REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

IMPROVED EXTREME LEARNING MACHINES AND APPLICATIONS

MOHANAD ABD SHEHAB AL_KARAWI

Ph.D. THESIS
DEPARTMENT OF ELECTRONICS AND COMMUNICATIONS ENGINEERING
PROGRAM OF ELECTRONICS ENGINEERING

ADVISER
ASSIST. PROF. DR. NİHAN KAHRAMAN

İSTANBUL, 2018

REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

IMPROVED EXTREME LEARNING MACHINES AND APPLICATIONS

A thesis submitted by MOHANAD ABD SHEHAB AL_KARAWI in partial fulfillment of the requirements for the degree of **DOCTOR OF SCIENCE** is approved by the committee on 06.12.2018 in Department of Electronics and Communications Engineering, Electronics Engineering Program.

Thesis Adviser

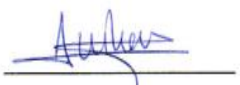
Assist. Prof. Dr. Nihan KAHRAMAN

Yıldız Technical University

Approved By the Examining Committee

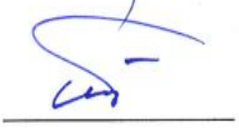
Assist. Prof. Dr. Nihan KAHRAMAN

Yıldız Technical University



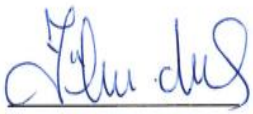
Prof. Dr. Tülay YILDIRIM

Yıldız Technical University



Prof. Dr. Zümray DOKUR

Istanbul Technical University



Assist. Prof. Dr. Bülent Bolat

Yıldız Technical University

Assist. Prof. Dr. Berk Gökberk

MEF University

This work was supported by the Ministry of Higher Education and Scientific Research, Al-Mustansiriyah University, Baghdad, Iraq.

ACKNOWLEDGEMENTS

Firstly, all praise and thanks are due to Allah for his countless favors from beginning of my life to end.

Foremost, I would like to express my sincere gratitude to my academic advisor: Dr. Nihan KAHRAMAN for the continuous support of my PhD study, her keen supervision, patience, advice, motivation, encouragement, and immense knowledge. This dissertation would not have been accomplished without her tremendous help, guidance, and wealth of knowledge. It has been a great honor and a pleasure to work with her.

I would like to acknowledge the help I have received from the generous academic staff in the department of electronics and communications engineering at Yildiz Technical University. I am hugely appreciative to all of them.

I would like to offer my special thanks to professors of my committee members for taking time out of their busy schedules and accepting to review my work and give their valuable feedback and comments.

I would like to express my profound love, respect and appreciation to the symbols of bestowal and my role models in life, my late father Abd Shehab (may Allah have mercy on him) and my mother (may Allah bless and protect her). Also, I would like to express my great thanks and appreciation to my brother Dr. Muayad Abd Shehab and my sisters for their prayers, advices, and encouragement. I would have never become the person that I am today without them.

Last but not least, I would also like to extend my appreciation and gratitude to my wife and my children, who are my life flowers and fruit of my happiness, for their encouragement and continuous support.

October, 2018

Mohanad Abd Shehab

TABLE OF CONTENTS

Page

LIST OF SYMBOLS xi
LIST OF ABBREVIATIONS xiii
LIST OF FIGURES xv
LIST OF TABLES xvii
ABSTRACT xix
ÖZET xxii

CHAPTER 1

INTRODUCTION 1

1.1 General Literature Review 1
1.2 Motivation and Objectives 3
1.3 Major Contributions 4
1.4 Thesis Organization 7

CHAPTER 2

ELM METHOD AND PRELIMINARIES 9

2.1 Introduction 9
2.2 Characteristics of the Standard ELM Approach 10
2.3 ELM Theory and Feature Mapping 14
2.4 ELM Models 15
2.5 ELM Optimizing Approaches 17
2.5.1 Incremental Approaches 18
2.5.1.1 Incremental ELM (I-ELM) 19
2.5.1.2 Enhanced Incremental ELM (EI-ELM) 20
2.5.1.3 Convex Incremental ELM (CI-ELM) 20
2.5.1.4 Error Minimized ELM (EM-ELM) 21
2.5.2 Pruning Approaches 22
2.5.3 Regularization Approaches 23

CHAPTER 3

OPTIMUM, PROJECTED, AND L2-TIKHONOV REGULARIZED EXTREME LEARNING MACHINE METHODS 24

3.1 Introduction 24
3.2 ELM Structure Preliminary 25

3.3 Standard ELM with Optimum Hidden Nodes (OELM) 26
3.4 Projection ELM 28
3.5 Regularized ELM (RELM) 31
3.6 ELM vs. RELM Comparison 34
3.7 L-curve with Regularization ELM 35
3.8 Experimental Results 38
3.8.1 Standard ELM with Optimum Hidden Nodes (OELM) 39
3.8.2 Projection ELM (PELM) 41
3.8.3 L-curve Tikhonov Regularization ELM (TRELM) 42
3.9 Minimized RELM 45
3.9.1 Leave-One-Out Cross-Validation (LOO-CV) 46
3.9.2 RELM with PRESS and SVD 47

CHAPTER 4

GRAPH BASED EXTREME LEARNING MACHINE WITH L21-NORM REGULARIZATION 51

4.1 Introduction 51
4.2 Related Works 52
4.2.1 Feature Extraction 52
4.2.2 Feature Selection 53
4.2.3 The Regularization 54
4.2.4 Graph Laplacian Regularization 56
4.3 Graph Regularized ELM (GELM) 58
4.4 The Proposed L21GELM 60
4.5 Algorithm Headlines 62
4.6 Experimental Study 63
4.6.1 Face recognition datasets 64
4.6.2 Object recognition datasets 65
4.6.3 Computing Complexity Analysis 70
4.6.4 Parameters Sensitivity Analysis 71
4.6.5 L21GELM with Local and Manifold Subspace Learning 73
4.6.6 L21GELM with Noisy Environment 75
4.7 Comparison with other Classifiers 76
4.8 Unified ELM with L21GELM Classifier 77

CHAPTER 5

EXTREME LEARNING MACHINE WITH THE ENSEMBLE 78

5.1 Introduction 78
5.2 Ensemble Classifier Schemes 79
5.3 Ensemble Combiner Schemes 80
5.3.1 Algebraic combiners 81
5.3.2 Voting based methods 81
5.3.3 Other combination rules 82
5.4 Ensemble Learning 82
5.5 Review of ELM Ensemble 84
5.6 The Proposed Weighted Majority Voting 86

5.7 The Proposed Ensemble Pruning 87
5.8 The Proposed Fast RELM with Weighted Voting Ensemble (WV-RELM) ..
..... 88
5.8.1 Generation Approach of the Fast HAT-matrix 89
5.8.2 Generation Approach of the Fast Output Weights β 91
5.9 Computational Complexity of the RELM with Fast PRESS 95
5.10 Experimental Study 97

CHAPTER 6

CONCLUSION AND FUTURE WORK 106

6.1 Conclusions 106
6.2 Future works 108

REFERENCES 112
CURRICULUM VITAE 125

x, y	vectors
X, Y	matrices
$\hat{y}$	estimation of y
x_i	i^{th} sample
x_{ij}	i^{th} sample, j^{th} entry
N	number of input features
n	dimension of the input samples (attributes)
d	reduced dimension of the input samples
m	number of output classes
L	Number of hidden neurons
L_{opt}	optimum number of hidden nodes
I	identity matrix
w_i	hidden layer weights of neuron i
b_i	hidden layer bias of neuron i
$f(x)$	feature mapping function
$G(x)$	transfer or activation function
$h(x)$	hidden layer output function
h_{K+1}	the output of $K+1$ hidden node
H	hidden layer output matrix
$H^{\dagger}$	Moore-Penrose pseudo-inverse of H
HAT	transformation matrix
β	hidden layer output weights
$\hat{\beta}$	least-square hidden layer output weights solution
β_{opt}	optimum hidden layer output weights
T, t_j	output target
Tr	Matrix trace
$\|.\|$	absolute value
$\|\|.\|\|, \|\|.\|\|_2$	L2 norm
$\|\|.\|\|_{21}$	L21 norm
e	error
ε	root mean square error
ξ_i	least square error
MSE^{PRESS}	PRESS minimum square error
MSE_{CV}	cross validation minimum square error
U	orthogonal matrix of SVD factorization
D	diagonal matrix with singular values (eigen value in diagonal matrix)

V	orthogonal matrix of SVD factorization
Ω_{ELM}, $K(x_i, x_j)$	ELM kernel of $h(x_i, x_j)$ feature spaces
λ, λ_1, λ_2	regularization parameters
λ_{opt}	optimum regularization parameter
Ł	graph Laplacian matrix
W	weight matrix of adjacency graph
D	diagonal matrix of adjacency graph
SF	stability factor
α_i	final weights assignment of ensemble ELM
σ	standard deviation
σ_{max}	maximum eigenvalue
σ_{min}	minimum eigenvalue
$error_{LOO}$	leave-one-out error

ANN	Artificial Neural Network
BP	Backpropagation
CV	Cross Validation
ELM	Extreme Learning Machine
EVELM	Evolutionary Voting Extreme Learning Machine
Fast PRESS	Fast PREdiction Sum of Squares
FNN	Feedforward Neural Networks
G/KDA	Generalized or Kernel Discriminant Analysis
GA	Genetic Algorithm
GCV	Generalized Cross-Validation
GELM	Graph ELM
GPU	Graphics Processing Unit
KPCA	Kernel Principal Component Analysis
L21GELM	Graph based ELM with L_{21}-norm Regularization
LDA	Linear Discriminant Analysis
LGE	Linear Graph Embedding
LOO	Leave One Out
LOO-CV	Leave One Out-Cross Validation
LPP	Linearity Preserving Projection
LR	Logistic Regression Classifier
LSE	Least Square Error
MSE	Mean Square Error
NNC	Nearest Neighbor Classifier
NPE	Neighbourhood Preserving Embedding
OAA	One-Against-All
OELM	Optimum Hidden Nodes ELM
OLS	Orthogonal Least-Squares
PCA	Principal Component Analysis
PELM	Projection ELM
PRESS	PREdiction Sum of Squares
PSO	Particle Swarm Optimization
RBF	Radial Base Function
RELM	Regularized ELM
RMSE	Root Mean Square Error
SL	Subspace Learning
SLFN	Single-Layer Feedforward Network
SRC	Sparse Representation based Classification
SSE	Sum Square Error
SVD	Singular Value Decomposition

SVM	Support Vector Machine
TR	Tikhonov-Regularized
TRELM	Tikhonov Regularization ELM
TROPELM	Tikhonov Regularization Optimally Pruned ELM
UCI	machine learning repository of University of California at Irvine
VELM	Voting Extreme Learning Machine
WVELM	Weighted Voting Extreme Learning Machine
WVRELM	Weighted Voting Regularized Extreme Learning Machine

