight w .

As for the first issue, we design a threshold T_k between the relations \approx and \prec or \succ . If the difference of Cr_k between two alternatives is smaller than T_k , they are taken as indifferent regarding the k -th criterion. As for the second issue, we

Algorithm 4.2 Architecture Selection for ELM with MCDM Model

Input:

- Training set \mathbb{X} ;
- A set of available model parameters $\mathbb{M} = \{M_1, \dots, M_n\}$;
- Control parameter a , threshold parameters T_A and T_Q , and weight parameter w .
- 1: **for** each available model parameter $M_i (i = 1, \dots, n)$ **do**
- 2: Train an ELM model based on \mathbb{X} ;
- 3: Get the training error and Q -value, denoted as R_{emp_i} and Q_i ;
- 4: Let $Q_i = 0$ if $R_{emp_i} \geq a$;
- 5: **end for**
- 6: Based on T_A and T_Q , get the relations between any two ELM models regarding training accuracy and Q -value from $\{\succ, \prec, \approx\}$.
- 7: Fix the preference preorders of $\{M_1, \dots, M_n\}$ regarding training accuracy and Q -value;
- 8: Compute the dominated index and dominating index of each M_i based on Eqs. (4.15) and (4.16), i.e., $\phi^\succ(M_i) = w\psi_A^\succ(M_i) + (1 - w)\psi_Q^\succ(M_i)$ and $\phi^\prec(M_i) = w\psi_A^\prec(M_i) + (1 - w)\psi_Q^\prec(M_i)$;
- 9: Calculate the informativeness of each M_i , i.e., $info(M_i) = \phi^\prec(M_i) - \phi^\succ(M_i)$;
- 10: Select the model parameter M^* with the maximum value of $info(M_i)$, i.e. $M^* = \operatorname{argmax}_{M_i \in \mathbb{M}} info(M_i)$;

Output:

Selected model parameter M^* .

assign the importance of the criteria from a set of feasible weights W such that $w_k \in W$ and $\sum_{k=1}^m w_k = 1$.

4.3.3 Proposed Architecture Selection Algorithm

By integrating the Q -value in Eq. (4.12), the training accuracy of ELM, and the MCDM model described in Section 4.3.2, the LGEM based architecture selection for ELM with MCDM model is described as Algorithm 4.2.

4.4 Experimental Comparisons

In this section, we will conduct some experiments to show the feasibility and effectiveness of the proposed algorithm.

4.4.1 Methods of Comparison

Three methods are listed in this section for performance comparison:

- **ASLGEM-ELM:** This is a method proposed by Wang et al. [81] for ELM architecture selection based on LGEM with sigmoid activation function. Similar to the proposed method, it fixes the value of $R_{SM}^*(Q)$ and to find the maximum Q value. The learning process is described as Algorithm 4.1.
- **Cross Validation (CV):** 2-fold CV, 5-fold CV and 10-fold CV are adopted in the experiments. In a k -fold CV, the data set is divided into k partitions, and k classifiers are trained. In training each classifier, $k - 1$ partitions are taken as the training set, and the rest one partition is selected as the validation set. Finally, the parameter with the lowest CV error is selected.
- **Architecture Selection with MCDM System:** Algorithm 4.2 is used.

4.4.2 Experimental Setting

The three methods are conducted on 16 benchmark classification data sets as shown in Table 5.1. For each data set, 70% data are randomly selected as the training set, and the rest 30% are selected as the testing set. The experiments are conducted 50 trials for each data set. Finally, the average value and standard deviation are recorded. The simulations are carried out under MATLAB 7.9.0

Table 4.2: Selected Datasets for Performance Comparison

Dataset	#Example	#Attribute	#Class	Class Distribution
SPECTF	267	44	2	212/55
Cancer	699	9	2	458/241
Bupa	345	6	2	145/200
German	1000	24	2	700/300
Haberman	306	3	2	225/81
Pima	768	8	2	268/500
Wdbc	569	30	2	212/357
Wpbc	198	33	2	47/151
Chart	600	60	6	100×6
Cotton	356	21	6	55/49/30/118/77/27
Dermatology	366	34	6	112/61/72/49/52/20
Glass	214	10	6	70/76/17/13/9/29
Libras	360	90	15	24×15
Vowel	990	10	11	90×11
Yeast	1484	6	10	244/429/463/44/51/163/35/30/20/5
Soybean	683	35	19	20×9/88/44×2/92/91×2/15/14/16/8

environment in a PC with a 3.16-GHz Intel Core Duo CPU, a 4-GB memory, and 64-bit windows 7 system.

The parameter setting for ASLEGEM-ELM is described as follows. According to Eq. (4.11), the difference between the maximum and minimum values of target output A and the number of training samples N can be obtained once the training data is given. The maximum possible value of the MSE B and the confidence level of the R_{SM}^* can be preselected before training the classifier. Furthermore, a sample with square error larger than 0.25 is considered to be incorrect. Thus, the constant a is set as 0.25.

The proposed method adopts the same setting as ASLGEM-ELM to calculate the Q -value. Moreover, the thresholds for both training accuracy T_A and Q -value T_Q are set as 0.01. The weight for the criteria are tuned as $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Theoretically, the maximum number of hidden nodes for ELM should be equal to the number of training samples. However, it is time consuming

and unnecessary. Thus, for both ASLGEM-ELM and the proposed method, 25 different numbers of hidden nodes are tested, i.e., $M = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150\}$.

4.4.3 Experimental Analysis

Table 4.3 reports the average number of hidden nodes and testing accuracy of the methods over 50 trials. It can be seen that in 5-fold CV and 10-fold CV, the proposed method achieves the highest testing accuracy. Moreover, the proposed method not only gets the smallest variance but also obtains the best classification accuracy on 6 data sets out of 17. Compared with ASLGEM, the testing accuracy of our method is better on most data sets. This indicates that the combination of Q -value and training accuracy is effective in improving the generalization ability of SLFN.

As seen in Table 4.3, the proposed method tends to recommend a relatively larger number of hidden nodes. The average number of hidden nodes selected by the proposed method is even larger than that selected by 10-fold CV. Even though a large number of hidden nodes may cause overfitting, the architecture can better represent the underlying function. The high testing accuracy gives an evidence that the architecture selected by our method does not suffer from overfitting much.

Table 4.4 reports the average execution time of the five methods on different data sets. The learning speeds of ASLGEM and the proposed method are similar. Both of them are about 4 times faster than 5-fold CV and 8 times faster than 10-fold CV. With k -fold CV conducted L times, $k \times L$ classifiers are required to

be trained, which is definitely time consuming. While in the proposed method, the computation on SSM does not relate on the number of training samples N . However, the selection on R_{SM}^* requires to train a classifier, which will dominate the total execution time of the architecture selection. If using traditional back-propagation neural network (BPNN) or multilayer perceptron neural network (MLPNN), the time complexity will be very high. Fortunately, ELM is capable to train the SLFN with fast speed, which guarantees a high efficiency of the proposed method. The time complexity of decision making by using MCDM is low thus can be ignored in the learning process.

It is observed from Table 4.3 that all the methods have achieved a poor result on data set *Yeast*. This data set is typically imbalanced, i.e., 76% samples are included in 3 classes and the rest are included in 7 classes. The classes with a small number of training samples are called minority classes. On this data set, ASLGEM demonstrates the worst performance. Moreover, the optimal number of hidden nodes selected by ASLGEM is 1, which indicates that it fails to perform an effective search in the space. The major reason could be that ASLGEM focuses on minimizing the overall error that selects MSE as the key component in Q -value determination. However, it is adverse to the minority classes, since these classes have trivial contribution to the overall error. On the other hand, the proposed method achieves an accuracy 22% higher than ASLGEM. Based on this strategy, the selected number of hidden nodes is more convincing.

Table 4.3: Comparisons of Different Model Selection Methods: Average Number of Nodes and Testing Accuracy for 50 Trials

Datasets	2-fold CV		5-fold CV		10-fold CV		ASLGEM		Proposed	
	Nodes	Accuracy	Nodes	Accuracy	Nodes	Accuracy	Nodes	Accuracy	Nodes	Accuracy
SPECTF	4.80	79.63 \pm 3.76	8.14	78.98 \pm 3.45	11.78	78.63 \pm 4.72	1.76	79.63 \pm 3.75	1.76	79.63 \pm 3.75
Cancer	17.62	96.14 \pm 1.51	23.70	96.35 \pm 1.39	24.70	96.48 \pm 1.33	4.50	91.73 \pm 6.80	88.14	95.07 \pm 1.73 \uparrow
Bupa	19.06	70.08 \pm 4.42	27.40	69.81 \pm 3.91	20.04	70.25 \pm 4.07	23.20	70.54 \pm 4.21	23.80	70.69 \pm 4.16 \uparrow
German	36.30	74.14 \pm 2.47	56.80	74.57 \pm 2.66	57.60	74.49 \pm 2.33	20.82	72.93 \pm 2.98	41.90	74.35 \pm 2.99 \uparrow
Haberman	8.38	74.28 \pm 3.72	9.34	74.28 \pm 3.80	10.16	73.28 \pm 3.46	6.48	74.46 \pm 4.01	9.14	74.41 \pm 3.86 $\downarrow\downarrow$
Pima	19.06	76.26 \pm 2.31	19.94	76.45 \pm 2.36	22.20	76.37 \pm 2.14	7.52	74.43 \pm 3.98	34.40	76.10 \pm 3.07 \uparrow
Wdbc	24.04	95.50 \pm 1.63	40.46	95.54 \pm 1.75	37.28	95.68 \pm 1.48	4.62	86.16 \pm 9.71	91.20	95.22 \pm 1.52 \uparrow
Wpbc	8.68	76.71 \pm 4.72	18.94	76.03 \pm 5.54	21.20	75.93 \pm 5.99	2.80	76.44 \pm 4.03	2.80	76.44 \pm 4.03
Chart	79.80	90.82 \pm 2.63	117.20	92.53 \pm 2.12	116.40	92.01 \pm 2.04	18.90	74.19 \pm 6.94	134.80	92.68 \pm 2.11 \uparrow
Cotton	47.40	90.02 \pm 2.45	67.00	91.03 \pm 2.92	66.80	91.08 \pm 2.64	13.18	75.08 \pm 6.36	89.00	91.20 \pm 2.42 \uparrow
Dermatology	40.80	96.75 \pm 1.89	63.60	97.02 \pm 1.62	63.60	96.96 \pm 1.69	7.36	77.04 \pm 7.65	115.60	96.96 \pm 1.25 \uparrow
Glass	28.00	82.00 \pm 5.65	39.80	84.13 \pm 4.81	43.60	85.72 \pm 5.28	10.22	70.72 \pm 7.18	34.20	84.97 \pm 4.28 \uparrow
Libras	54.00	72.70 \pm 4.42	76.60	74.74 \pm 3.89	86.60	75.24 \pm 4.38	35.40	67.57 \pm 6.03	120.40	74.13 \pm 4.36 \uparrow
Vowel	136.00	86.02 \pm 2.36	145.00	87.29 \pm 2.13	143.40	86.80 \pm 2.50	43.40	67.51 \pm 4.62	146.80	87.43 \pm 2.00 \uparrow
Yeast	35.90	58.95 \pm 2.34	44.90	58.77 \pm 2.51	47.20	59.28 \pm 2.36	1.00	30.53 \pm 2.51	134.60	57.96 \pm 2.38 \uparrow
Soybean	76.40	93.67 \pm 1.38	102.00	94.06 \pm 1.55	107.80	94.12 \pm 1.37	20.40	77.35 \pm 5.82	110.60	94.41 \pm 1.42 \uparrow
Avg.	39.77	82.10 \pm 2.98	53.80	82.60 \pm 2.90	55.02	82.64 \pm 2.99	13.85	72.89 \pm 5.41	73.70	82.60 \pm 2.83 \uparrow

Note: For each dataset, the highest testing accuracy is in bold face. For method *Proposed*, \uparrow and $\downarrow\downarrow$ respectively represent that compared with method *ASLGEM*, the testing accuracy of the selected model is improved or not.

4.4.4 Trade-off Analysis

Fig. 4.1 demonstrates the mean and standard deviation of training accuracy and Q -value of 20 ELM trials. It can be observed that on all the data sets, the training accuracy and the Q -value are trade-offs. When the training accuracy increases, the Q -value decreases. Although the increasing amplitude differs a lot on different data sets, the overall trends are roughly consistent. A larger Q -value represents that more unseen samples are included into the training process. As a result, it is more difficult for a classifier to correctly classify the training samples. Thus, it is necessary to achieve a trade-off between the training accuracy and Q -value. This observation strongly supports the motivation of adopting MCDM model in selecting ELM architecture.

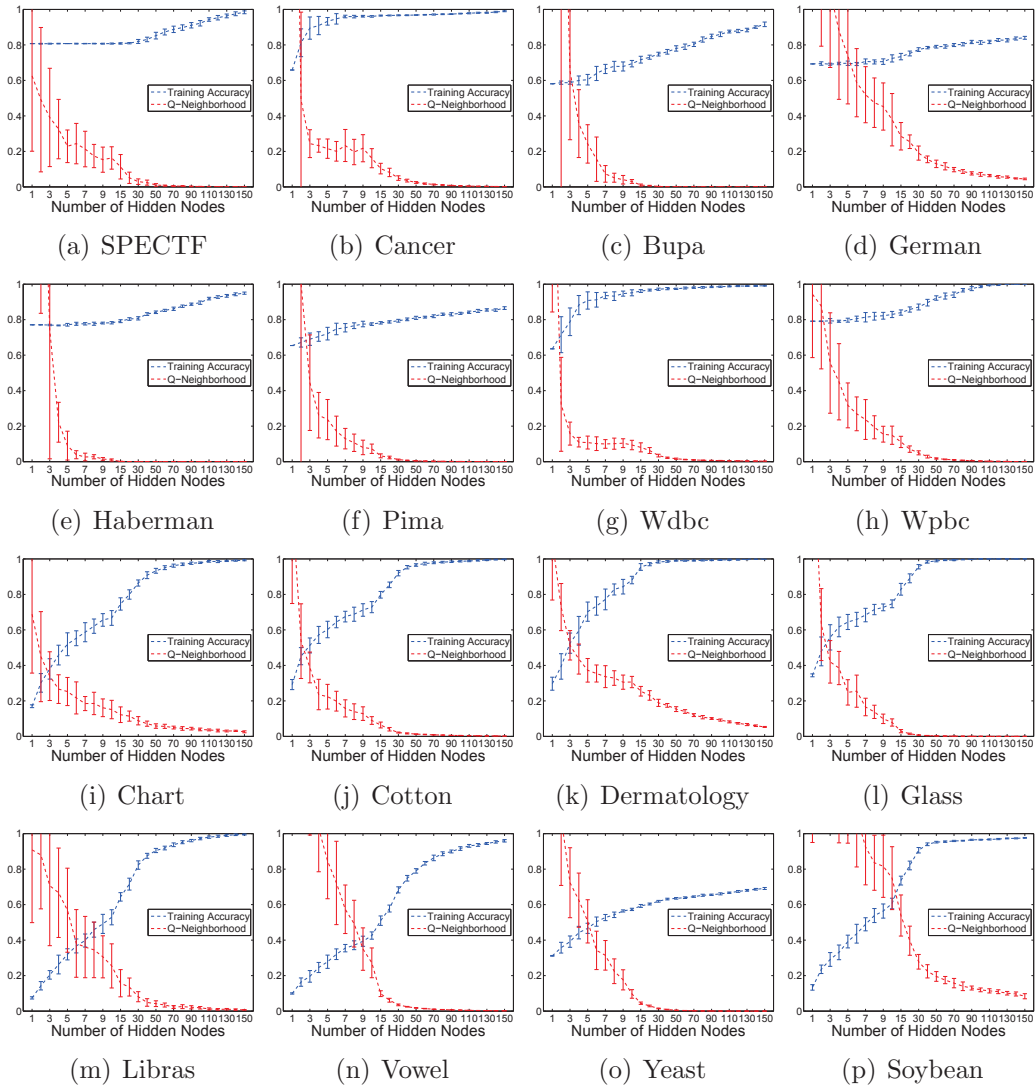


Figure 4.1: Mean and standard deviation of training accuracy and Q -value of 20 ELM trials.

Table 4.4: Comparisons of Different Model Selection Methods: Average Validation Time for 50 Trials

Datasets	2-fold CV	5-fold CV	10-fold CV	ASLGEM	Proposed
SPECTF	1.9300	7.0693	15.4722	1.7110	1.7110
Cancer	3.7631	13.4969	29.5853	3.3830	3.3837
Bupa	2.2174	7.7479	16.7152	1.8112	1.8112
German	5.6132	19.5794	42.5299	4.7331	4.7331
Haberman	1.9241	6.7620	14.5097	1.5912	1.5925
Pima	4.0548	14.6291	32.4217	3.7347	3.7353
Wdbc	3.5749	12.4932	27.2203	3.0283	3.0283
Wpbc	1.4939	5.3555	11.5793	1.2318	1.2324
Chart	3.8351	13.4326	28.9366	3.2227	3.2239
Cotton	2.2951	8.2259	18.0758	2.0636	2.0642
Dermatology	2.4913	9.1651	20.0533	2.2093	2.2105
Glass	1.5088	5.5159	11.7356	1.2767	1.2786
Libras	2.6330	9.1910	19.5117	2.1803	2.1809
Vowel	5.0688	19.5943	42.5630	4.7658	4.7677
Yeast	7.4740	27.4222	59.7796	6.5776	6.5783
Soybean	4.1999	14.5262	32.1540	3.5403	3.5403
Avg.	3.3798	12.1379	26.4277	2.9413	2.9420

4.5 Conclusion

In this chapter, a LGEM based architecture selection method with MCDM model for ELM is proposed, which measures the informativeness of architectures by their dominated and dominating indices. Focusing on finding the optimal number of hidden nodes for ELM, a trade-off between generalization ability and classification accuracy is determined by the MCDM model. Experiments on 16 data sets demonstrate the feasibility and the effectiveness of the proposed method. Compared with CV and ASLGEM-ELM, our method not only gives a higher classification accuracy, but also provides a faster learning speed.

Chapter 5

Interval ELM for Large-scale Data Based on Uncertainty Reduction

In the previous chapters, we have addressed the fuzzy measure representation and architecture selection problem of ELM. In this chapter, we focus on data compression, which can further reduce the computation complexity of ELM. We propose some discretization methods to convert continuous-valued attributes into intervals, and introduce a new concept of class label fuzzification to represent the dependency among different classes. Then, the classification problem is transformed into a regression problem on interval data, which largely reduces the computation complexity. Sixteen real-life data sets are used in experimental testing to demonstrate the effectiveness of our approach.

5.1 Introduction

Classification is useful for extracting meaningful information from large-scale data set. Traditional classification techniques include Artificial Neural Network (ANN), Decision Tree (DT), Support Vector Machine (SVM), etc. Practically, the

classification accuracy is related to three characteristics, i.e., sample size, number of attributes, and classification model [60]. Although a positive causality between data volume and classification accuracy can usually be found, the model accuracy may or may not improve while increasing data volume [41]. Furthermore, huge amount of data (like hundreds of gigabytes of data) is difficult to fit into the memory of computers and causing serious problems in learning.

Training ELM on massive data with high dimensionality is still a very challenging problem. It is noted that the main time complexity of training an ELM is in calculating the pseudo-inverse of the hidden layer output matrix, especially if the size of the matrix is large. We believe that, this problem can be partially solved by some data compression techniques. Discretization [37], is a common method for quantizing continuous attributes. Motivated by the above situation, we propose an interval ELM model for large-scale data classification. First, each conditional attribute is discretized into a number of intervals based on uncertainty reduction. Then, the center and range of each interval are represented by the mean and standard deviation. Afterwards, the samples in the same intervals with regard to all the conditional attributes are merged as one record, and a fuzzification process is performed on the class labels. Finally, the original data set is compressed into a smaller one with fuzzy classes, and the interval ELM model is built up on the compressed data.

The rest of this chapter is organized as follows: in section 5.2, a brief introduction of our work is given. In section 5.3, we present a framework for attribute discretization and class fuzzification. In section 5.4, we apply the attribute discretization framework to ELM and propose the interval ELM model. In section

5.5, we present some experimental results to show the feasibility and effectiveness of the propose model. Finally, Section 5.6 presents the conclusions.

5.2 Challenges in Learning from Large-Scale Data for ELM

Applying ELM to large-scale data classification is a challenging problem. Several alternatives are listed in the following for handling this problem.

1. **Sequential learning:** the large-scale data set can be divided into small subsets, then the training instances are sequentially presented to the learning algorithm.
2. **Divide-and-conquer strategy:** the data matrix is divided into a number of small sub-matrices, then a learner is trained for each sub-matrix, and the results are integrated based on some techniques in linear algebra.
3. **Sample and feature selection:** perform both feature selection and sample selection on the large-scale data set in order to refine the samples and remove data redundancy, then a learner is trained on the refined data.