LIST OF FIGURES

Page

Figure 2.1 SLFN Feedforward Network Architecture of the ELM Model 11
Figure 2.2 SLFN construction based on Kwok's theory 18
Figure 3.1 The relationship between average testing RMSE and the hidden nodes number for a certain data 26
Figure 3.2 Double-stage learning method: dimensionality reduction approach 29
Figure 3.3 Single-stage learning method: base projection approaches 29
Figure 3.4 Accuracy vs. hidden nodes number relationship for the breast cancer dataset 34
Figure 3.5 RMSE vs. hidden nodes number relationship for the wpbc_dataset 34
Figure 3.6 An L-curve 35
Figure 3.7 The relationship between testing accuracy with (a). hidden nodes number (b). the k-fold number for Pima Indians Diabetes data set 40
Figure 3.8 The relationship between testing accuracy with (a). hidden nodes number (b). the k-fold number for Wine data set 40
Figure 3.9 The relationship between testing accuracy with (a). hidden nodes number (b). the k-fold number for Segment data set 41
Figure 3.10 L-curve regularization method for Segment image data base 43
Figure 3.11 All ELM approaches accuracies for USPS dataset 44
Figure 3.12 All ELM approaches accuracies for AR dataset 44
Figure 4.1 Different regularization types 54
Figure 4.2 ORL, Yale, Extended Yale B and AR - face recognition dataset samples 64
Figure 4.3 An example of images of two-object recognition (COIL20 and USPS) datasets 65
Figure 4.4 ELM methods comparison testing accuracies for face recognition on dataset ORL 68
Figure 4.5 ELM methods comparison testing accuracies for face recognition on dataset YALE 68
Figure 4.6 ELM methods comparison testing accuracies for face recognition on dataset YALE-B 68
Figure 4.7 ELM methods comparison testing accuracies for face recognition on dataset COIL20 69
Figure 4.8 ELM methods comparison testing accuracies for face recognition on dataset USPS 69
Figure 4.9 L21GELM performance for different trains per subject against the incremental of hidden nodes on datasets ORL 69
Figure 4.10 L21GELM performance for different trains per subject against the incremental of hidden nodes on datasets YALE 69

Figure 4.11 L21GELM performance for different trains per subject against the incremental of hidden nodes on datasets YALE-B 70
Figure 4.12 Comparison of time for different training samples of ELM models on the ORL dataset 70
Figure 4.13 Comparison of time for different training samples of ELM models on the YALE dataset 70
Figure 4.14 RELM recognition rate to the variations of the λ parameter for ORL 71
Figure 4.15 RELM recognition rate to the variations of the λ parameter for YALE ... 71
Figure 4.16 RELM recognition rate to the variations of the λ parameter for YALE-B 71
Figure 4.17 λ_1& λ_2 parameters sensitivity analysis of L21GELM on ORL (Train 2).. 72
Figure 4.18 λ_1& λ_2 parameters sensitivity analysis of L21GELM on YALE (Train 3)72
Figure 4.19 λ_1& λ_2 parameters sensitivity analysis of L21GELM on COIL20 (Train 8) 72
Figure 4.20 Testing accuracies of AR and USPS datasets with t-iterations 73
Figure 4.21 Testing accuracy for different noise levels on ORL (Train 2) dataset 76
Figure 4.22 Testing accuracy for different noise levels on YALE (Train 5) dataset ... 76
Figure 5.1(a, b, c) ELM classifiers using three different random parameters 83
Figure 5.2 Ensemble system of m-multiple classifiers/models 83
Figure 5.3 Computational complexity comparison for 20 trails, N=1000 96
Figure 5.4 Computational complexity comparison for 50 trails, N=1000 96
Figure 5.5 ELM methods comparison testing accuracies for face recognition on dataset ORL 102
Figure 5.6 ELM methods comparison testing accuracies for face recognition on dataset YALE 102
Figure 5.7 ELM methods comparison testing accuracies for face recognition on dataset YALE-B 102
Figure 5.8 ELM methods comparison testing accuracies for face recognition on dataset COIL20 102
Figure 5.9 ELM methods comparison testing accuracies for face recognition on dataset USPS 102
Figure 5.10 WV-RELM performance for different trains per subject verses the incremental of (L) on ORL 103
Figure 5.11 WV-RELM performance for different trains per subject verses the incremental of (L) on YALE 103
Figure 5.12 λ-parameters sensitivity analysis of RELM on ORL with different trains 104
Figure 5.13 λ-parameters sensitivity analysis of RELM on YALE with different trains 104
Figure 5.14 λ-parameters sensitivity analysis of RELM on YALE-B with different trains 104
Figure 5.15 The computational complexity for the waveform database against the number of the hidden nodes 104

LIST OF TABLES

Page

Table 2.1 Most common ELM activation functions 13
Table 3.1 OELM performance results for classification samples 39
Table 3.2 OELM performance results for regression samples 39
Table 3.3 PELM performance results 42
Table 3.4 TRELM performance results for classification data sets 43
Table 3.5 TRELM performance results for regression data sets 43
Table 3.6 Performance comparison for AR data sets 45
Table 3.7 Performance comparison for USPS data sets 45
Table 4.1 Details of the face and object datasets 65
Table 4.2 Variation of some samples per subject for the ORL_database 66
Table 4.3 Performance (mean accuracy± standard deviation) % for ELM methods on ORL dataset 67
Table 4.4 Performance (mean accuracy± standard deviation) % for ELM methods on YALE dataset 67
Table 4.5 Performance (mean accuracy± standard deviation) % for ELM methods on YALE-B dataset 67
Table 4.6 Performance (mean accuracy) % for ELM methods on COIL-20 dataset 69
Table 4.7 Performance (mean accuracy) % for ELM methods on USPS dataset 69
Table 4.8 L21GELM accuracy of different training sampling on ORL dataset for L=500 hidden nodes 74
Table 4.9 L21GELM accuracy of different training sampling on YALE dataset for L=500 hidden nodes 74
Table 4.10 L21GELM accuracy of different training sampling on YALE-B dataset for L=500 hidden nodes 74
Table 4.11 Testing accuracy for different noise levels on ORL (Train 2) dataset 76
Table 4.12 Testing accuracy for different noise levels on YALE (Train 5) dataset 76
Table 4.13 Different state of the art classifier accuracies (%) on AR face recognition dataset for different dimension 77
Table 5.1 The computational complexity of the RELM with normal LOO-CV and fast PRESS algorithms for 20 trails 96
Table 5.2 The computational complexity of the RELM with normal LOO-CV and fast PRESS algorithms for 50 trails 96
Table 5.3 Details of the UCI, Face, and Object datasets 97
Table 5.4 Performance comparisons with different ELM methods 98
Table 5.5 Performance (mean accuracy± standard deviation) % for ELM methods on ORL dataset 100
Table 5.6 Performance (mean accuracy± standard deviation) % for ELM methods on YALE dataset 101

Table 5.7 Performance (mean accuracy± standard deviation) % for ELM methods on YALE-B dataset 101
Table 5.8 Performance (mean accuracy) % for ELM methods on COIL-20 dataset 101
Table 5.9 Performance (mean accuracy) % for ELM methods on USPS dataset ... 101
Table 5.10 Different state of the art classifier accuracies (%) on YALE_B face recognition dataset for different dimension..... 105
Table 5.11 Different state of the art classifier accuracies (%) on AR face recognition dataset for different dimension..... 105

ABSTRACT

IMPROVED EXTREME LEARNING MACHINES AND APPLICATIONS

Mohanad Abd Shehab

Department of Electronics and Communications Engineering

Doctorate Thesis

Adviser: Assist. Prof. Dr. Nihan KAHRAMAN

Extreme Learning Machine (ELM) is a simple, fast, and precise machine learning algorithm but still leaves much area for improvement. In standard ELM, the hidden nodes are randomly initiated and fixed without iteratively tuning. However, the optimal number of hidden layer neurons is the key of ELM generalization performance where initializing this number randomly by trial and error is not reasonably satisfied. Using different techniques such as Leave-One-Out Cross Validation (LOO-CV) to optimize the hidden layer size is costly due to the repeatedly training and evaluation of the model on the whole data set with the exception of one sample. In this work, an efficient and reliable approach called Optimum ELM (OELM) was developed to firstly determine the minimum hidden layer size that yields an optimum generalization and performance. Moreover, another approach that exploits the merits of orthogonal projections with Singular Value Decomposition (SVD) was proposed in order to tackle the problems of randomness and correlated features in the input data. This approach that is called hereafter Projected ELM (PELM) achieves more than 2% advance in average accuracy.

Tikhonov Regularization in the form of squared loss penalty with ELM (TRELM) that regularizes and improves the matrix computations utilizing L-curve criterion and SVD was implemented as a most appropriate candidate to this work. Unlike the most iterative

methods, L-curve can estimate optimum regularization parameter by illustrating a curve with flexible points that represents the trade-off between minimizing the training error and the residual of output weight. The proposed TRELM was tested based on three different scenarios of data sizes: small, moderate, and big datasets.

Due to simplicity, robustness and less time consuming of OELM and PELM, it is recommended to use them in small and even moderate data. TRELM demonstrated that enhancing the ELM performance needs to enlarge the size of hidden nodes. As a result, in big data, increasing hidden node number in TRELM is necessary which concurrently leads to better accuracy.

A novel extension of ELM termed Graph-based ELM with L_{21}-norm Regularization (L21GELM), which merges efficient approaches, was developed to improve the robustness and compactness over the Graph Extreme Learning Machine (GELM).

Many linear and nonlinear subspace learning (SL) methods were adopted herein to reduce the similar properties of input neighborhood pixels and hence extract only the discriminative features. Due to the nonlinear mapping of the activation functions and some SL approaches that may destroy the local consistency properties in the dataset, the graph Laplacian is intervened into the optimization function as a demand to exploit the class similarity advantages of intrinsic construction to enhance the overall model accuracy. Finally, an iterative algorithm with proved convergence for solving L_{21}-norm optimization problem was involved to eliminate the irrelevant features and to decrease the outlier's effects.

The outcomes demonstrated that the L21GELM was more general and accurate (more than 3% advance in average accuracy) than other ELM variants, also it can handle large-scale data problems with good feature selection and less sensitive to noisy environments.

Precisely evaluating the compactness of ELM is difficult due to the randomness in hidden layer nodes number, weight and bias values. To reduce the randomness, resultant overfitting and large variance of the ELM, a selective Weighted Voting ensemble model based on Regularized ELM which called (WVRELM) is investigated. It can strongly enhance the overall performance including accuracy, variance and time consumption. Optimizing the regularization parameter of the RELM using Leave-One-Out Cross Validation (LOO-CV) method is a time-consuming approach. Thereby, efficient PREdiction Sum of Squares (PRESS) criteria that utilizes Singular Value Decomposition was proposed to address the slow execution issue with low human intervention. Moreover, a simple ensemble pruning approach based on the eigenvalues for the input

weight matrix was developed owing to its capability to produce the discriminative parameters and select the optimal ensembles. As an advantage, in this thesis, the weights assignment for base classifiers were based on the same PRESS error metric and the variation of the output weight vector (β) solutions used for the RELM, thus, it could reduce computational cost and space requirement.

The outcomes demonstrated that the average accuracy of WV-RELM was approximately 3% higher than other ELM variants, stable with low variance, faster with 2.8 folds than RELM with LOO-CV.

Various well-known benchmark datasets such as face with facial expression and object recognition as well as different state-of-the-art learning approaches were examined and compared with the proposed approaches. Consequently, the data representations handled by various feature spaces in training data enhanced significantly the classification efficiency as compared to other related popular algorithms.

Keywords: extreme learning machines, singular value decomposition, Tikhonov regularization, optimum hidden nodes, dimensional reduction (subspace learning), ensemble learning, weighted majority voting.