In this chapter, another approach is proposed, i.e., discretization of conditional attributes and fuzzification of decision labels. When all attributes are continuous, it is hard to find samples with exactly the same values. As a result, similar samples are treated as entirely different with each other, which lead to data redundancy. On the contrary, with discrete attributes, the data set can be made more compact and short, hence the learning is more effective and efficient. However, discretization may sometimes become intractable due to the heavy

matrix and integral operations involved. For example, the discretization process will be time consuming when there are too many training attributes. Moreover, error always exists after discretization, thus a tradeoff between accuracy and compression rate should be considered in real applications.

In order to handle large-scale data, we also fuzzify class labels by learning a set of memberships. In decision theory, membership could be considered as a kind of capacity, which weakens the probability axiom of countability. In other words, it reflects the likelihood of a conditional event. Finally, the fuzzification of class labels could be realized by computing the mean of the decision labels in the same conditional group.

5.3 Discretization of Conditional Attributes and Fuzzification of Decision Label

In this section, some uncertainty measurements are introduced firstly, then the attribute discretization framework and the label fuzzification model are proposed. For simplicity, only the binary classification problem is considered.

5.3.1 Uncertainty Measurement

The most widely used uncertainty measurements are information entropy [66] and Gini-index [6]. Suppose \mathbb{S} is a labeled sample set with L classes, the information entropy and Gini-index of set \mathbb{S} are respectively defined as:

$$f(\mathbb{S}) = - \sum_{l=1}^L p_l \log_2 p_l, \quad (5.1)$$

and

$$f(\mathbb{S}) = 1 - \sum_{l=1}^L p_l^2, \quad (5.2)$$

where p_l is the probability of class l in \mathbb{S} . Clearly, a more balanced probability distribution will lead to a larger value of these two measures. When all the samples are from the same class, i.e. $p_l = 1$ for a certain $l \in \{1, \dots, L\}$, they reach their minimum, which lead to the smallest uncertainty. When the numbers of samples from all the classes are equivalent, i.e., $p_l = 1/L$ for $l = 1, \dots, L$, they reach their maximum, which lead to the largest uncertainty.

5.3.2 Discretization of Conditional Attributes Based on Uncertainty Reduction

It is noteworthy that for a continuous-valued attribute, the number of available cut-points is infinite. In order to tackle this problem, a feasible solution for attribute discretization is proposed. As preliminaries, some basic concepts will be introduced firstly. Several notations are adopted in this section, i.e., $\mathbb{X} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i = [x_{i1}, \dots, x_{in}]^T \in R^n, y_i \in \{1, \dots, L\}, i = 1, \dots, N\}$ is a training set that contains N labeled samples with n input attributes and L possible labels, and all the attributes are continuous ones.

Definition 1. (candidate cut-point) Let A_j be the j -th attribute for the sample set \mathbb{X} . Suppose the values of all the samples in \mathbb{X} with regard to A_j are ranked by ascending order, i.e., $x_{1j}^* \leq x_{2j}^* \leq \dots \leq x_{Nj}^*$, where $x_{1j}^*, x_{2j}^*, \dots, x_{Nj}^* \in \{x_{1j}, x_{2j}, \dots, x_{Nj}\}$. The midpoint of any two adjacent values in this order is considered as a candidate cut-point of attribute A_j with respect to \mathbb{X} .

Obviously, there are $N - 1$ candidate cut-points for each attribute. Sup-

pose the set of candidate cut-points for A_j as \mathbb{C}_j , then $\mathbb{C}_j = \{\hat{x}_{ij} | \hat{x}_{ij} = (x_{ij}^* + x_{i+1,j}^*)/2, i = 1, \dots, N - 1\}$. Suppose A_j is split into two intervals by a candidate cut-point \hat{x} , accordingly, \mathbb{X} is divided into two sets, i.e., $\mathbb{X}_1 = \{\mathbf{x}_i \in \mathbb{X} | x_{ij} \geq \hat{x}\}$ and $\mathbb{X}_2 = \{\mathbf{x}_i \in \mathbb{X} | x_{ij} < \hat{x}\}$. The splitting performance of \hat{x} is then computed as:

$$SP(\mathbb{X}, \hat{x}) = f(\mathbb{X}) - \sum_{k=1}^2 \frac{|\mathbb{X}_k|}{|\mathbb{X}|} f(\mathbb{X}_k), \quad (5.3)$$

where $|\mathbb{X}|$ is the cardinality of set \mathbb{X} .

By adopting the uncertainty measurement of information entropy, a more effective performance evaluation model is given as [58]:

$$SP(\mathbb{X}, \hat{x}) = \frac{f(\mathbb{X}) - \sum_{k=1}^2 \frac{|\mathbb{X}_k|}{|\mathbb{X}|} f(\mathbb{X}_k)}{- \sum_{k=1}^2 \frac{|\mathbb{X}_k|}{|\mathbb{X}|} \log_2 \frac{|\mathbb{X}_k|}{|\mathbb{X}|}}, \quad (5.4)$$

where $f(\mathbb{X})$ is given in Eq. (5.1). Then, our attribute discretization framework is described as Algorithm 5.1.

As a result, Algorithm 5.1 returns a number of cut-points of attribute A_j for sample set \mathbb{X} . Suppose this number is $\mathbf{d} - 1$, then attribute A_j is discretized into \mathbf{d} intervals, which divide \mathbb{X} into \mathbf{d} subsets. By performing the above-mentioned processes on all $A_j, j = 1, \dots, n$, the discretization process is finished.

5.3.3 Fuzzification of Decision Label

The inconsistency of class labels in the same conditional group, i.e., same conditional attributes but different class labels, will affect the final classification accuracy. We proposed a label fuzzification method for solving this problem by transferring the training samples' attributes to approximate interval values. For example, suppose there is a continuous attribute ranging from a to b . After dis-

Algorithm 5.1 Attribute Discretization Based on Uncertainty Reduction

Input:

A continuous-valued attribute A_j ; sample set \mathbb{X} ; purity threshold $\theta \in (0, 1]$, and number threshold N^* .

Output:

A set of selected candidate cut-points $\hat{\mathbb{C}}_j$ for A_j .

- 1: Initialize $\Omega = \{\mathbb{X}\}$, and $\hat{\mathbb{C}}_j = \emptyset$;
 - 2: **while** Ω is not empty **do**
 - 3: Select a sample set from Ω , denoted by \mathbb{X}^* ;
 - 4: **if** $|\mathbb{X}^*| < N^*$ or $\max_{l=i,\dots,L} p_l > \theta$ **then**
 - 5: Continue;
 - 6: **else**
 - 7: Sort the samples in \mathbb{X}^* with ascending values of A_j ;
 - 8: Get all the available candidate cut-points of A_j in \mathbb{X}^* , i.e., \mathbb{C}_j ;
 - 9: Calculate the splitting performance of each candidate cut-point in \mathbb{C}_j based on Eq. (5.4);
 - 10: Select the candidate cut-point \hat{x} with the highest splitting performance;
 - 11: Split \mathbb{X}^* into two sets \mathbb{X}_1 and \mathbb{X}_2 by \hat{x} ;
 - 12: Delete \mathbb{X}^* from Ω , add \mathbb{X}_1 and \mathbb{X}_2 into Ω , and let $\hat{\mathbb{C}}_j = \hat{\mathbb{C}}_j \cup \hat{x}$;
 - 13: **end if**
 - 14: **end while**
 - 15: **return** $\hat{\mathbb{C}}_j$
-

cretization with 4 arities, and three cut-points as \hat{x}_1 , \hat{x}_2 , and \hat{x}_3 . This attribute is discretized into 4 intervals, i.e., $[a, \hat{x}_1)$, $[\hat{x}_1, \hat{x}_2)$, $[\hat{x}_2, \hat{x}_3)$ and $[\hat{x}_3, b]$, respectively. There are many ways to represent each interval. A sensible way is to compute the center and range of each interval as the mean and standard deviation of the samples.

Finally, by performing the discretization on all the n attributes and computing the mean value for each of their intervals, the samples with the same attribute values are considered as belong to the same conditional group. However, these samples may have different class labels. In this case, the frequencies of different class labels are recorded and used. And this process is the so-called *fuzzification of class labels*. The whole fuzzification process is described as Algorithm 5.2.

Algorithm 5.2 Fuzzification of Class Labels

Input:

Training set $\mathbb{X} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i = [x_{i1}, \dots, x_{in}]^T \in R^n, y_i \in \{0, 1\}, i = 1, \dots, N\}$, purity threshold $\theta \in (0, 1]$, and number threshold N^* .

Output:

Two compressed data sets: center matrix $\hat{\mathbb{X}}_c$ and range matrix $\hat{\mathbb{X}}_r$.

- 1: Let $\hat{\mathbb{X}}_c = \mathbb{X}$ and $\hat{\mathbb{X}}_r = \mathbb{X}$;
 - 2: **for** $j = 1, \dots, n$ **do**
 - 3: Call Algorithm 5.1 on \mathbb{X} and A_j with θ and N^* , suppose the number of arities is \mathbf{d}_j and the cut-points are $\hat{x}_1, \dots, \hat{x}_{\mathbf{d}_j-1}$;
 - 4: **for** $k = 1, \dots, \mathbf{d}_j$ **do**
 - 5: Let $\mathbb{X}_k = \{\mathbf{x}_i \in \mathbb{X} | x_{ij} \in [\hat{x}_{k-1}, \hat{x}_k)\}$, where \hat{x}_0 and $\hat{x}_{\mathbf{d}_j}$ are the lower and upper bounds of A_j ;
 - 6: Let $\bar{x}_{kj} = \sum_{\mathbf{x}_i \in \mathbb{X}_k} x_{ij} / |\mathbb{X}_k|$, and $sd_{kj} = std(\{x_{ij} | \mathbf{x}_i \in \mathbb{X}_k\})$;
 - 7: $\forall \mathbf{x}_i \in \mathbb{X}_k$, set $x_{ij} = \bar{x}_{kj}$ in $\hat{\mathbb{X}}_c$ and $x_{ij} = sd_{kj}$ in $\hat{\mathbb{X}}_r$;
 - 8: **end for**
 - 9: **end for**
 - 10: For $\hat{\mathbb{X}}_c$ and $\hat{\mathbb{X}}_r$, merge the same samples as one record, compute its class label as the mean of the original labels in $\hat{\mathbb{X}}_c$ and standard deviation of the original labels in $\hat{\mathbb{X}}_r$;
 - 11: **return** $\hat{\mathbb{X}}_c$ and $\hat{\mathbb{X}}_r$.
-

When Algorithm 5.2 is used on a training set, the number of samples will become smaller, and some of the class labels will become fuzzy class labels. As a result, both $\hat{\mathbb{X}}_c$ and $\hat{\mathbb{X}}_r$ are compressed compared with \mathbb{X} . Usually, $\hat{\mathbb{X}}_c$ can be directly used to train the learning model, and $\hat{\mathbb{X}}_r$ can be used to evaluate the bias of $\hat{\mathbb{X}}_c$. It depends on the problem that whether $\hat{\mathbb{X}}_r$ is useful or not.

When the number of features in a data set is large, the samples are difficult to have same attribute values, which may make the discretization process useless. In order to discover compact representations of high-dimensional data before the discretization process, the correlation-based feature selection (CFS) method [18] is adopted to remove redundant or irrelevant features. CFS has three main advantages [63]: 1) it can model the feature dependencies in data; 2) it is independent of the base classifiers, and 3) it has lower computational complexity than other methods.

5.4 Interval Extreme Learning Machine

In this section, the interval ELM model is developed based on the previous discretization and fuzzification results, followed by an illustrative example.

With the fuzzification of class labels, a binary classification problem is turned into a regression problem with interval data. However, in many real world applications, there exists dependency [52] in data, which causes undue increase in width of results and makes them less useful in real applications. For simplicity, only the center information of the interval data is being used. Center method (CM) is combined with ELM to develop a fast regression model.

Let $\hat{\mathbb{X}}_c = \{(\mathbf{x}_i^c, y_i^c)\}_{i=1}^{\hat{N}}$ be the center matrix of \mathbb{X} compressed by Algorithm 5.2.

Then, each sample is represented as (\mathbf{x}_i^c, y_i^c) , where $\mathbf{x}_i^c = [x_{i1}^c, \dots, x_{in}^c]^T \in R^n$, and $y_i^c \in [0, 1]$, $i = 1, \dots, \hat{N}$. The SLFN on $\hat{\mathbb{X}}_c$ is modeled as

$$\sum_{j=1}^{\hat{N}} \beta_j^c g(\mathbf{w}_j^c \cdot \mathbf{x}_i^c + b_j^c) = y_i^c, i = 1, \dots, \hat{N}. \quad (5.5)$$

By taking it as a regression problem, basic ELM is conducted on $\hat{\mathbb{X}}_c$. Finally we can get

$$\beta^c = \mathbf{H}^{c\dagger} \cdot \mathbf{Y}^c, \quad (5.6)$$

where $\mathbf{Y}^c = [y_1^c, \dots, y_{\hat{N}}^c]^T$, and \mathbf{H}^c is the corresponding hidden layer output matrix. The entire learning process is shown as follows:

- Step 1: Perform a feature selection process (i.e., CFS method described in chapter 2) on the training set, and obtain the selected data set i.e., \mathbb{X} .
- Step 2: Apply Algorithm 5.2 on \mathbb{X} , get the center matrix \mathbb{X}^c .
- Step 3: Apply basic ELM algorithm on \mathbb{X}^c , which produces the ELM model parameters .
- Step 4: Given a binary testing sample \mathbf{x}^* , if the output value of the ELM model falls in $[0,0.5]$, it is considered as negative, otherwise is considered as positive.

Fig. 5.1 describes the flow of the whole process.

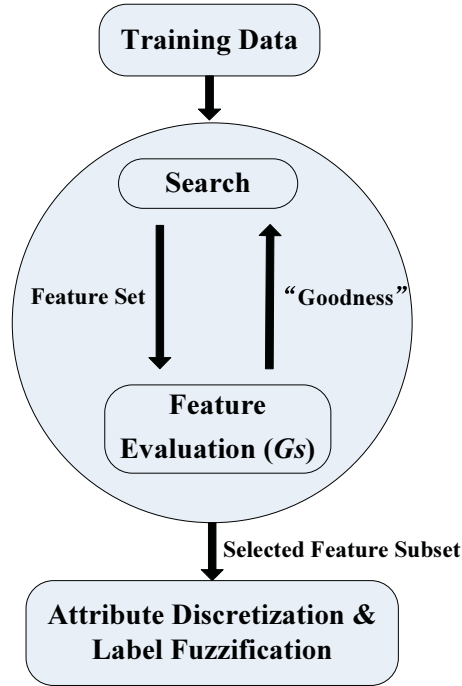


Figure 5.1: Learning structure of IELM: G_s is the heuristic calculated by CFS

5.5 Experimental Comparisons

In this section, some experimental comparisons are presented to show the feasibility and effectiveness of the proposed method.

5.5.1 Methods of Comparison

Five methods are listed in this section for performance comparison.

Basic ELM: The basic ELM algorithm is directly performed on the training samples.

Interval ELM with Equal Width Discretization: This method, denoted by *IELM+EWD* in short, applies equal width discretization to the conditional attributes. For each attribute, the whole range is divided into a fixed number of intervals with equal width.

Interval ELM with IDE3: This method, denoted by *IELM+IDE3* in short, adopts the same framework of the proposed method. However, the splitting performance of a CCP is measured by Eq. (5.3) rather than Eq. (5.4), and the frequency measure is by Eq. (5.1). The heuristic is the same with the IDE3 DT induction method.

Interval ELM with CART: This method, denoted by *IELM+CART* in short, also adopts the same framework of the proposed method. The splitting performance of a CCP is measured by Eq. (5.3), and the frequency measure is by Eq. (5.2). The heuristic is the same with the CART DT induction method.

Interval ELM with Uncertainty Reduction (proposed): The proposed method is applied.

5.5.2 Experimental Design

The five methods are first conducted on 15 small UCI benchmark data sets as shown in Table 5.1. Then, the Basic-ELM, IELM+IDE3, IELM+CART and the proposed methods are compared on a large-scale data set named *Skin*, which contains 163,371 training samples and 81,686 testing samples with 3 conditional features. The equal-width method for this data set is omitted since the number of intervals to be discretized is hard to decide, which may lead to an incomparable result.

Different parameters on the 15 small data sets are tested in order to investigate the changing trends of the results produced by the compared methods. For equal-width discretization, 5 different numbers of intervals have been tried, i.e., {80, 40, 20, 10, 5}. For the IELM+IDE3, IELM+CART and the proposed

method, 5 different values of N^* , i.e., $\{25, 20, 15, 10, 5\}$, and 5 different values of θ , i.e., $\{0.75, 0.8, 0.85, 0.9, 0.95\}$ have been tested, which result in 25 different pairs of (N^*, θ) . As for the large-scale data set, only one parameter setting, i.e., $(N^*, \theta) = (10, 0.9)$ is applied.

Sigmoid activation function is adopted in ELM. In order to eliminate the random effect, two-fold cross validation is conducted 50 times for each data set. Finally, 100 results are derived and the average value is observed.

The feature selection process is first carried out with the machine learning toolbox WEKA 3.7. Then, the algorithms are carried out under MATLAB 7.9.0 environment in a PC with a 3.16-GHz Intel Core Duo CPU, a 4GB memory, and 64-bit windows 7 system.

5.5.3 Performance Comparison on Small UCI Benchmark Data

The performance of the proposed method is sensitive to the parameter combination of (N^*, θ) , where N^* indicates the minimum number of instances in an interval and θ measures the purity of the instances in it. That is to say, the splitting on a subset will be terminated when the number of instances in it is a smaller than N^* or the class purity of the instances is larger than θ . Theoretically, a bigger θ and smaller N^* will lead to a higher accuracy and a worse compression rate, where the compression rate is defined as

$$\begin{aligned} & \textit{Compression rate} \\ = & \frac{\textit{Number of samples in the reduced data set}}{\textit{Number of samples in the original data set}} \end{aligned} \quad (5.7)$$

Fig. 5.2 and Fig. 5.3 respectively show the testing accuracy and compression

rate of the proposed method with different parameter settings of (N^*, θ) . From Fig. 5.2, it can be seen that on most of the data sets, the testing accuracy will increase with the increase of θ and decrease of N^* . Although the increasing amplitude differs a lot on different data sets, the overall trends are roughly consistent. Similar trends could also be found on the compression rate in Fig. 5.3, which clearly demonstrate that the data set can be compressed to a great extent with a larger N^* and smaller θ . However, a highly compressed data set will lead to a poor testing accuracy, thus, it is necessary to achieve a trade-off between the testing accuracy and compression rate.

The optimal parameters of the methods are given in Table 5.2. With these parameters, Table 5.4 shows the best performance of the methods and the corresponding compression rate. It can be seen that the proposed method achieves the highest average testing accuracy. Besides, it gives the best performances on 7 data sets out of 15. On the other hand, highly compressed data can enable opportunities for significant performance improvement by reducing redundancies in the original data. This effect is visible in data sets *German*, *Haberman*, and *Wdbc*, where the proposed method not only achieves the best accuracy but also the best compression performance (i.e. 50.10%, 42.48% and 77.5%, respectively). Moreover, the accuracy of the proposed method and the basic ELM are more or less similar.

In addition, some observations and analyses on several special cases are given as follows.

Result on data set *Australian*: It is observed from Fig. 5.3(c) that the compression performance of the proposed method on this data set is mainly influenced by

parameter N^* and is insensitive to θ . Besides, as shown in Fig. 5.2(c), the change of testing accuracy is trivial with different parameter settings. This observation demonstrates a possibility of achieving a good compression while maintaining a high accuracy.

Result on data set *Sonar*: It is observed from Fig. 5.3(1) that this data set cannot be compressed. This observation might be caused by two reasons: 1) during the discretization process, the numbers of arities are large, and 2) compared with other data sets, the number of attributes is large. In this case, samples are hard to have same attribute values and cannot be merged together. Accordingly, as shown in Fig. 5.2(1), the accuracy has no obvious change. Two solutions can be considered for handling this case: 1) set larger value for N^* and smaller value for θ , and 2) further reduce the number of attributes by other feature selection method.

Result on data sets *SPECTF* and *Wpbc*: It can be seen from Fig. 5.3(b) and Fig. 5.3(o) that the compression rates of these two data sets are dropping sharply when $N^* = 25$. One major reason is the high similarity among the samples, which generates very limited number of arities during the discretization process. In practice, smaller number of arities gives a higher chance of merging different samples. However, as shown in Fig. 5.2(b) and Fig. 5.2(o), the proposed method can maintain the accuracy with highly compressed data on *SPECTF* but is failed on *Wpbc*. Thus, it is necessary to distinguish the samples with different class labels but with very close attribute values.

Finally, paired Wilcoxon signed rank tests are conducted on the results given in Table 5.4, where the p -values are listed in Table 5.5. In the tests, the p -value

is the probability of whether the null hypothesis is true [68]. The significance level of 0.05 is adopted, which indicates that the two referred methods are statistically different when the p -value is small than 0.05. It can be seen that the proposed method is significantly different from all the other methods, which strongly testifies its effectiveness on small data sets.