ÖZET

GELİŞTİRİLMİŞ AŞIRI ÖĞRENME MAKİNELERİ VE UYGULAMALARI

Mohanad Abd Shehab

Elektronik ve Haberleşme Mühendisliği Bölümü
Doktora Tezi

Tez Danışmanı: Dr. Öğr. Üyesi Nihan KAHRAMAN

Aşırı Öğrenme Makineleri (AÖM) basit, hızlı ve hassas bir makine öğrenme algoritmasıdır, ancak iyileştirme için hala çok alan bırakır. Standart AÖM 'de, gizli düğümler rasgele bir şekilde ayarlanmadan rasgele başlatılır ve sabitlenir. Bununla birlikte, gizli katman nöronlarının optimal sayısı, bu sayının rasgele bir şekilde deneme ve hata ile başlatılmasının makul ölçüde tatmin edici olmadığı AÖM genelleme performansının anahtarıdır. Gizli katman boyutunun optimize edilmesi için Ayrı Bırakılan Çapraz Doğrulama (ABÇD) gibi farklı tekniklerin kullanılması, bir örnek dışında tüm veri seti üzerinde modelin tekrar tekrar eğitilmesi ve değerlendirilmesi nedeniyle maliyetlidir. Bu çalışmada, öncelikle optimum bir genelleme ve performans sağlayan minimum gizli katman boyutunu belirlemek için Optimum AÖM (OÜÖM) adı verilen verimli ve güvenilir bir yaklaşım geliştirilmiştir. Ayrıca, giriş verilerindeki rastlantısallık ve bağlantılı özellikler ile ilgili problemlerin üstesinden gelmek için, Tekil Değer Ayrışımı (TDA) ile ortogonal projeksiyonların yararlarını açıklayan bir başka yaklaşım önerilmiştir. Bundan sonra uzayı değiştirilmiş ÜÖM (UD- AÖM) olarak adlandırılan bu yaklaşım, ortalama doğrulukta % 2'den daha fazla ilerleme sağlar.

Tikhonov Düzenleyici yöntemi ve TDA kullanarak matris hesaplamalarını düzenli hale getiren ve geliştiren ÜÖM (TD AÖM) ile kare kayıp cezası şeklinde yeniden yapılanma, bu çalışmaya en uygun aday olarak uygulanmıştır. En yinelemeli yöntemlerden farklı olarak, L-eğrisi, eğitim hatasını ve çıktı ağırlığının kalanını en aza indirgemek arasındaki dengeyi temsil eden esnek noktalara sahip bir eğri göstererek optimum regülasyon parametresini tahmin edebilir. Önerilen TDAÖM, küçük, orta ve büyük veri setleri olmak üzere üç farklı veri boyutuna dayanarak test edilmiştir.

Basitlik, sağlamlık ve OAÖM ve UD-AÖM'in daha az zaman harcaması nedeniyle, bunların küçük ve hatta ılımlı verilerde kullanılması tavsiye edilir. TD AÖM, AÖM performansının, gizli düğümlerin boyutunu büyütmek için ihtiyaç duyduğunu göstermiştir. Sonuç olarak, büyük verilerde, TD AÖM'de artan gizli düğüm sayısı daha fazla doğruluk sağlayabilmek adına artmaktadır.

Grafik Aşırı Öğrenme Makinesi (GAÖM) üzerinde sağlamlığı ve kompaktlığı geliştirmek için verimli yaklaşımları birleştiren L21-norm Regülasyonu (L21GAÖM) ile AÖM'nin yeni bir grafik taanlı AÖM geliştirilmiştir.

Birçok lineer ve lineer olmayan alt uzay öğrenimi (AUÖ) yöntemleri, girdi mahalli piksellerin benzer özelliklerini azaltmak ve böylece sadece ayırt edici özellikleri ayıklamak için burada kabul edilmiştir. Aktivasyon fonksiyonlarının doğrusal olmayan haritalanması ve veri kümesindeki yerel tutarlılık özelliklerini bozabilecek bazı AUÖ yaklaşımları nedeniyle, grafik Laplacian, genel modelin geliştirilmesi için içsel yapının sınıf benzerlik avantajlarından yararlanmak için bir talep olarak optimizasyon fonksiyonuna müdahale eder. Son olarak, ilgisiz özellikleri ortadan kaldırmak ve aykırı etkilerini azaltmak için L21-norm optimizasyon problemini çözmek için kanıtlanmış yakınsama ile yinelemeli bir algoritma yer almıştır.

Sonuçlar, L21GAÖM'nin diğer AÖM değişkenlerinden daha genel ve doğru (ortalama doğrulukta % 3'ten fazla ilerleme) olduğunu, ayrıca iyi özellik seçimi ve gürültülü ortamlara karşı daha az hassas olan büyük ölçekli veri sorunlarının üstesinden gelebileceğini göstermiştir. AÖM'nin kompaktlığını tam olarak değerlendirmek, gizli katman düğüm sayısı, ağırlık ve önyargı değerlerindeki rastlantısallık nedeniyle zordur. Elde edilen rastlantısallığı azaltmak için, AÖM 'nin aşırı yükü ve büyük bir varyansı olan, (SAOAÖM) adlı Düzenlenmiş AÖM'ye dayanan bir seçici Ağırlıklı Oylama modeli (AOMDAÖM) incelenmiştir. Doğruluk, varyans ve zaman tüketimi de dahil olmak üzere

genel performansı güçlü bir şekilde artırabilir. Ayrı Bırakılan Çapraz Doğrulama yöntemini kullanarak DAÖM'nin düzenli hale getirme parametresini optimize etmek zaman alıcı bir yaklaşımdır. Böylelikle, düşük insan müdahalesi ile yavaş uygulama sorununu ele almak için Tekil Değer Ayrıştırma'yı kullanan verimli tahminle kareler toplamı kriterleri önerilmiştir. Ayrıca, ayırt edici parametreler üretme ve en uygun toplulukları seçebilme yeteneği sayesinde girdi ağırlığı matrisi için özdeğerlere dayanan basit bir topluluk budama yaklaşımı geliştirilmiştir. Bir avantaj olarak, bu tezde, temel sınıflandırıcılar için ağırlıkların atanması aynı PRESS hata metriğine ve DAÖM için kullanılan çıktı ağırlık vektörü (β) çözümlerine dayandırılmış, bu sayede hesaplama maliyeti ve yer gereksinimini azaltılabilmiştir.

Sonuçlar, AOMDAÖM 'in ortalama doğruluğunun, düşük varyanslı, diğer ELM türlerine göre yaklaşık % 3 daha yüksek olduğunu, ABÇD ile DAÖM'den 2.8 kat daha hızlı olduğunu göstermiştir.

Yüz ifadesi ve nesne tanıma yanı sıra farklı son teknoloji öğrenme yaklaşımları gibi çeşitli tanınmış kıyaslama verileri incelenmiş ve önerilen yaklaşımlarla karşılaştırılmıştır. Sonuç olarak, eğitim verisinde çeşitli özellik alanları tarafından ele alınan veri sunumları, diğer ilgili popüler algoritmalara kıyasla sınıflandırma verimliliğini önemli ölçüde artırmıştır.

Anahtar Kelimeler: aşiri öğrenme makineleri, tekil değer ayrışımı, Tikhonov düzenleyici, optimum gizli düğüm, boyut azaltma, topluluk öğrenmesi, ağırlıklı oylama.

CHAPTER 1

INTRODUCTION

1.1 General Literature Review

Machine learning (ML) is a collection of computational algorithms that use previous experiences to interpret and reveal the unknown or hidden relationships in the real empirical data. It is closely related to other studies including statistics, pattern recognition, data mining and artificial intelligence. The learning approaches have found wide applications in the fields of engineering, business, and science. The machine learning techniques may face many challenges; a common concern is the learning of a complex pattern from a given training data, which is usually a subset of all possible feature samples, so as to generalize the massive unidentified data.

Artificial Neural Network (ANN) is one of the powerful learning approaches of machine learning techniques that have attracted numerous researchers and have played an important role on function approximation (regression analysis) [1, 2], classifications or data processing (such as clustering, filtering, etc.) [3–5], pattern recognitions [6–8], system controls [9, 10], and so on.

ANNs are imitative to human brain which can simulate and model any arbitrary function through learning from observed data [11–13]. Nevertheless, some aspects need to be taken into account before employing ANNs:

(1) Selection of models: Choosing a most suitable model is essential for using ANNs, it depends on the data representations and applications. An overly complex model may lead to overfitting problem, while a simple model with few parameters may underfit the data.

(2) Learning algorithm: Many algorithms exist and available for training ANN models. However, selecting and tuning the algorithm for training data may require huge experiments.

Based on the connections between neurons and different statuses fitting, different ANN architectures are available; for example single-layer feedforward, multi-layer feedforward, and recurrent neural networks. Among these architectures, Feedforward Neural Networks (FNNs) are most common and have been extensively investigated and applied due to their simplicity and universal approximation capabilities.

Single-hidden Layer Feedfoward Network (SLFN) is the simplest kind of the FNNs. It includes three layers which are input layer to insert and adjust the input information, hidden layer to treat and process the data and lastly the output layer to represent the output value of the system. In the traditional SLFN learning algorithms, iterative tuning of parameters in the hidden layer is required. Three major approaches are available to train a SLFN [14] which are: 1) gradient descent based method; 2) least squares based method; and 3) standard optimization based method. The backpropagation (BP) [15] is an example of the above mentioned algorithms which based on gradient-descent. However, the training steps in these algorithms are tedious and time consuming due to their slow convergence rates. Moreover, such algorithms often suffer from common shortcomings such as local minimum.

A variant approach named Extreme Learning Machine (ELM) has been recently introduced and developed for learning the SLFN [16], it has attracted a large number of research communities owing to its superiority over other well-known sophisticated ANN algorithms such as Back-Propogation (BP) [17] and Fuzzy Neural Networks (FNN) [18].

Extreme learning machines (ELMs) were proposed originally as learning algorithms for the neural networks [16], and then extended to the "generalized" SLFNs [19, 20]. The main features of ELMs lies in two valuable aspects: First, the hidden and output weights of the network are determined independently which is beneficial for example in facilitating the implementation of parallel or distributed systems, in addition to its well performance in most classification problems. Second, the hidden layer parameters are random generating instead of iteratively tuning as in the common fashioned way of training SLFNs. Choosing random hidden neuron parameters (weight, bias, and size), which remain unchanged during the learning phase, will lead to very low computational complexity and less human intervention. Moreover, producing the universal approximation with better generalization capability is guaranteed which subsequently eases the analytical determination of output weight by employing a proper loss function such as hinge or squared loss to obtain the prediction error. The training of output weights

is essentially a regularized least squares (ridge regression in statistics) problem which can be solved efficiently in a closed form; hence, ELMs gained great importance. They have been successfully applied in various classification applications and function approximation with extreme fast speed and convenience in implementation.

The training speed of ELM can be of thousand times faster than the classical approaches [16, 19] such as back-propagation algorithm (BP) and support vector machine (SVM). Additionally, ELM is very simple and tends to good generalization performance via achieving the smallest for both the norm of output weights and the training error [5].

However, the adequately of the ELM performance is largely dependent on the network architecture and hence should be chosen appropriately. For any application, a smaller network than what is required is probably not able to learn the problem well, while a larger network may lead to overfitting [21]. Both cases result in poor generalization performance.

1.2 Motivation and Objectives

ELM is of easy implementation because of the simple structure and unique parameter selection strategy. However, some of the user-defined parameters still have vital impacts on the performance. Among those that plays the most critical role is the network number of hidden neurons L. Other user choices, such as activation functions and learning rules, have small effects on the network performance since the available options are limited. In contrast, the choices of L typically ranges from several to thousands that has greatly influence of the ELM performance, also the problems of underfitting and overfitting are commonly associated with improper L selection [19, 22]. Therefore, to overtake the issue of determining hidden neurons L precisely, several ELM models have been proposed which can properly select the network topology according to certain criteria. For more details, see section 2.4.

The ease of implementation indeed offers the potentiality to convert ELM into an automatic ANN learning algorithm, since the only obstacle is the determination of L, which is much easier comparing for example to the stopping criterion, learning rate and number of epochs required in BP. Although many previously proposed algorithms claim to be automatic, they essentially just transfer the determination of one set of parameters to another, and users still have to make rigorous choices that may have big impacts to the

algorithm behavior and performance. The real automatic learning algorithm should have the following properties:

- No or minimal user intervention.
- The user choices should have limited impact on the final performance and decision.

In addition to the overall performance and automation, which are substantially influenced by the hidden layer size and slightly the learning approaches that optimize different ELM constraints, ELM approaches also have significant drawbacks, which are:

- The method of extracting and calculating the output weights is a kind of least squares minimization method, which may easily suffer from computational limitations like matrix singularity.
- ELM is sensitive to noisy, outliers and unbalanced input data.
- Lastly, the compactness of ELM under the randomness in hidden layer nodes number, weight and bias values is not easily evaluated.

1.3 Major Contributions

The objective of this thesis is to study and investigate the most significant ELM problems and difficulties that restrict the optimum performance of ELM variant algorithms, then provide reliability issues that can overcome these difficulties and enhance the ELM solution keys. Theories of the recommended solutions are depending on various knowledge bases including statistical, optimization, regularization, and mixing with another mathematical or machine learning approaches.

In this thesis, a series of ELM based algorithms that aims at improving the performance and achieving the true automatic algorithms which meet the proposed criteria are proposed. This is mainly achievable if speedy and robust approaches like Singular Value Decomposition (SVD) [23–25] are used. Such approaches exploit the advantages of orthogonal projections to improve discriminative parameter selections and Allens PREdiction Sum of Squares (PRESS) formula [26] which is an efficient direct way for computing the Leave-One-Out Cross Validation (LOO-CV) error [27]. Instead of using LOO training error as a selection criterion of parameters, PRESS formula can provide a reliable estimation of the ELM generalization performance. It has been employed in many ELM algorithms [28–33]. However, even though PRESS formula greatly speeds up the LOO-CV process, the time required for evaluating a sufficient number of ELM

parameters and selecting the best one is still quite long, threatening the noteworthy advantages of ELM.

In this work, finding the optimum topology of the ELM and the L_2-regularized ELM (RELM) is firstly investigated [34]. Since it utilizes PRESS and SVD techniques, the optimal network parameters can be quickly selected without intervention and the efficiency can be greatly improved as well.

ELM activation functions and some Subspace Learning (SL) approaches have almost nonlinear characteristics that enhance features extraction performance while destroy the local consistency properties. For this reason, the graph Laplacian can be used to regulate the samples in the same class to similar outputs in order to enhance the overall model accuracy [35].

Regularization [36] is a technique that used to prevent the model weight values from becoming large through penalizing them. The idea of penalization is to add the model weight values to the computation of the error term. If weight values are included in the total error term that is being minimized, then smaller weight values will generate smaller error values. Regularization tends to decrease prediction error, reduce the overfitting and diverge from the inverse of singular matrix. Many types of regularization can be employed within the ELM optimization function, among these types are, the L_1 (Lasso regression), L_2 (Ridge regression), and the proposed L_{21} methods. In this work, ELM with proved convergence (Graph L_{21}-norm regularization algorithm) is also introduced and developed as a second proposed technique [37] in order to solve the resultant optimization problem and yield compact hidden layer that clearly enhances the performance. In addition to above improvements, ELM with L_{21} can be considered as an alternative tool with extreme speed to the Sparse Representation based Classification (SRC) approaches that can handle the noisy input data properly.

Instead of using only one model, working with an ensemble of such models can combine their outputs and consequently reduces the risk of selecting of an inefficient performing or poorly classifier. Dietterich [38] reported three primary statistical, computational, and representational reasons for using an ensemble based system. The statistical reason is simply associated to how representing or distributing huge data or inadequate data properly; the computational reason is related to the problem of inappropriate model selection (i.e. which one should be chosen among many models that can solve a given

problem). Finally, the representational reason is how to treat the cases when the chosen model cannot properly handle the required decision boundary.

The ensemble principle presented with ELM is an effective approach to reduce the randomness in hidden layer parameters and to improve the networks generalization ability [39]. Both ensemble and network with regularization can be performed with ELM and it's variants not solely to alter the network topology but to provide another solution that can tackle efficiently the design problems of ELM architecture with the parameters uncertainty during the training process. The improved RELM with the ensemble is our third proposed tool for enhancing the overall generalization.

Among massive types of optimization techniques, convex optimization techniques can be employed to process and solve many serious difficulties in ELM with soft mathematical manner such as producing closed-form solutions. Additionally, these techniques are much related, simple and have speed advantages with different ELM approaches.

Almost all researchers have focused on the face recognition and their data sets because major difficulties which may come across the face and its expression recognition by computers are primarily due to the varying imaging conditions (lighting, direction and viewpoint changes induced by body movement) along with other effects such as facial expressions, aging, occlusions etc.

The researchers of image analysis and processing, computer vision, machine learning, pattern recognition and other areas are working jointly, motivated greatly by a number of possible practical applications.

From learning efficiency point of view, we try always to ensure and maintain three vital-demands required for the robust machine learning: high learning accuracy, fast learning speed, and least human invention.

More specifically, the objectives of this thesis can be summarized as follows:

1) Finding the optimum number of hidden layer neurons without using any type of trial and error or time-consuming method. The primary goal of this thesis is to devise an extremely fast and effective algorithm that can find the appropriate network structure of ELM and its variants models.
2) Utilizing implicitly SVD within the ELM approaches to retain the fast ELM learning advantage and to reduce dramatically the complexity of the generalized matrices.

3) Exploiting good computational approaches to perform efficient version of PRESS (Fast PRESS) that can search and evaluate the optimal network parameters for the ELM and its variants, as a result, it can guarantee both the speed advantage of the ELM and the good performance.
4) Different approaches like SVD, L-curve, and Fast PRESS are combined effectively to determine the network structure (optimum hidden nodes and the regularization parameters) rather than the user manually selecting or tuning these parameters.
5) To reduce model overfitting and increase the stability and accuracy of the ELM and its variants, graph ELM with L_{21} regularization is developed and investigated for face and object recognition.
6) In each round of simulation, the ELM solution usually fluctuates due to the randomly generated parameters of the hidden layer. Therefore, we attempt to improve the stability of ELM and ensure good generalization performance in each round of simulation by implementing an ELM ensemble approach. It is based on the largest weighted voting between the generalized samples. It can strongly decrease the randomness effect and handle only the effective samples.
7) In addition to the ELM ensemble method and in spite of Sparse Recognition Classifier (SRC) has a different methodology to ELM, it can exploit different regularizations like graph and L_{21} in order to fortify the classification against the difficult noisy data or when the ELM is weak to predict and classify properly.
8) Attempting to address and decrease network parameters that play a minor role in the network output as they increase the network complexity.

1.4 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 briefly reviews the theoretical preliminaries of the ELM that are designed based on SLFNs to provide the background knowledge for the proposed algorithms. In addition, a literature review that shows the most significant ELM models is presented.

Chapter 3 exhibits the generally proposed frameworks of the Optimized, Projected and Tikhonov Regularized ELMs (OELM, PELM, and TRELM) that are based on different mechanisms like Least Square Regression, SVD, and L-curve. They introduce various algorithms to estimate the optimum hidden nodes and some parameters that are needed during the learning processes.

Chapter 4 illustrates the Graph ELM with L_{21}-regularization (L21GELM) for classification in details, it can jointly exploit the discriminative information, geometric structure of data and effective regularization to enhance the classification capabilities.

Chapter 5 analyzes the investigation of an ELM ensemble that integrates many single RELM networks with weighted majority voting (WV-RELM). It also includes the proposal of an optimized RELM construction that based on both SVD and Fast PRESS approaches.

Chapter 6 concludes this work, discusses the results, and suggests for future works.

CHAPTER 2

ELM THEORY AND PRELIMINARIES

2.1 Introduction

Artificial neural network (ANN) is a popular machine learning technique, which is designed to model and fit the way in which the brain performs a particular task or function of interest. It is in nature highly complex, nonlinear and parallel. To make neural networks capable of solving complex problems, there are two information-processing capabilities [14]. First, needing to massive parallel distributed structure of neural network that makes it vigorous in computation. Second, it has the ability to learn its parameters and therefore generalize. Among all the structure of neural networks, feedforward neural networks have been the most extensively used in many fields because of their powerful nonlinear mapping abilities. They are capable of providing the model for a large class of natural and artificial phenomena, and they outperform the classical parametric techniques. Nevertheless, the learning algorithms for neural networks are usually far slower than desired. Conventionally, all the traditional learning algorithms are suffering from the tiresome parameter tuning process, where all the layers parameters (mostly the weights and biases) need to be tuned. Sometimes it may take hours even days to train the whole network. The gradient descent-based methods are part of the traditional learning methods, which have been widely used in various learning algorithms of feedforward neural networks. They give satisfactory responses once they have been properly trained. However, they could not be used to train the networks with threshold or digital nodes directly as the threshold functions are non-differentiable. Moreover, in some practical applications of neural networks, a fast response to external events within an extremely short time is expected. The gradient-descent based methods are probably not good choices

when frequent retraining is needed, especially for complex and large-scale applications and when higher learning accuracy and generalization performance are required.

One decade ago and for some ANNs like FNN, Baum [40] proved that fixing the connections on one level and simply adjusting the connections on the other level is possible, and no additional benefit is sensible by adjusting the weights on both levels simultaneously through using any algorithm. This implied that the input weights can be fixed and one may only adjust the output weights. Unlike the traditional methods where all the parameters have to be tuned during the learning process, it is proved by Huang [16, 19] that the input weights and hidden layer weights and biases of the single-hidden layer feedforward neural networks SLFNs can be randomly assigned and fixed without tuning. This SLFN was called Extreme Learning Machine ELM because of its simple network architecture and extremely fast learning speed. Many ELM simulation results on artificial and real applications have rigorously proved that ELM can produce good generalization performance, most of these applications are on image object detection [41–43], time series analysis [44–47], fault detection [48, 49], and remote sensing [50].

Recently, ELM and its variants have attracted a great number of researchers and engineers, they have been applied to various types of real applications including palmprint recognition system [51], no-reference image quality assessment [52], bank clients classification [53], sales forecasting in fashion retailing [54], arrhythmia classification in ECG [55], bacterial protein subcellular localization prediction [56], microarray gene expression cancer diagnosis [57], prediction of melting point of organic compounds [58], fault diagnosis on hydraulic tube tester [59], power utility non-technical loss analysis [60], land cover classification [61], terrain modelling [62], identification of plant species [63], and classification of music genres [64], etc.

2.2 Characteristics of the Standard ELM Approach

Extreme Learning Machine (ELM) [16, 19, 20, 5, 65–67] is a Single-hidden layer Feedforward Neural Network (SLFN) with arbitrarily chosen input weights and biases as shown in Figure 2.1. It is based on the clue that indeed the SLFNs with an adequate number of hidden nodes (L) and random assigned input weights and biases with almost any nonlinear activation function can universally approximate any continuous functions on any compact input sets.