5.5.4 Performance Comparison on Large-Scale Data

The performance comparison of Basic-ELM, IELM+IDE3, IELM+CART and the proposed method on the large-scale data set *Skin* are shown in Table 5.6. It is observed that the result of IELM+IDE3, IELM+CART and the proposed method are similar, where the proposed method is slightly better than the other two. Although the IELM based methods have a worse accuracy than Basic ELM (average 8% worse), they demonstrate a faster learning speed (average 5 times faster). Moreover, they show a comparable result when the number of hidden nodes is large. For example, the proposed method achieves the accuracy of 96.08 ± 4.11 with 100 hidden nodes, which is only 3% lower than Basic ELM. Considering the execution time, this level of tradeoff is acceptable.

Table 5.3 shows the compression rate of data set *Skin*. The proposed method gives the best compression rate of 11.46% where the others are around 10%. This observation gives a conclusion that the proposed method has potential for solving large-scale data classification problems.

5.5.5 Performance Comparison with SVM and KNN

We further compare the proposed method with the state-of-the-art classification models, i.e., support vector machine (SVM) and k -nearest neighbor (KNN). It

Table 5.1: Selected Small Binary Data Sets for Performance Comparison

Data Set	# Attributes (original)	# Attributes (reduced)	# Samples	# Hidden Nodes
CT	36	4	221	10
SPECTF	44	7	267	10
Australian	7	4	690	20
Bupa	6	4	345	5
Cancer	9	5	683	10
German	3	2	1000	10
Haberman	3	2	306	20
Heart	5	3	270	10
Ionosphere	32	4	351	20
Pima	8	5	768	5
Plrx	12	4	182	5
Sonar	60	9	208	10
Transfusion	4	2	748	10
Wdbc	30	6	569	20
Wpbc	33	3	198	10

Table 5.2: Optimal Parameter Settings of Different Methods on the Selected Small Data Sets

Data Set	IELM+EWD	IELM+IDE3		IELM+CART		Proposed	
	# inter-vals	θ	N^*	θ	N^*	θ	N^*
CT	40	0.95	5	0.95	5	0.95	10
SPECTF	10	0.9	25	0.9	10	0.8	10
Australian	40	0.8	10	0.8	25	0.8	25
Bupa	80	0.75	15	0.75	15	0.75	25
Cancer	5	0.9	25	0.95	25	0.95	10
German	40	0.8	15	0.8	10	0.8	5
Haberman	20	0.95	5	0.9	5	0.85	20
Heart	20	0.85	15	0.8	5	0.9	25
Ionosphere	80	0.9	10	0.85	5	0.9	5
Pima	40	0.9	10	0.95	25	0.95	20
Plrx	10	0.8	25	0.95	25	0.8	25
Sonar	10	0.95	15	0.95	25	0.9	5
Transfusion	80	0.95	10	0.95	10	0.95	5
Wdbc	20	0.85	20	0.85	15	0.85	5
Wpbc	80	0.8	5	0.95	10	0.8	5

Table 5.3: Compression Rate (%) of the large-scale data Set *Skin*

	IELM+IDE3	IELM+CART	Proposed
# Original Data	163371	163371	163371
# Reduced Data	17870	16995	18722
Compression Rate	10.94	10.40	11.46

Table 5.4: Performance Comparisons of Different Methods on the Selected Small Data Sets: Testing Accuracy (%) and Compression Rate (%)

Data Set	Basic ELM		IELM+EWD		IELM+IDE3		IELM+CART		Proposed	
	Accuracy	Rate	Accuracy	Rate	Accuracy	Rate	Accuracy	Rate	Accuracy	Rate
CT	90.96 \pm 1.34	90.77 \pm 1.32	100.00	100.00	90.64 \pm 1.34	81.90	90.89 \pm 1.19	81.45	90.93 \pm 1.31	83.71
SPECTF	79.21 \pm 1.16	79.27 \pm 1.27	100.00	100.00	79.43 \pm 1.35	94.76	79.34 \pm 1.43	94.38	79.45 \pm 1.41	94.76
Australian	78.47 \pm 1.03	76.52 \pm 0.82	61.16	61.16	78.61 \pm 1.11	82.75	78.69 \pm 1.12	77.83	78.67 \pm 1.06	85.94
Bupa	61.12 \pm 1.58	61.56 \pm 1.19	99.71	99.71	61.53 \pm 1.79	97.39	61.23 \pm 1.63	97.39	62.16 \pm 1.56	97.97
Cancer	96.58 \pm 0.56	97.04 \pm 0.67	61.64	61.64	96.95 \pm 0.40	77.89	96.82 \pm 0.37	70.57	96.97 \pm 0.35	74.96
German	70.67 \pm 0.52	36.60 \pm 5.06	17.20	17.20	70.51 \pm 1.04	47.40	69.40 \pm 1.13	45.40	70.83 \pm 1.22	50.10
Haberman	73.92 \pm 1.66	63.52 \pm 11.36	21.24	21.24	65.83 \pm 13.68	39.22	64.20 \pm 14.67	37.25	75.90 \pm 1.51	42.48
Heart	72.71 \pm 2.02	73.19 \pm 2.76	92.96	92.96	72.72 \pm 2.06	82.96	72.87 \pm 1.95	82.96	73.99 \pm 3.23	92.96
Ionosphere	87.91 \pm 2.64	85.18 \pm 2.51	80.91	80.91	86.03 \pm 3.83	58.12	85.29 \pm 4.47	50.71	86.38 \pm 3.88	56.13
Pima	76.27 \pm 1.69	76.28 \pm 1.75	99.61	99.61	76.61 \pm 2.20	99.74	77.02 \pm 2.12	99.74	77.01 \pm 1.74	99.74
Plrx	71.31 \pm 0.49	71.38 \pm 0.22	96.15	96.15	71.43 \pm 0.00	94.51	71.33 \pm 0.50	93.96	71.40 \pm 0.24	97.80
Sonar	77.38 \pm 3.99	77.55 \pm 4.48	100.00	100.00	77.15 \pm 4.30	100.00	77.57 \pm 4.23	100.00	77.62 \pm 4.36	100.00
Transfusion	76.56 \pm 0.95	44.94 \pm 15.74	11.63	11.63	75.72 \pm 2.65	33.42	75.93 \pm 2.06	33.42	76.44 \pm 1.35	33.56
Wdbc	95.72 \pm 0.97	95.92 \pm 0.92	100.00	100.00	96.08 \pm 0.95	77.15	96.08 \pm 0.86	75.57	96.10 \pm 0.88	77.50
Wpbc	78.83 \pm 1.88	79.00 \pm 2.16	90.40	90.40	78.89 \pm 2.15	72.22	78.28 \pm 3.49	61.11	78.87 \pm 1.98	84.34
Avg.	79.17 \pm 1.50	73.92 \pm 3.48	75.51	75.51	78.54 \pm 2.59	75.96	78.33 \pm 2.75	73.45	79.51 \pm 1.74	78.13

Note: For each data set, the highest accuracy is in bold face.

Table 5.5: Paired Wilcoxon’s Signed Rank Tests of Testing Accuracies (p Values)

Method	IELM +EWD	IELM +IDE3	IELM +CART	Proposed
Basic ELM	0.5614	0.9341	0.8040	0.0181†
IELM+EWD	—	0.1205	0.2078	0.0012†
IELM+IDE3	—	—	0.4212	0.0012†
IELM+CART	—	—	—	0.0003†

Note: For each test, † represent that the two referred methods are significantly different with the significance level 0.05.

Table 5.6: Comparative Results on the large-scale data Set *Skin*: Testing Accuracy (%) and Training Time (seconds)

Method # Hidden N- odes	Basic ELM		IELM+IDE3		IELM+CART		Proposed	
	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
20	97.38 \pm 0.31	0.7612	88.68 \pm 7.25	0.1046	86.65 \pm 7.08	0.0900	88.54 \pm 7.79	0.1050
40	98.68 \pm 0.13	1.8924	88.33 \pm 6.68	0.2462	88.08 \pm 6.83	0.2208	88.92 \pm 6.34	0.2668
60	99.01 \pm 0.08	2.7126	89.82 \pm 6.92	0.3504	91.70 \pm 5.84	0.3536	91.72 \pm 6.30	0.4080
80	99.26 \pm 0.04	3.9768	93.04 \pm 6.41	0.6008	91.57 \pm 6.73	0.5844	92.89 \pm 6.43	0.6670
100	99.37 \pm 0.04	5.3236	94.51 \pm 5.14	0.8358	93.04 \pm 6.30	0.8540	96.08 \pm 4.11	0.9190

is noteworthy that SVM and KNN are conducted on the data sets only with continuous valued attributes. For SVM, the slack variable C is fixed as 100, and gaussian radial basis function (RBF) kernel $\mathcal{K}(\mathbf{x}, \mathbf{x}_i) = \exp(-\frac{\|\mathbf{x}-\mathbf{x}_i\|}{2\sigma^2})$ is selected

Table 5.7: Performance Comparison of the Proposed Method with SVM and KNN on the Selected Small Data Sets

Data Set	Proposed	SVM	KNN
CT	90.93 ± 1.31	90.45 ± 1.38	88.57 ± 1.17
SPECTF	79.45 ± 1.41	80.69 ± 1.29	74.52 ± 1.58
Australian	78.67 ± 1.06	78.53 ± 0.81	70.98 ± 1.47
Bupa	62.16 ± 1.55	58.74 ± 2.01	47.37 ± 1.99
Cancer	96.97 ± 0.35	96.32 ± 0.54	96.17 ± 0.32
German	70.83 ± 1.22	69.93 ± 0.21	66.15 ± 1.12
Haberman	75.90 ± 1.51	73.71 ± 0.64	71.06 ± 1.44
Heart	73.99 ± 3.23	71.87 ± 1.17	64.69 ± 1.94
Ionosphere	86.38 ± 3.88	92.34 ± 0.92	85.56 ± 1.22
Pima	77.01 ± 1.74	76.34 ± 0.68	70.11 ± 0.99
Plrx	71.40 ± 0.24	70.16 ± 1.52	63.98 ± 2.39
Sonar	77.62 ± 4.36	74.21 ± 2.18	73.00 ± 1.61
Transfusion	76.44 ± 1.35	76.62 ± 0.33	69.59 ± 1.75
Wdbc	96.10 ± 0.88	96.83 ± 0.50	95.53 ± 0.38
Wpbc	78.87 ± 1.98	78.98 ± 1.50	74.02 ± 1.85
Avg.	79.51 ± 1.74	79.05 ± 1.05	74.09 ± 1.41

Note: For each data set, the highest accuracy is in bold face.

consistently with kernel parameter $\sigma = 1$. For KNN, the number of nearest neighbors k is set as 10. We repeat two-fold cross-validation 50 times for both SVM and KNN, then observe the average accuracy and standard deviation. The performance comparison is shown in Table 5.7. It is clear that the proposed method has achieved the best performance on 10 data sets out of 15, which demonstrates a great potential of it in solving various classification problems.