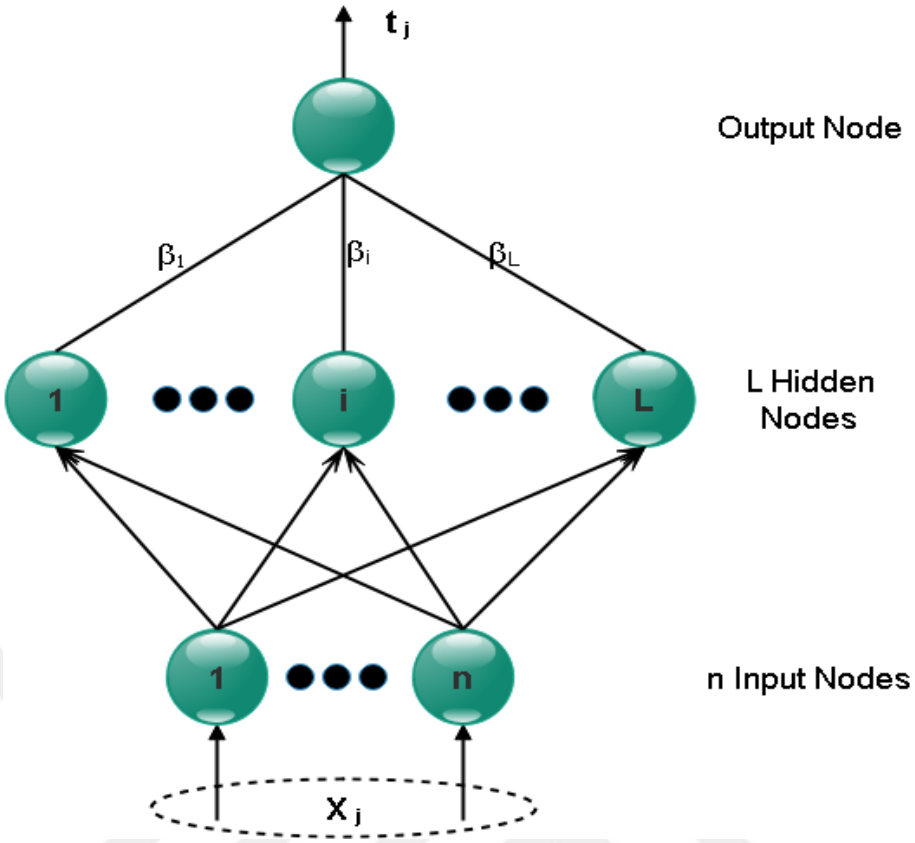


Figure 2.1 SLFN Feedforward Network Architecture of the ELM Model

According to ELM theory [19], any nonlinear piecewise continuous activation function $G(w, b, x)$, can be used for feature mapping to approximate any continuous target function. Some examples of these functions are: sigmoid, tangential, Gaussian, hinging, and ridge polynomials functions.

The hidden layer output function for N input size, n input nodes and L hidden nodes can be written as:

$$h(x) = [G(w_1,\ b_1,\ x_1), \dots, G(w_L,\ b_L,\ x_N)] \tag{2.1}$$

The output function of SLFNs is:

$$t_j = \sum_{i=1}^{L} \beta_i . G(w_i x_j + b_i) \qquad j = 1, \dots, N \tag{2.2}$$

With a sufficient number of L hidden nodes, ELM can be extended to generalize SLFN with a large variety of feature mapping functions on the hidden layer as given in equation 2.3:

$$\|f_L(x) - f(x)\| < \epsilon \tag{2.3}$$

where ϵ is a small error tolerance value.

Consider a data set containing N training samples each with n features $(x_j, t_j) \in R^n \times R^m$, $j = 1, 2, \dots, N$ with n = number of input attributes (features), m = number of output classes (targets), and $G(x) =$ activation function, the estimated output $y_j = \hat{t}_j$ is mathematically modeled as:

$$f_L(x) = y_j = \hat{t}_j = \sum_{i=1}^{L} \beta_i . G\left(w_i x_j + b_i\right) = t_j + \epsilon_j = h(x) . \beta \quad j = 1, \dots, N \quad (2.4)$$

ELM aims to minimize the relative error between $\hat{t}_j$ and t_j, i.e, $min \sum_{j=1}^{L} \left\| \hat{t}_j - t_j \right\|$

where β_i: the output weight vector connecting the i[th] hidden nodes and the output nodes.

w_i: the randomly chosen input weight vector connecting the i[th] hidden nodes and the input nodes or the center of the i[th] hidden node.

b_i: the randomly chosen bias of the i[th] hidden nodes or the threshold or impact factor of the i[th] hidden nodes; x_j: the input attributes; y_j: the actual output.

In matrix form, equation 2.2 is equivalent to $H . \beta = T \rightarrow min_\beta \|H . \beta - T\|$, where $\|.\|$ stands for the norm in Euclidean distance, then $\beta = H^\dagger T$, where $H^\dagger = (H^T H)^{-1} H^T$ is Moore-Penrose generalized pseudo inverse of hidden layer output matrix H [68].

In general, if the hidden nodes number (L) is equal to the size of training samples (N), matrix H is square, invertible and solution can be obtained with error zero. However, for $N \neq L$, the generalized pseudo inverse using Moore-Penrose is adopted to compute the solution with arbitrarily small error. Besides, according to Bartlett's theory [69], to achieve smallest training error for feedforward networks, the norm of the output weight should be small as possible. Therefore, ELMs not only tend to achieve smallest training error but also good generalization performance.

The desired output weights and targets can be expressed with equation 2.5

$$\beta = H^\dagger T = (H^T H)^{-1} H^T T \;\&\; \hat{T} = H\beta = H(H^T H + \lambda I)^{-1} H^T T = HAT . T \quad (2.5)$$

where,

$$H\ (w_1, \cdot\cdot, w_L;\ b_1, \cdot\cdot, b_L;\ x_1, \cdot\cdot, x_N) = \begin{bmatrix} G(w_1, b_1, x_1) & \cdots & G(w_L, b_L, x_1) \\ \vdots & \cdots & \vdots \\ G(w_1, b_1, x_N) & \cdots & G(w_L, b_L, x_N) \end{bmatrix}_{N \times L} \quad (2.6)$$

where $\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L\times m}$ & $T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{N\times m}$, $T_{ij} = \begin{cases} 1 \ for\ vectors\ of\ class_i = j \\ 0 \ for\ vectors\ of\ class_i \neq j \end{cases}$

The i[th] column of T is the output target of the i[th] hidden nodes with respect to inputs $x_1, \cdots, x_N$.

The output functions (activation functions) of the hidden layer mapping that results in ELM feature space can be expressed as in Table 2.1:

Table 2.1 Most common ELM activation functions

Commonly used activation (mapping) functions in ELM, not limited to :	
Sigmoid function	$G(w,b,x) = \dfrac{1}{1+\exp(-(w.x+b))}$
Hyperbolic tangent function	$G(w,b,x) = \frac{1-\exp(-(w.x+b))}{1+\exp(-(w.x+b))} = \tanh(-(w.x+b))$
Gaussian function (RBF)	$G(w,b,x) = \exp(-b\|x-w\|)$
Multiquadric function	$G(w,b,x) = (\|x-w\|+b^2)^{1/2}$
Hard limit function	$G(w,b,x) = \begin{cases} 1 & if\ w.x+b \leq 0 \\ 0 & otherwise \end{cases}$
Fourier basis/ Cosine function	$G(w,b,x) = cos(w.x+b)$

The following steps describe the typical ELM approach for both classification and regression:

Algorithm 2.1 Standard ELM

Given a training set containing of N samples each with n features as $\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \cdots, N\}$, activation function $G(x)$, and hidden nodes number L;

1. Assign randomly input weight vectors or centers w_i and hidden nodes bias b_i, $i = 1, \cdots, L$.
2. Using equation 2.6, calculate the hidden layer output matrix H $(w_1, \cdots, w_L, b_1, \cdots, b_L, x_1, \cdots, x_N)$.
3. Calculate the output weight β as: $\beta = H^\dagger T$, where $H^\dagger = (H^T H)^{-1} H^T$ is the Moore-Penrose generalized inverse of hidden layer output matrix H.

According to the perspective of evaluation the samples are divided into the training and the testing sets. The training sets are used firstly for getting the value of the output weight (β), then

4. Employing the pre-calculated β in step 3 to approximate or classify the test patterns using the following equation $y_{test} = h(x).\beta = H_{test}\beta$ for regression, then apply $L_{test} = arg_{max}^{row}(y_{test})$ for classification, the arg function returns the index of the maximum value for each row of y_{test}.
   where, $L_{test}(m \times 1)$ is the output label of m testing instances.

2.3 ELM Theory and Feature Mapping

The basic ELM algorithm is established on the following two principles [19]:

- When the size of training samples equals the number of hidden nodes, i.e., $N = L$, it can randomly assign hidden layer parameters (the input weights and biases) and the output weights β is analytically calculated by simply inverting H to realize zero training error. Consequently, the output weights calculation is done in a single step here.
- When the size of training samples is not equal to the number of hidden nodes, and in most cases $N > L$, one can still randomly assign the parameters of hidden nodes and calculate the output weights β by using a pseudoinverse of H to give a small nonzero training error $\epsilon > 0$. Here also the calculation of β is done in a single step and does not need extensive training procedure.

The two concepts above can be formed in the following two theorems [19]:

Theorem 1. Given a standard SLFN with L hidden nodes and $G(.)$ activation function which is infinitely differentiable in any interval, for N arbitrary distinct samples(x_i, t_i) where $x_i \in R^n$, $t_i \in R^m$, $i = 1, \cdots, N$ and for any $w_i \in R^n$ and $b_i \in R, i = 1, \cdots, L$ randomly chosen, the hidden layer output matrix H in equation 2.6 of the SLFN is invertible and $\|H\beta - T\|_2 = 0$.

Theorem 2. Given any small positive error value $\epsilon > 0$ and infinitely differentiable activation function $G(.)$, there exists $N > L$ such that for N arbitrary distinct samples(x_i, t_i) where $x_i \in R^n$, $t_i \in R^m$, $i = 1, \cdots, N$ and for any $w_i \in R^n$ and $b_i \in R$, $i = 1, \cdots, L$ randomly chosen, according to any continuous probability distribution and with probability one, $\|H_{N\times L}\beta_{L\times m} - T_{N\times m}\|_2 < \epsilon$.

According to theorems 1 and 2, the output weights can be derived by finding the least-square solutions to the above linear system which is given by $\hat{\beta} = H^{\dagger}T$, where $H^{\dagger}$ is the Moore-Penrose that represents the generalized inverse of hidden layer output matrix *H*.

The ELM algorithm has the following essential properties making it an appealing solution:

1. **Minimum training error:** The solution $\hat{\beta} = H^{\dagger}T$ is one of the least-squares solutions of a general linear system $H.\beta = T$, hence the minimum training error can be reached by this solution:

$$\left\|H\hat{\beta} - T\right\|_2 = \left\|HH^{\dagger}T - T\right\|_2 = min_{\beta}\left\|H\hat{\beta} - T\right\|_2 \tag{2.6}$$

Although almost all learning approaches demand reaching the global minimum training error, however, most of them cannot reach it because of local minima and training with infinite iterations is not allowable in applications.

2. **Smallest norm of weights:** Additional, the specific solution $\hat{\beta} = H^{\dagger}T$ has the smallest norm among all the least-squares solutions of $H.\beta = T$;

$$\left\|\hat{\beta}\right\|_2 \leq \left\|H^{\dagger}T\right\|_2 \leq \|\beta\|_2 \tag{2.7}$$

3. **Unique solution:** The smallest norm least-squares solution of $H.\beta = T$ is unique, which is

$$\hat{\beta} = H^{\dagger}T \tag{2.8}$$

2.4 ELM Models

The ELM models can be divided broadly into two categories: the constructive methods (or growing) such as Incremental ELM (I-ELM) [20] and error minimized ELM (EM-ELM) [70]; and the destructive methods (or pruning) such as pruned ELM (P-ELM) [71] and optimally-pruned ELM (OP-ELM) [72]. Incremental ELM (I-ELM) starts with very small L hidden nodes number and terminates the growth of L when the training error becomes less than the expected error e or L exceeds the predefined maximum L_{max}. However, the output weights of I-ELM has to be predetermined every time a new hidden neuron is added. Correspondingly, Convex Incremental ELM (CI-ELM) [73] uses the same I-ELM termination criteria but output weights are determined using convex optimization learning scheme [74]. The parameters of the hidden nodes of ELM are randomly selected, consequently these parameters may affect the network slightly and

make it more complex. Enhanced Incremental ELM (EI-ELM) [67] has been proposed to overcome this issue by including some hidden nodes over each learning step as well as adding the node that leads to the largest reduction in the residual error to the network. Error-Minimized ELM (EM-ELM) [70, 75] has been reported to increase the learning procedure speed. It can add hidden neurons randomly one by one or chunk by chunk and update the output weights incrementally, thereby the produced network will be more compact by avoiding of insignificant neurons. Due to the shortcomings of the constructive methods regarding their tendency to accumulate a large number of hidden neurons when numerous iterative steps are included, Dynamic ELM (D-ELM) [76] has been proposed. In this method, the hidden neurons are added and dynamically removed based on their importance to the network performance. Bidirectional ELM (B-ELM) [77] has been introduced for further increase in the speed of the hidden neurons selection process. It has higher rapidity compared to other incremental ELM algorithms. This is achieved by creating a number of temporary networks with various L number at each step and then the "best" network (has smallest L among those having the best generalization performance) will be selected. In essence, B-ELM algorithm tends to reduce network output error to 0 at an extremely early learning stage. As referred to [19], ELM can achieve zero training error with max probability, as long as L is large enough. Deng, J., et al. [78] have proposed a two-stage ELM algorithm that integrates ELM and leave-one-out (LOO) cross-validation with a stepwise procedure. This integration is able to automatically estimate the size of the network and also improve the compactness of the ELM model. In the Adaptive ELM (A-ELM) algorithm [79], the size of the hidden layer may decrease, increase or stay constant during the training process. Du, D., et al. [80] achieves high compactness through extending the two-stage ELM to a locally regularized framework. Wang, N., et al. [81] adopted recursive orthogonal least squares to conduct forward selection and backward elimination of hidden neurons in ELM. Luo, J., et al. [82] introduced a sparse Bayesian approach for learning a compact ELM model. The constructive algorithms mentioned above share one serious issue that they all follow the I-ELM concept and use the same architecture selection criteria which have an expected learning accuracy or error (e) and a maximum number of hidden nodes ($Lmax$). However these criteria are hard to specify (i.e, enough e and large enough $Lmax$) and may experience seriously overfitting issue and hence affect the performance.

In other words, these incremental ELM algorithms solely transfer the selection of L into e and $Lmax$, which is not a reliable approach to ease the burden of tuning efforts. In some applications, the training data are received sequentially, thereby, Liang, N.Y., et al. [83] proposed the online sequential ELM (OS-ELM) approach to extend ELM for online sequential data. This approach can learn the data one-by-one or chunk-by-chunk with fixed or differing chunk sizes. On the other hand, the training data in some applications are of large scale and have imbalanced distribution (i.e., the weights of training samples in some classes are much larger than those of other classes or the data points are associated with a unique cost or an importance coefficient). Therefore, to overcome this issue and tackle the problem of training imbalanced data, the weighted ELM (W-ELM) algorithm [84] has been suggested. This approach adds different penalty coefficients to the training errors corresponding to different inputs. The equality constraints in traditional ELM model can be replaced by inequality constraints which lead to quadratic programming (QP) and sparse problem. Bai, Z., et al. [85] succeeded in reducing the storage space and testing time significantly by proposing a sparse ELM (S-ELM). Zhang, K., et al. [86] introduced a robust ELM that depends on the AL (Augmented Lagrange) optimization concept but has been used for the regression only. Akusok, A., et al. [87] introduced a complete approach which manifested a successful utilization of a high-performance extreme learning machines (ELMs) Toolbox for large data handling. Later, Chen, K., et al. [88] developed a Regularized ELM (RELM) approach which can iteratively reweight the assigned weights using the least squares method. It adapted many robust regularized schemes and used for regression only. It is stable and accurate until 40% outlier levels.

Higher precise converges as well as less sensitivity to local minima for the network is usually achieved with a pruning algorithm. However, pruning algorithms have some disadvantages including the difficulty in determining the starting network structure to initiate pruning and the problem of the computational complexity at the beginning since the network starts from a larger size than needed.

2.5 ELM Optimizing Approaches

For optimizing the ELM structure, three main approaches can be identified: constructive approaches; pruning approaches; and regularization approaches, as well as combinations of them.

2.5.1 Incremental Approaches

Kwok and Yeung succeeded in analyzing the SLFN constructive algorithms [89] comprehensively. They built a simple tactic for the SLFN construction to find the network characteristics such as the size. Their tactic commences with an empty SLFN and then the hidden nodes are added singly (i.e. one by one). As illustrated in Figure 2.2, the SLFN parameters can be separated broadly into four parts: the hidden parameters of the prior network (A), the output weights of the prior SLFN (B), the hidden parameters of the new added neuron (C) and the output weight corresponding to the new added neuron (D). Assume the current SLFN has *K* hidden neurons, one is attempting to add a single hidden neuron, i.e., the (*K+1*)th hidden neuron. Whenever a new node is added, some parameters tuning is processed. According to the Kwok's algorithm, every time a new hidden neuron is added, the previous input parameters (A) are fixed. Several objective functions have been proposed for the optimization of the input parameters of the new neuron (C). Then, with the optimized parameters (C), all the output weights (B, D) are calculated via pseudo inverse.

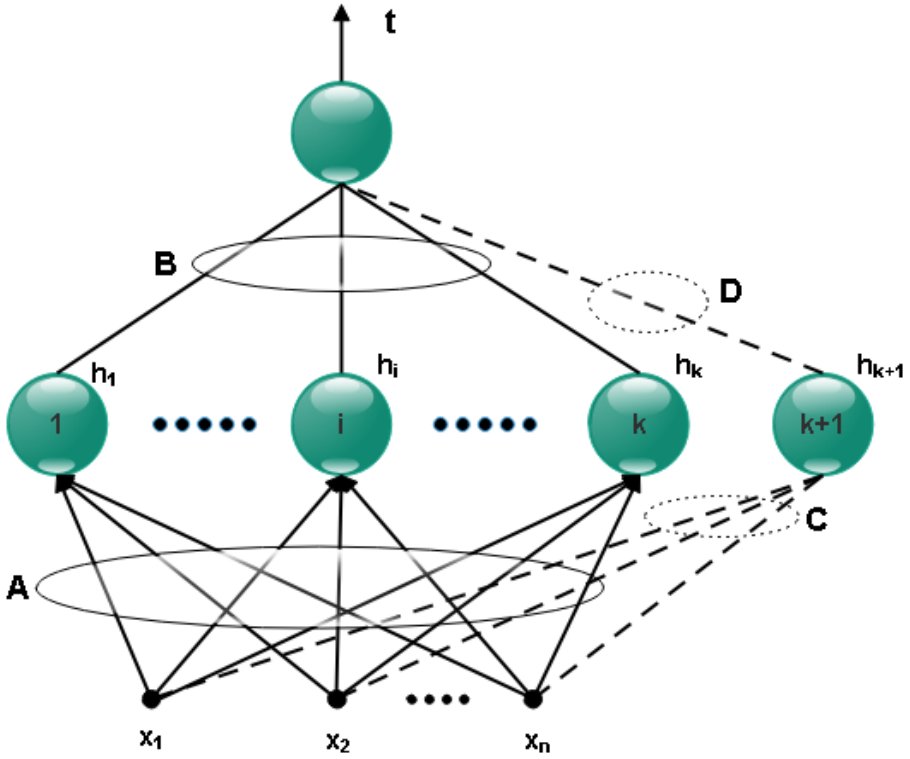


Figure 2.2 SLFN construction based on Kwok's theory

Starting from a SLFN without hidden neurons, $K = 0$, the Kwok's algorithms is recapped as follows:

1. Arbitrarily assigned hidden neurons are added, which have outputs h_{K+1}.

2. The input parameters of the neuron (C) are optimized using some algorithms which are applied on selected objective functions.

3. With the new input parameters, recalculate the outputs of the new neuron h_{K+1}.

The new as well as previous network are consolidated, and then the optimal output weights are computed via the Least Square (LS) method as follows:

$$\hat{\beta} = H_{K+1}^{\dagger} t = (H_{K+1}^{T} H_{K+1})^{-1} H_{K+1}^{T} t \tag{2.9}$$

4. $K = K + 1$. whenever the obtained Sum Squared Error (SSE) of the new SLFN is acceptable, the training will be stopped; otherwise, the training will go back to step 1 for adding of another neuron.

Incremental ELM methods such as I-ELM, EI-ELM, CI-ELM and EM-ELM [20, 70, 73, 67] have proved that increasing the number of hidden nodes is probably decreasing the residual error of SLFNs and restricts it close to zero. Unlike classical ELM with a fixed network architecture, Incremental ELMs start from a small network and establish the network structure via adding hidden nodes one by one or chunk by chunk until the expecting training accuracy is gained. Nevertheless, all experimental results have manifested that the incremental ELM methods have learning time many folds longer than that of standard ELM. This is due to the one by one calculations of the n output weights in incremental ELMs compared to the standard ELM which computes the n output weights solely once, i.e, when *L* hidden nodes are adopted.

2.5.1.1 Incremental ELM (I-ELM)

In the Incremental ELM (I-ELM) algorithm [20], nearly all parameters are created once and then fixed as shown in Figure 2.2. Every time when the new neuron is added, the parameters (A) and (B) are held constant as previous values whereas parameters (C) are created randomly and then fixed. Only the parameter (D) is determined according to the following formula,

$$\beta_{K+1} = \frac{e_K^T h_{K+1}}{h_{K+1}^T h_{K+1}} \tag{2.10}$$

The learning procedure of the I-ELM algorithm can be summarized as following, starting from an empty SLFN without hidden neurons, $K = 0$, $e_K = t$.

1. Add a random hidden neuron, whose outputs are h_{K+1}.

2. Calculate output weight of this new neuron with equation 2.10.

3. The error is then updated based on $e_{K+1} = e_K - \beta_{K+1} h_{K+1}$.

4. $K = K + 1$. Based on the SSE of the new SLFN, If it is acceptable, the training will be stopped; otherwise, the training will go back to step 1 for the addition of another neuron.

The convergence of the I-ELM has been proven mathematically [20]. However, most of its parameters are randomly generated as they are not tuned once. I-ELM usually results in a SLFN algorithm much larger than required to reach a desirable or acceptable error level.

2.5.1.2 Enhanced Incremental ELM (EI-ELM)

The Enhanced Incremental ELM (EI-ELM) [70] can improve the I-ELM by selecting the input parameters (C) of each new neuron from a random candidate pool. The technique is shown below,

Set a pool size N_c, starting from an empty SLFN without hidden neurons, $K = 0$, $e_K = t$.

1. Firstly, create a candidate pool randomly for input parameters of the new neuron which has size of N_c. The outputs of each candidate will be $\left\{h_{K+1}^{(1)}, h_{K+1}^{(2)}, \dots, h_{K+1}^{(N_c)}\right\}$.

2. For each candidate in the pool, calculate its corresponding optimal output weight with equation 2.10, then compute the SSE if adding this neuron.

3. The candidate that leads to minimal SSE is selected and added to the SLFN.
4. $K = K + 1$. Based on the SSE of the new SLFN, If it is acceptable, the training will be stopped; otherwise, the training will go back to step 1 for adding of another neuron.