Finally, for the big data set *skin*, SVM has achieved an accuracy of 99.89% with the training time of 21.37 seconds. Although the performance of SVM is better than that achieved by the proposed method, its execution time is much higher, which demonstrates a low efficiency. Besides, when conducting KNN on this data set, a memory overflow happens due to the large number of samples. Thus, KNN is not an efficient method on big data under our problem environment.

5.6 Conclusions

In this chapter, an interval ELM model was developed for large-scale binary classification problems. This model is built up with two techniques, i.e., discretization of conditional attributes and fuzzification of class labels. Experimental comparisons demonstrate that the proposed method is not only able to improve generalization capability, but is also effective to compress data of large volume.

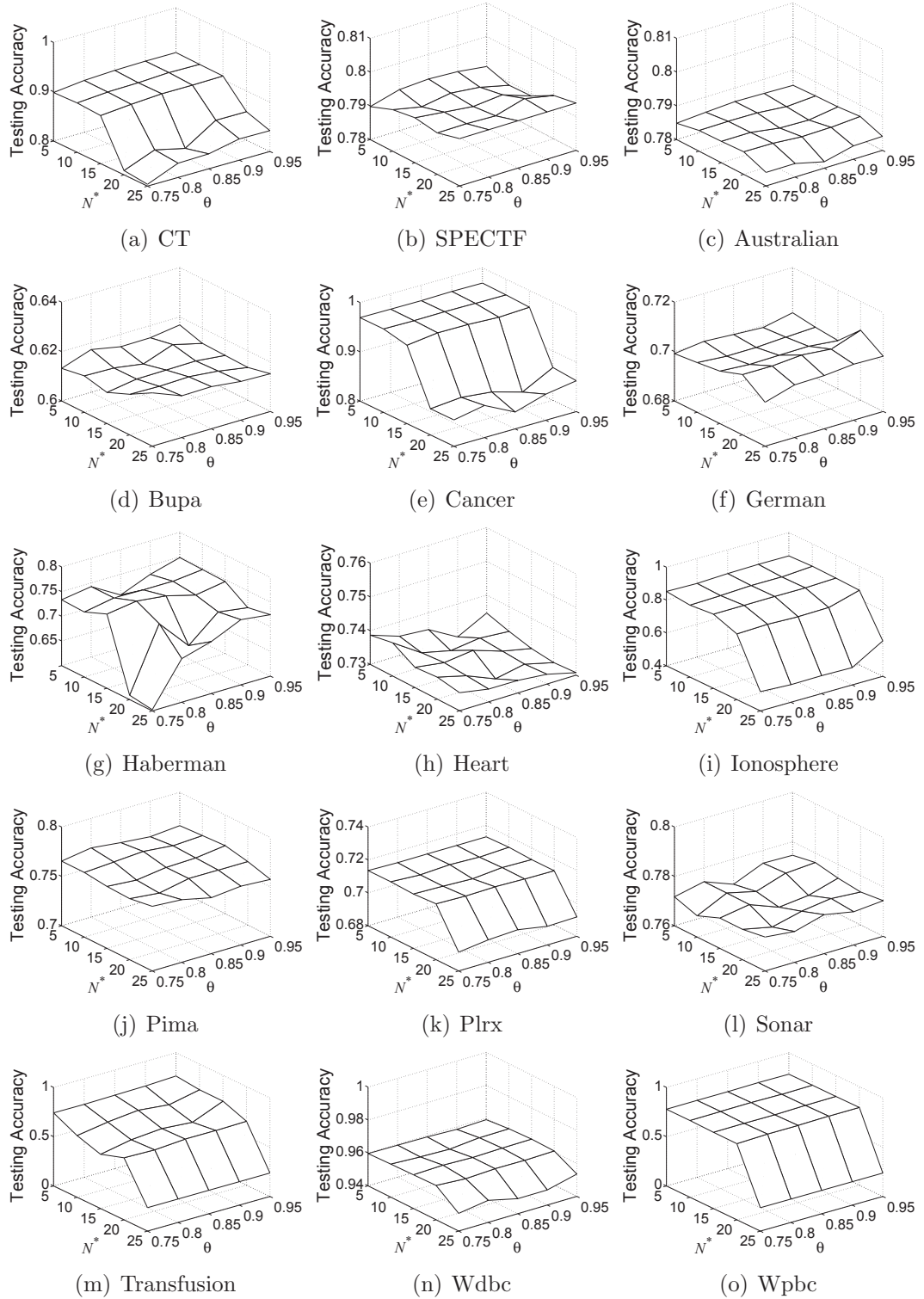


Figure 5.2: Testing accuracy with different settings of parameters N^* and θ of the proposed method.

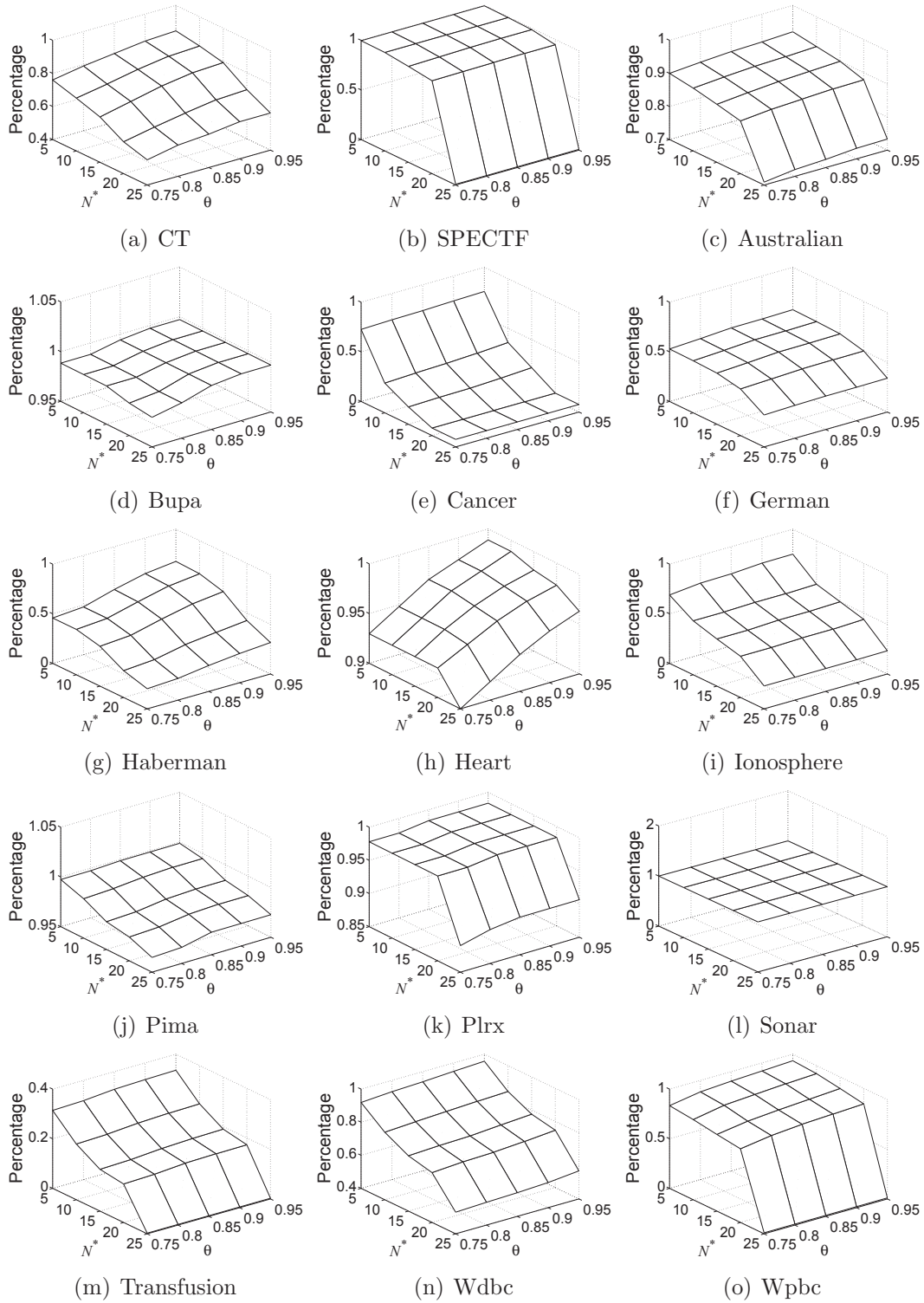


Figure 5.3: Percentage of the compressed data size to the original data size with different settings of parameters N^* and θ .

Chapter 6

Conclusions and Future Works

In this thesis, several ELM-based techniques have been developed with different motivations. We also applied FELM to perform fast unit combination strategy evaluation in RTS game. In this chapter, we conclude our works and provide future research directions. Section 6.1 presents a qualitative evaluation of each developed technique. An analysis is conducted to detail the speciality and limitation of each technique. 6.2 lists the possible research directions in the future.

6.1 Evaluation on the Developed ELM Techniques

6.1.1 Fuzzy Extreme Learning Machine

Compared with other gradient descent techniques, the advantage of using FELM for fuzzy measure determination is that the iterative learning mechanism can be removed without losing accuracy. In this model, the learning process of fuzzy measure determination can be regarded as solving a linear system with $2^n - 1$ variables, where each variable represents a subset of fuzzy measure. The developed set selection algorithm is able to locate the related fuzzy measure subset for each information source efficiently.

This algorithm is suitable to be used by different kinds of fuzzy integrals. Two new fuzzy integrals are proposed to model the interaction among features in game, i.e., mean-based fuzzy integral and order-based fuzzy integral. Mean-based fuzzy integral considers all the interactions that involve current information sources in fuzzy measure and then takes an average. Order-based fuzzy integral focuses on the highest proportion information sources, which considers the information source with highest resources first and estimates the interactions with all the other and so on. Compared with CI, these two new developed fuzzy integrals could achieve a better performance.

According to the experimental result, FELM is proved to be able to achieve faster learning speed and higher testing accuracy than traditional GA and P-SO. In addition, FELM is applied to the unit combination strategy evaluation problem in RTS game. Fuzzy measure and fuzzy integral are adopted to learn the performance of unit combination. The value of each fuzzy measure subset could be learned with the values of different fuzzy integrals. As the value of each subset is fully observed, the model is able to evaluate the unknown situation in the complicated game environment and assign a performance score to the strategy. Furthermore, fuzzy measure could represent super-additive and sub-additive situation among features, which is able to optimize the strategy planning model.