The EI-ELM rises the computation time to increase the network's compactness.

2.5.1.3 Convex Incremental ELM (CI-ELM)

Because of its capability for tuning more parameters, the Convex Incremental ELM (CI-ELM) [73] is able to improve the I-ELM depends on the convex optimization method. According to the SLFN structure (Figure 2.2), CI-ELM determines both the optimum parameters (D) as well as rescales the previous output weights (B) instead of determining

only parameters (D) as the I-ELM. The output weight of the new added neuron is calculated using equation 2.11,

$$\beta_{K+1} = \frac{e_K^T(e_K-(t-\beta_k h_{K+1}))}{(e_K-(t-\beta_k h_{K+1}))^T(e_K-(t-\beta_k h_{K+1}))} \quad (2.11)$$

All the previous output weights are rescaled as,

$$\beta_k = (1-\beta_{K+1})\beta_k, \;\; k = 1, 2, \dots, K \quad (2.12)$$

The CI-ELM training procedure is illustrated below,

Commencing with an empty SLFN without hidden neurons, $K = 0$, $e_K = t$.

1. A random hidden neuron which has outputs (h_{K+1}) is added.

2. Determine the optimal output weight of the new neuron according to equation 2.11. After that, rescale output weights of other hidden neurons as equation 2.12.

3. $K = K + 1$. If the SSE of the new SLFN is acceptable, stop training; otherwise, the training will go back to step 1 for adding of another neuron.

2.5.1.4 Error Minimized ELM (EM-ELM)

Error Minimized ELM (EM-ELM) [70, 75] has been proposed as an efficient constructive version of the original ELM and I-ELM. In this algorithm, each time the random new hidden neurons are added, the output weights (B and D) are held to be least square optimum. However, instead of using the pseudo inverse repeatedly, the EM-ELM provides a recursive way to update those output weights, which works more efficient than the original ELM with trial and error. The EM-ELM algorithm has two versions: either adding hidden neurons one by one; or adding hidden neurons batch by batch. Here, the first version is only presented (i.e. adding hidden neurons one by one) due to the difficulty in awareness of how big the batch must be and the easiness in making the network size large. In the EM-ELM algorithm, two intermediate matrices (D_K, U_K) are defined and given in equations 2.13 and 2.14,

$$D_K = \frac{h_{K+1}^T(I-H_K H_K^\dagger)}{h_{K+1}^T(I-H_K H_K^\dagger)h_{K+1}} \quad (2.13)$$

$$U_K = H_K^\dagger(I - h_{K+1}D_K) \quad (2.14)$$

Then all the output weights (including previous neurons and new neuron) are calculated as:

$$\beta_{K+1} = H_{K+1}^{\dagger} t = \begin{bmatrix} U_K \\ D_K \end{bmatrix} t \tag{2.15}$$

Similar to the ELM, the EM-ELM always keeps the output weights to be least square optimum. Consequently, more compact SLFN in comparison to I-ELM, EI-ELM and CI-ELM can be obtained whilst gaining very close accuracy. Moreover, EM-ELM has high efficiency than the direct ELM since it works in a repetitive way to update the optimum output weights. Additionally, the EM-ELM algorithm possesses closed-form update rules for the output weights when new neurons are added, resulting in a faster growing step.

2.5.2 Pruning Approaches

The pruning approaches, in contrast to the incremental approaches, produce a larger group of neurons than necessary. Then, they pick the preferable subset of candidate neurons based on particular criteria (e.g. Chi squared test χ^2 criterion and the information-gain criterion) for being used in the SLFN. In the Pruned ELM [71] approach, a large set of candidate neurons is engendered and ranked based on the outcomes of statistical tests. Use a separate validation set, the optimum threshold for this criterion is determined using the Akaike Information Criterion (AIC) [90], later, the retraining of the network is performed on the whole training set. On the other hand, the Optimum Pruned ELM (OP-ELM) [72] approach takes advantages of the linearity of the ELM approach in the output of the hidden layer. Indeed, this allows a swift and optimal ranking (regarding of training error) of the candidate neurons via Least Angle Regression (LARS) [91] or Multi Response Sparse Regression (MRSR) [92]. Once the candidate neurons have been ranked, the optimum hidden neurons are calculated employing the leave-one-out error. The latter can be efficiently computed utilizing PRESS statistics [26]. The OP-ELM approach can be applied to both regression and classification problems. In this approach, the appropriate number of neurons is chosen according to the Leave-One-Out criterion.

Both the pruning and incremental approaches are quite similar apart from the term ’pruned’ which suggests that the network structure is started with a large network and neurons which are then removed one by one. Moreover, the major difference between them is in their initializations, where rather than engendering new neurons over every step in I-ELM, the whole list of candidate neurons is created and ranked as the first step in OP-ELM, and then the neurons to be added are taken from that ranked candidate list of neurons. Wang, N. et al. [93] succeeded in modifying the OP-ELM approach to one

which is more efficient called Constructive Multi-output ELM. In their proposed approach, many regressors of an MRSR-like algorithm can be added in each step instead of a single one.

2.5.3 Regularization Approaches

Another alternative method for selecting the subset of hidden neurons is regularizing the network. In this method, it is potential to generate a large set of hidden neurons, and stop overfitting by a proper regularization method based on the nature of data applied and their resultant set of equations.

The Regularized ELM (R-ELM) [5, 94] for example, is a regularization approach in which the set of candidate neurons is fixed and taken large enough, and to prevent overfitting, L_2-norm is often used.

The ELM algorithm may have some problems when encountering correlated or irrelevant data. Therefore, L_1-regularization is usually applied on the hidden layer neurons, and most likely pruning some of them out. The resulted L_1-regularized ELM approach is widely recognized as an Optimally-Pruned ELM (OP-ELM) [28]. Consequently, and in addition to the pros of sparsity, the output weights stay small and the overfitting is prevented.

Finally, incorporating the optimization of an L_2-regularization parameter in the OP-ELM to yield the Tikhonov regularized OP-ELM (TROP-ELM) [95]. It is an efficiently variant of the OP-ELM, in which it can utilize and mix the SVD approach to compute pseudo-inverse $H^{\dagger}$. The L_2-regularization is also simple and can be easily combined with L_1-regularization to handle a wide variety of data types for achieving the best outcomes [96, 36].

CHAPTER 3

OPTIMUM, PROJECTED, AND L2-TIKHONOV REGULARIZED EXTREME LEARNING MACHINE METHODS

3.1 Introduction

Guang and his group have reported that a single hidden layer feedforward neural network (SLFN) which has sufficient number of hidden neurons, arbitrarily assigned input weights and biases with almost any nonlinear activation function can universally approximate any continuous functions or any compact input sets with zero or randomly small error [16, 19]. It possesses extra merits including least human invention, high learning efficiency, and fast learning speed [97–99].

Unlike the popular thinking of most practical ANN implementations that need to tune all the parameters of the feedforward networks, one may not necessarily adjust the input weights and biases in ELM approaches and the output weights can be simply calculated by least square optimization. It is clear from the previous chapter that ELM with sufficient L number of hidden neurons can learn L distinct observations, accordingly the performance of the ELM is mainly influenced by the number of hidden nodes (L). Many algorithms related to ELM are employed to find L, most of them are depending on error criteria, however they need large consuming time due to its consecutive nature of process and absolutely ELM will lose its salient feature that is the fast advantage. L_1 and L_2 regularization techniques are two closely related techniques that can be used by training the ELM machine learning algorithms to reduce model overfitting and increase accuracy, i.e, lead to a model with better predictions. The L_1 (Lasso regression) weight regularization penalizes weight values by adding the sum of their absolute values to the error term while L_2 (Ridge regression) weight regularization penalizes weight values by adding the sum of their squared values to the error term. The Regularized ELM (RELM)

is a well-known variation of ELM, it has two vital parameters that effect the performance which are the hidden nodes number (L) and the regularization parameter (λ).

In this chapter, we propose three enhancements to ELM. They can identify the vital parameters of the ELM like the number of hidden neurons L, regularization parameter λ, and the input weights w_i and biases b_i, it can highlight them as:

The first investigation is yielding an efficient and very fast approach that can find the optimum L for the ordinary ELM. The second contribution is to develop an approach based on L-Curve method suitable to extract the optimum λ for the RELM. The third study is exploiting the SVD factorization in determining the best input weights w_i and biases b_i. The Last achieved aim is focused on finding a numerical algorithm based on mixing SVD and PRESS approaches for optimally select both L or/and λ simultaneously by same method. These approaches have been proved that they can keep the fast and good generalization of the ELM and its variants.

3.2 ELM Structure Preliminary

ELM is a successful algorithm for both classification and regression [16, 19, 99, 100], also extending the ability to include clustering [101]. It can be used efficiently with online sequential [83] as well as self-adaptive evolutionary algorithms [102], and in handling big data [87, 103]. Almost all ELM researches have revealed that choosing the number of neurons in the hidden layer (L) is a key factor in determining the overall SLFN network architecture and performance, and consequently ELM is very sensitive to the initial number of hidden neurons. In ELMs with a fixed network architecture, the only factor that needs to be specified is the size of the SLFNs (the hidden nodes number). Selection of the optimum number of the hidden nodes is still unknown, it is set by the user and usually adjusted by trial and error [77, 104, 105] or time consuming approaches like LOO-CV [27, 71, 72]. For a certain data set, Figure 3.1 illustrates that the best average testing root mean square error (RMSE) is around 0.0934 when 20, 30, or 40 hidden nodes were adopted, once the number of hidden nodes was raised up to 50, the testing RMSE increased to 0.0945. This is due to non-monotonic sequence in the residual error of SLFNs when the number of hidden nodes increases.

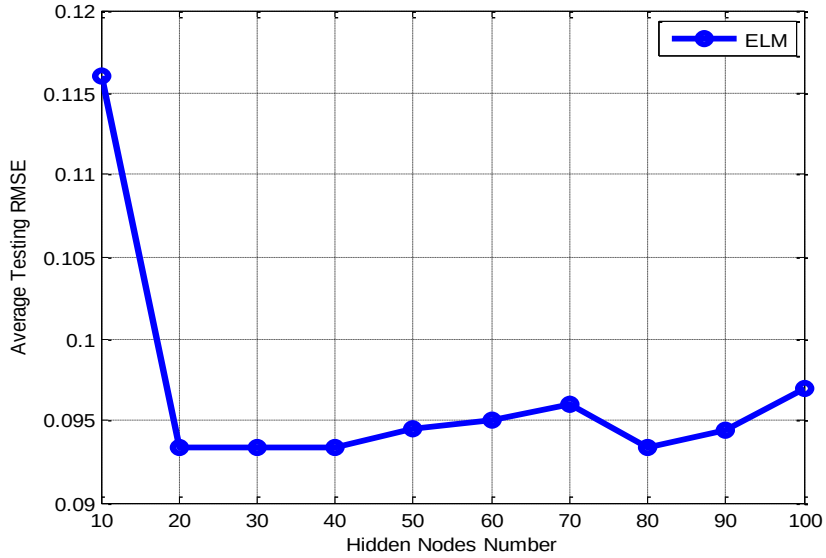


Figure 3.1 The relationship between average testing RMSE and the hidden nodes number for a certain data [77]

To tackle this issue, researchers have proposed many ELM models that can properly select the network topology according to certain criteria. These models can be divided mainly into two categories: the constructive (growing) methods as Incremental ELM (I-ELM) [20, 47] and Error Minimized ELM (EM-ELM) [75]; and the destructive (pruning) methods like Pruned ELM (P-ELM) [71] and Optimally-Pruned ELM (OP-ELM) [72]. All the model structure selection techniques aforementioned here have serious shortcomings, which are, they initiate a starting network structure, then gradually adjust the network depending on the error metric. They usually converge progressively with the lack of how to determine the starting network topology. Also, they significantly increase runtime, which is a limiting factor for training large ELM models and they are of less important in large scale data tasks.