The main disadvantage of FELM is the sign of the learned output weight is determined by data, which hinders the way to extend FELM to other kinds of fuzzy measures, such as efficiency measure. Furthermore, the size of hidden layer output matrix may increase exponentially with the number of information source, which results in a high computational burden in learning the network.

6.1.2 LGEM based Architecture Selection for ELM with MCDM Model

The extended localized generalization error model (LGEM) aims to find an upper bound of the error between the target function and the learned function. The extended LGEM is able to provide useful guidelines for architecture selection in order to improve the generalization ability of SLFNs trained by ELM. In this part, we propose a new architecture selection method with MCDM model. The superiority of MCDM is that it can provide accurate priority information of the architectures with regard to different criteria independently. We incorporate two conflicting criteria, i.e., training accuracy and Q -value estimated by LGEM. The determination of Q -value depends on three components, i.e., training error, stochastic sensitivity measure (SSM), and a constant value.

There are four advantages of the proposed model:

1. The error bound provided by LGEM is effective, even for some special cases, i.e., $Q \rightarrow \infty, S_Q \rightarrow T$, and $Q \rightarrow 0, S_Q \rightarrow D$;
2. The model could be extended to other learning techniques;
3. The model has low time complexity, i.e., $O(M_n)$;
4. A trade-off between training accuracy and Q -value can control the number of hidden nodes to a reasonable range.

However, there is a major disadvantage of the model, i.e., the training samples are assumed to be uniformly distributed. When the distribution is not uniform, a new localized generalization error bound has to be derived.

6.1.3 Interval Extreme Learning Machine

Interval extreme learning machine (IELM) is developed for large-scale data classification with continuous-valued attributes. The interval ELM model is built up based on two techniques, i.e., discretization of conditional attributes and fuzzification of class labels. First, inspired by the traditional decision tree (DT) induction algorithm, each conditional attribute is discretized into a number of intervals based on uncertainty reduction scheme. Then, the center and range of each interval are calculated as the mean and standard deviation of the values. Afterwards, the samples in the same intervals with regard to all the conditional attributes are merged as one record, and a fuzzification process is performed on the class labels. As a result, the original data set is transferred into a smaller one with fuzzy classes, and the interval ELM model is developed. Experimental comparisons demonstrate the feasibility and effectiveness of the proposed approach.

The model design principle is to combine different data compression techniques, i.e., discretization, feature selection and class label fuzzification. Discretization is able to transfer a continuous attribute to discrete one, which significantly reduce the number of values for each attribute. The fuzzificated class label reflects the frequency of different classes, which reduces the inconsistency after discretization. Furthermore, feature selection is used to reduce redundant attribute and increase the chance of merging different samples. The experiments show that the proposed method achieves the highest average testing accuracy on 7 data sets out of 15. This phenomenon indicates that the highly compressed data can enable opportunities for performance improvement by reducing redundancies

in the original data.

However, the main disadvantage of our model is that the compression process is sensitive to the number of attributes. Generally, large number of attributes may decrease the probability in merging samples. According to the experimental result, the model is unable to achieve a satisfactory compression rate if the attribute number is larger than 9. It is necessary to develop an effective attribute reduction method to improve the model effectiveness. Moreover, how to use the range information of interval and how to extend the current model to multiclass classification problems are two major directions regarding this work in the future.

6.2 Future Works

This section briefly discusses the possible future works, which aim to overcome the limitations mentioned in Section 6.1.

The future work regarding fuzzy ELM may focus on two directions. First, we try to design different ELM structures to determine various types of fuzzy measures such as efficiency measure. An important problem for this topic is to control the sign of the output weights of the hidden layer output matrix. In order to learn efficiency measure, we wise to keep all the learned output weights are positive. Second, we try to improve the robustness of ELM algorithm, especially find out how to deal with the situation with insufficient valid data. On the other hand, regarding the application on RTS game, we will try to develop an autonomous real-time system that is capable of generating appropriate unit movement based on the current strategy evaluation model.

The future work regarding interval ELM may focus on tree directions. First, it

is necessary to develop a more effective attribute reduction method and discretization heuristic to support the learning process. Second, it might be interesting to extend the work to multi-class classification problems. Third, it will be useful to discuss how to make use of the range information.

Finally, the further research on architecture selection of ELM may concentrate on developing heuristic method to determine the weights of different criteria. Moreover, we would like to apply our method to game research problem such as technology development strategy selection in RTS game.

Appendix A.

A..1 Derivation of the Stochastic Sensitivity Measure for Single-Layer Feedforward Network with Sigmoid Function

As described in [81], SLFNs with sigmoid activation function could be described as

$$f_{\theta}(x) = \sum_{j=1}^M \beta_j g(w_j x + b_j) = \sum_{j=1}^M \beta_j \frac{1}{1 + \exp(-(w_j x + b_j))} \quad (\text{A..1})$$

According to Taylor's series expansion: $\frac{1}{1+x} = \sum_{t=0}^{\infty} (-1)^t x^t (-1 < x < 1)$, we have

$$f_{\theta}(x) = \sum_{j=1}^M \beta_j \sum_{t=0}^{\infty} (-1)^t (\exp(-(w_j x + b_j)))^t, (0 < \exp(-(w_j x + b_j)) < 1) \quad (\text{A..2})$$

Ignoring the terms which larger than 1, we have:

$$f_{\theta}(x) \approx \sum_{j=1}^M \beta_j (1 - \exp(-(w_j x + b_j))) \quad (\text{A..3})$$

Suppose that $S_j = \sum_{i=1}^n (w_{ij} x_i + b_{ij})$ and $S_j^* = \sum_{i=1}^n (w_{ij} (x_i + \Delta x_i) + b_{ij})$. Then we have

$$\begin{aligned}
& E_{S_Q}((\Delta y)^2) \\
&= E_{S_Q} \left(\left(\sum_{j=1}^M \beta_j (1 - \exp(-S_j^*)) - \sum_{j=1}^M \beta_j (1 - \exp(-S_j)) \right)^2 \right) \\
&= E_{S_Q} \left(\left(\sum_{j=1}^M \beta_j (\exp(-S_j) - \exp(-S_j^*)) \right)^2 \right)
\end{aligned} \tag{A.4}$$

Let $V_j = \exp(-s_j) - \exp(-s_j^*)$, then we have:

$$E_{S_Q}((\Delta y)^2) = \sum_{j=1}^M \sum_{i=1}^M \beta_i \beta_j E_{S_Q}(V_i V_j) \tag{A.5}$$

Based on it, we have:

$$\begin{aligned}
E_{S_Q}(V_i V_j) &= E_{S_Q}(\exp(-S_i - S_j)) - E_{S_Q}(\exp(-S_i - S_j^*)) \\
&\quad - E_{S_Q}(\exp(-S_i^* - S_j)) + E_{S_Q}(\exp(-S_i^* - S_j^*))
\end{aligned} \tag{A.6}$$

According to the central limit theorem, $\exp(S_j)$ and $\exp(S_j^*)$ have a log-normal distribution. Thus

$$\begin{aligned}
E_{S_Q}(\exp(-S_i^* - S_j^*)) &= \exp \left(\frac{\text{Var}(S_i^* + S_j^*)}{2} - E(S_i^* + S_j^*) \right) \\
&\approx 1 + \frac{\text{Var}(S_i^* + S_j^*)}{2} - E(S_i^* + S_j^*)
\end{aligned} \tag{A.7}$$

Then,

$$\begin{aligned}
E_{S_Q}(V_i V_j) &= \frac{1}{2} (\text{Var}(S_i^* + S_j^*) + \text{Var}(S_i + S_j) \\
&\quad - \text{Var}(S_i^* + S_j) - \text{Var}(S_i + S_j^*))
\end{aligned} \tag{A.8}$$

Because:

$$\begin{aligned}
\text{Var}(S_i^* + S_j^*) &= \text{Var} \left(\sum_{k=1}^n (w_{ki}(x_k + \Delta x_k) + b_{ki}) + \sum_{k=1}^n (w_{kj}(x_k + \Delta x_k) + b_{kj}) \right) \\
&= \sum_{k=1}^n (w_{ki} + w_{kj})^2 \text{Var}(x_k) + \frac{Q^2}{3} \sum_{k=1}^n (w_{ki} + w_{kj})^2
\end{aligned} \tag{A.9}$$

Finally, we have:

$$E_{S_Q}((\Delta y)^2) = \frac{Q^2}{3} \sum_{j=1}^M \beta_j^2 \sum_{k=1}^n w_{kj}^2 \quad (\text{A..10})$$

Bibliography

- [1] P. L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.
- [2] B. Biggio, G. Fumera, and F. Roli. Multiple classifier systems for robust classifier design in adversarial environments. *International Journal of Machine Learning and Cybernetics*, 1(1):27–41, 2010.
- [3] L. Billard and E. Diday. Regression analysis for interval-valued data. In *Data Analysis, Classification, and Related Methods*, pages 369–374. Springer, 2000.
- [4] P. Bradley and O. Mangasarian. Feature selection via concave minimization and support vector machines. In *International Conference on Machine Learning*, volume 98, pages 82–90, 1998.
- [5] P. Bradley, O. Mangasarian, and W. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998.
- [6] L. Breiman, L. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth International Group, 1984.
- [7] G. Campanella and R. Ribeiro. A framework for dynamic multiple-criteria decision making. *Decision Support Systems*, 52(1):52–60, 2011.

- [8] B. Chacko, V. Vimal Krishnan, G. Raju, and P. Babu Anto. Handwritten character recognition using wavelet energy and extreme learning machine. *International Journal of Machine Learning and Cybernetics*, 3(2):149–161, 2012.
- [9] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.
- [10] T.-H. Chang and T.-C. Wang. Using the fuzzy multi-criteria decision making approach for measuring the possibility of successful knowledge management. *Information Sciences*, 179(4):355–370, 2009.
- [11] E. Diday. Introduction à l’approche symbolique en analyse des données. *RAIRO - Operations Research - Recherche Opérationnelle*, 23(2):193–236, 1989.
- [12] J. Figueira, S. Greco, and M. Ehrgott. *Multiple criteria decision analysis: state of the art surveys*. Springer Verlag, 2005.
- [13] C. Fonseca, P. Fleming, et al. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In *Proceedings. 5th International Conference on Genetic Algorithms*, volume 1, page 416. San Mateo, California, 1993.
- [14] G. Forman. An extensive empirical study of feature selection metrics for text classification. *The Journal of Machine Learning Research*, 3:1289–1305, 2003.
- [15] F. Gioia and C. Lauro. Basic statistical methods for interval data. *Statistica Applicata*, 17(1), 2005.

- [16] M. Grabisch. k -order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92(2):167–189, 1997.
- [17] M. Grabisch and J. Nicolas. Classification by fuzzy integral: performance and tests. *Fuzzy sets and systems*, 65(2):255–271, 1994.
- [18] M. Hall and L. Smith. Practical feature subset selection for machine learning. In *Proceedings of the 21st Australasian Computer Science Conference*, pages 181–198. Springer, 1998.
- [19] S. Haykin and N. Network. A comprehensive foundation. *Neural Networks*, 2(2004), 2004.
- [20] K. Hipel, K. Radford, and L. Fang. Multiple participant-multiple criteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 23(4):1184–1189, 1993.
- [21] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.
- [22] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.
- [23] Q. Hu, W. Pan, S. An, P. Ma, and J. Wei. An efficient gene selection technique for cancer recognition based on neighborhood mutual information. *International Journal of Machine Learning and Cybernetics*, 1(1):63–74, 2010.
- [24] G. Huang, Q. Zhu, and C. Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006.

- [25] G.-B. Huang and H. A. Babri. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. *IEEE Transactions on Neural Networks*, 9(1):224–229, 1998.
- [26] G.-B. Huang and L. Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70(16):3056–3062, 2007.
- [27] G.-B. Huang and L. Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71(16):3460–3468, 2008.
- [28] 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.
- [29] G.-B. Huang, Y.-Q. Chen, and H. A. Babri. Classification ability of single hidden layer feedforward neural networks. *Neural Networks, IEEE Transactions on*, 11(3):799–801, 2000.
- [30] G. B. Huang, D. H. Wang, and Y. Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107–122, 2011.
- [31] S.-C. Huang and Y.-F. Huang. Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE Transactions on Neural Networks*, 2(1):47–55, 1991.
- [32] K. Ishii and M. Sugeno. A model of human evaluation process using fuzzy measure. *International Journal of Man-Machine Studies*, 22(1):19–38, 1985.
- [33] K. Jabeur, J. Martel, and S. Khélifa. A distance-based collective preorder integrating the relative importance of the group’s members. *Group Decision and Negotiation*, 13(4):327–349, 2004.

- [34] K. Javed, H. Babri, and M. Saeed. Feature selection based on class-dependent densities for high-dimensional binary data. *Knowledge and Data Engineering, IEEE Transactions on*, 24(3):465–477, 2012.
- [35] W. Jun, W. Shitong, and F. Chung. Positive and negative fuzzy rule system, extreme learning machine and image classification. *International Journal of Machine Learning and Cybernetics*, 2(4):261–271, 2011.
- [36] N. B. Karayiannis and A. N. Venetsanopoulos. Fast learning algorithms for neural networks. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 39(7):453–474, 1992.
- [37] R. Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of the tenth national conference on Artificial intelligence*, pages 123–128. Aaai Press, 1992.
- [38] B. Kommineni, S. Basu, and R. Vemuri. A spline based regression technique on interval valued noisy data. In *Proceeding of the Sixth International Conference on Machine Learning and Applications, 2007*, pages 241–247. IEEE, 2007.
- [39] S. Kotsiantis and D. Kanellopoulos. Discretization techniques: A recent survey. *GESTS International Transactions on Computer Science and Engineering*, 32(1):47–58, 2006.
- [40] N. Krayiannis and M. Randolph-Gips. On the construction and training of reformulated radial basis functions. *IEEE Trans. Neural Networks*, 14(4):835–846, 2003.

- [41] O. Kwon and J. Sim. Effects of data set features on the performances of classification algorithms. *Expert Systems with Applications*, 40(5):1847–1857, 2013.
- [42] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- [43] K. Leung, M. Wong, W. Lam, Z. Wang, and K. Xu. Learning nonlinear multiregression networks based on evolutionary computation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 32(5):630–644, 2002.
- [44] Y. Li, P. H. Ng, and S. C. Shiu. Extreme learning machine for determining signed efficiency measure from data. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 21(supp02):131–142, 2013.
- [45] Y. J. Li, H. F. P. Ng, H. B. Wang, C. K. S. Shiu, and Y. Li. Apply different fuzzy integrals in unit selection problem of real time strategy game. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 170–177. IEEE, 2011.
- [46] E. Lima Neto and F. de Carvalho. Centre and range method for fitting a linear regression model to symbolic interval data. *Computational Statistics & Data Analysis*, 52(3):1500–1515, 2008.
- [47] E. Lima Neto and F. de Carvalho. Constrained linear regression models for symbolic interval-valued variables. *Computational Statistics & Data Analysis*, 54(2):333–347, 2010.

- [48] H. Liu, F. Hussain, C. Tan, and M. Dash. Discretization: An enabling technique. *Data mining and knowledge discovery*, 6(4):393–423, 2002.
- [49] Y. Liu. Unbiased estimate of generalization error and model selection in neural network. *Neural Networks*, 8(2):215–219, 1995.
- [50] D. Lowe. Adaptive radial basis function nonlinearities, and the problem of generalisation. In *First IEE International Conference on Artificial Neural Networks, 1989.*, (Conf. Publ. No. 313), pages 171–175. IET, 1989.
- [51] R. Mesiar. Generalizations of k -order additive discrete fuzzy measures. *Fuzzy sets and systems*, 102(3):423–428, 1999.
- [52] R. Moore. *Interval analysis*, volume 2. Prentice-Hall Englewood Cliffs, 1966.
- [53] T. Murofushi and M. Sugeno. Fuzzy measures and fuzzy integrals. *Fuzzy Measures and Integrals: Theory and Applications*. Springer-Verlag, New York, pages 3–41, 2000.
- [54] H. P. Ng, Y. Li, H. Wang, Y. Li, and C. S. Shiu. An order-based fuzzy integral to model feature interactions in rts games. *IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games*, Submitted.
- [55] W. W. Ng, D. S. Yeung, M. Firth, E. C. Tsang, and X.-Z. Wang. Feature selection using localized generalization error for supervised classification problems using rbfnn. *Pattern Recognition*, 41(12):3706–3719, 2008.
- [56] W. W. Ng, D. S. Yeung, X.-Z. Wang, and I. Cloete. A study of the difference between partial derivative and stochastic neural network sensitivity analysis for applications in supervised pattern classification problems. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics, 2004.*, volume 7, pages 4283–4288. IEEE, 2004.

- [57] J. G. Polhill and M. K. Weir. An approach to guaranteeing generalisation in neural networks. *Neural networks*, 14(8):1035–1048, 2001.
- [58] J. R. Quinlan. Improved use of continuous attributes in c4. 5. *Artificial Intelligence Research*, 4:77–90, 1996.
- [59] C. Rao and S. Mitra. Generalized inverse of a matrix and its applications. *J. Wiley, New York*, 1971.
- [60] S. Raudys and V. Pikelis. On dimensionality, sample size, classification error, and complexity of classification algorithm in pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(3):242–252, 1980.
- [61] B. Roy and R. Slowinski. Criterion of distance between technical programming and socio-economic priority. *RAIRO. Recherche opérationnelle*, 27(1):45–60, 1993.
- [62] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.
- [63] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.
- [64] J. Sahami, R. Dougherty, and M. Kohavi. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9-12, 1995*, page 194. Morgan Kaufmann, 1995.
- [65] D. Serre. *Matrices: Theory and applications*, volume 216. Springer, 2010.

- [66] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.
- [67] D. Shen, J. Zhang, J. Su, G. Zhou, and C. Tan. Multi-criteria-based active learning for named entity recognition. In *Proceedings. 42nd Annual Meeting on Association for Computational Linguistics*, page 589. Association for Computational Linguistics, 2004.
- [68] S. Siegel. Nonparametric statistics for the behavioral sciences. *Journal of Nervous & Mental Disease*, 125(3):497, 1956.
- [69] H. Simon. The sciences of the artificial. *MIT Press Books*, 1, 1996.
- [70] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar. Ranking a random feature for variable and feature selection. *Machine Learning Research*, 3:1399–1414, 2003.
- [71] M. Sugeno. Theory of fuzzy integrals and its applications. 1974.
- [72] S. Suresha, R. V. Babub, and H. Kimc. No-reference image quality assessment using modified extremelearningmachine classifier. *Applied Soft Computing*, 9(2):541–552, 2008.
- [73] J. Wang and Z. J. Wang. Using neural networks to determine sugeno measures by statistics. *Neural Networks*, 10(1):183–195, 1997.
- [74] R. Wang and S. Kwong. Active learning with multi-criteria decision making systems. *Pattern Recognition*, 47(9):3106–3119, 2014.
- [75] W. Wang, Z. Wang, and G. J. Klir. Genetic algorithms for determining fuzzy measures from data. *Journal of Intelligent and Fuzzy Systems*, 6(2):171–183, 1998.

- [76] X. Wang, A. Chen, and H. Feng. Upper integral network with extreme learning mechanism. *Neurocomputing*, 74(16):2520–2525, 2011.
- [77] X. Wang, Y. He, L. Dong, and H. Zhao. Particle swarm optimization for determining fuzzy measures from data. *Information Sciences*, 181(19):4230–4252, 2011.
- [78] X.-Z. Wang, C.-G. Li, D. S. Yeung, S. Song, and H. Feng. A definition of partial derivative of random functions and its application to rbfn sensitivity analysis. *Neurocomputing*, 71(7):1515–1526, 2008.
- [79] Z. Wang, R. Yang, K. H. Lee, and K. S. Leung. The choquet integral with respect to fuzzy-valued signed efficiency measures. In *2008 IEEE International Conference on Fuzzy Systems*, pages 2143–2148. IEEE, 2008.
- [80] B. Widrow and M. A. Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.
- [81] W. Xi-Zhao, S. Qing-Yan, M. Qing, and Z. Jun-Hai. Architecture selection for networks trained with extreme learning machine using localized generalization error model. *Neurocomputing*, 102:3–9, 2013.
- [82] X. Xu, J. Martel, and B. Lamond. A multiple criteria ranking procedure based on distance between partial preorders. *European Journal of Operational Research*, 133(1):69–80, 2001.
- [83] D. S. Yeung, P. P. Chan, and W. W. Ng. Radial basis function network learning using localized generalization error bound. *Information Sciences*, 179(19):3199–3217, 2009.

- [84] D. S. Yeung, W. W. Ng, D. Wang, E. C. Tsang, and X.-Z. Wang. Localized generalization error model and its application to architecture selection for radial basis function neural network. *IEEE Transactions on Neural Networks*, 18(5):1294–1305, 2007.