The main objective of this chapter is to develop fast and robust algorithms that can find the appropriate network structure for ELM and its variants without iterative solutions.

3.3 Standard ELM with Optimum Hidden Nodes (OELM)

Although numerous developments in ELM algorithm have been performed, classical ELM still has many advantages over most other ELM developed methods. They include: higher speed (ten times faster than the modified and developed ELM); simplicity of theory, convenience of implementation, and ease of parameters modification to make them more applicable with any regularization or matrix factorization; flexible

applicability to a wide range of feature mapping functions and kernels; and finally, easiness in developing hardware applications with the standard ELM [50, 65, 100].

According to investigations on various feedforward networks neural network studies, the number of neurons in the hidden layer (L) has a worthy relationship with the following parameters:

- The number of input (n) (features) and output (m) (targets/classes) nodes.
- The amount or the size and complexity of training data available (N).
- The training algorithm and the complexity of the learning function (the selection is based on cross-validation methods).
- The extent of accuracy and hence generalized error (ε).

The proposed solution in determining the first minimum optimum number of hidden nodes (L_{opt}) is by examining statistical-based approach (simple least square regression) combining with previous factors. L_{opt} is suitable for classical ELM learning method and it can be estimated as below:

$$L_{opt} = \begin{cases} \propto.(2n+m) & \text{for classification} \\ \propto.(2n+1) & \text{for regression} \end{cases} \quad (3.1)$$

$$\text{Here, } \propto = \begin{cases} 2^{\varepsilon} & all\ regression\ coefficients \neq 0 \\ 2^{\varepsilon+1} & otherwise \end{cases} \quad (3.2)$$

where ε: is the root mean square error that influences the dependent variable t_i.

To calculate the error disturbance, the model can take the form:

$$t_i = x_i^T \gamma + \varepsilon_i \,,\ i = 1, \cdots, N \quad (3.3)$$

The linear system can be written as:

$$X\hat{\gamma} = T \Rightarrow X^T X\hat{\gamma} = X^T T \quad (3.4)$$

Solving the above least square problem with

$$\hat{\gamma} = (X^T X)^{-1} X^T . T = X^{\dagger} T \quad (3.5)$$

where $X^{\dagger}$ is known as the pseudo-inverse or Moore-Penrose inverse [1]

The predicted response is $y_i = \hat{t}_i = \mathrm{X}.\hat{\gamma} = X(X^T X)^{-1} X^T . T = HAT.T$ (3.6)

where the HAT-matrix is the matrix that transforms the output T into the approximated output $\hat{T}$ as in equation 3.6.

[1] The matrix $X^T X$ is invertible (non-singular) exactly when its rank equals dimension *d*, which is usually the case if $N \geq d$. In case $N < d$, $X^{\dagger} = X^T (XX^T)^{-1}$.

The residual mean errors are: $\varepsilon_i = (t_i - y_i)^2$ and hence RMSE is $\varepsilon = \sqrt{\sum_{i=1}^{N} \varepsilon_i}$ (3.7)
$\propto$ is depending on data size and modeling complexity type, i.e, $\propto = 2^{\varepsilon}$ and this value is extended to $2^{\varepsilon+1}$ when one or more regression coefficients are zero, i.e, γ_i=0 which means absence of predictors from the model. This will increase the complexity of handling data and hence increasing the hidden nodes is necessary.

Although the input data distribution may not be linear, it is essential step to calculate (ε) in formulating the linear regression model as in equation 3.5 to simulate the least squares tactic that used to solve the output weights $\beta = H^{\dagger}T$ of the ELM.

3.4 Projection ELM (PELM)

ELM models like other machine learning approaches tend to have problems when irrelevant or correlated variables are present in the training data set, i.e, if the data features are strongly correlated and not independent and/or when the number of data samples is not close to that of hidden neurons. As a result, ELM may not converge and HH^T or H^TH matrix is nearly singular and therefore its inverse is numerically unstable.

SVD [2] matrix factorization principle can be employed in producing orthogonal subspaces and eigenvalue base matrices to solve the problem of correlated input features and to extract useful features with a dimensionality reduction [23, 24]. To suit non-square matrices, ny matrix may be decomposed into a set of characteristics eigenvector pairs as: $A^{m\times n} = UDV^T$, where $U^{m\times m}$, $V^{n\times n}$are orthogonal matrices and $D^{m\times n}$ is a diagonal matrix with singular values of A arranged in descending order of magnitude.

Since ELM structure is only one hidden layer, the projection is applied on the input layer between weight vector (w) and input attributes (X) [105, 106]. Instead of choosing random weight vectors for w, the normalized constrained hidden weights was selected based on input sample distributions [107], however, the only shortcoming in the constrained method is no guarantee to choose the proper constrained vectors distribution of all input sample. Depending on the SVD factorization facilities, two possible schemes [108] can be implemented. The first scheme is a two stage method. The two stage method as shown in Figure 3.2 is a type of the SVD dimensionality reduction, it can extract

[2] Some useful rules of SVD are:

If $\mathrm{SVD(A)} = \mathrm{UDV^T}$, $\mathrm{U} = \mathrm{AA^T}$, $\mathrm{V^T} = \mathrm{A^TA}$, $\mathrm{U^TU} = \mathrm{I}$, $\mathrm{V^TV} = \mathrm{I}$, $\mathrm{D} = \mathrm{D^T} \Rightarrow \mathrm{DD^T} = \mathrm{D^TD} = \mathrm{D^2}$, $\mathrm{U^T} = \mathrm{U^{-1}}$, $\mathrm{V^T} = \mathrm{V^{-1}}$, $\mathrm{SVD\ (AA^T)} = \mathrm{UD^2U^T}$, $\mathrm{SVD\ (A^TA)} = \mathrm{VD^2V^T}$.

discriminative feature but it needs more physical space and increase structure complexity of the model. Another method is the single stage method as shown in Figure 3.3, depending on the projection bases available with singular decomposition, the input weights are obtained and the hidden neurons are ranked by their contribution level and only those that are important are selected.

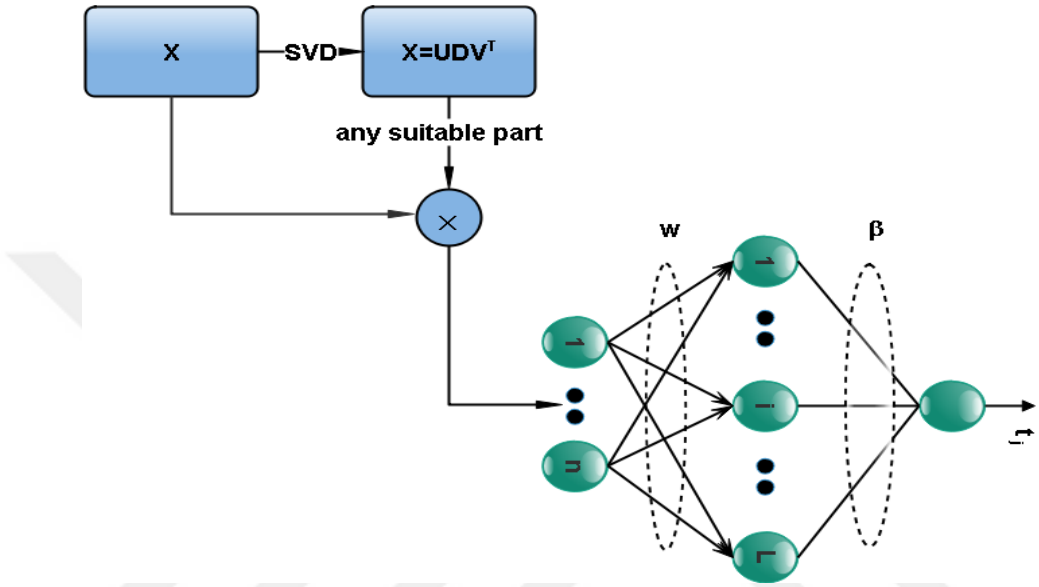


Figure 3.2 Double-stage learning method: dimensionality reduction approach

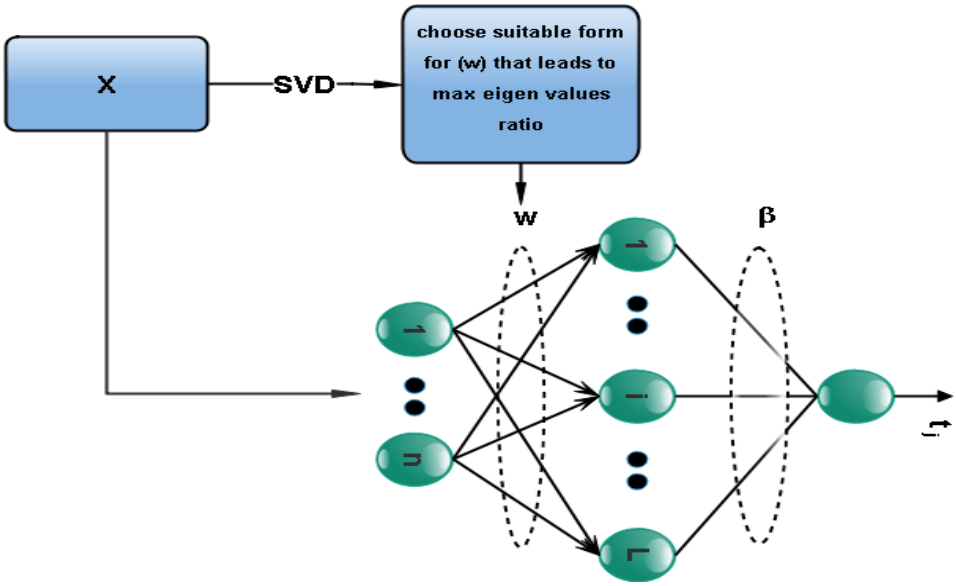


Figure 3.3 Single-stage learning method: base projection approaches

In the proposed method, we depend on different orthogonally subspace available in input data via using SVD to adopt many input weights forms as: $w_1 = V\sqrt{D^T}$, $w_2 = VD^TU$, $w_3 = VD^TU^T$, $w_4 = V\sqrt{D^T}U$, and $w_5 = V\sqrt{D^T}U^T$. Then find the input eigen values with $\sigma =$ eigen values $(w_i.X + b)$ and compute the ratio between the maximum and minimum eigen values as $c = \sqrt{\sigma_{max}/\sigma_{min}}$, the efficient weight vector (w_i) is that corresponding to max c.

Choosing the largest c value leads to large span between maximum and minimum eigen values and ensures generating discriminative feature mapping that can approximate all or almost functions distinctly, it can make $H^{\dagger}$ as square matrix with a minimum row or column dimension, nonsingular, stable and allowable for fast calculations. In other words, small c means the feature distributions are weak and correlated and the chance of generating a compactness mapping that predicates the desired function is low.

For a multiclass classification with m-categories, assuming that $t_i \in \{1, \dots, m\}$ for all samples, then the One-Against-All (OAA) method maps t_i to a m-dimensional vector $t_i = \left(t_1^i, \dots, t_m^i\right)^T$ whose entries are assigned according to the rule: if $t_i = m$, then, $t_m^i = 1$; else $t_m^i = 0/-1$. Thus, for a querying sample, its category is assigned to the index corresponding to the largest entry of network output vector.

The proposed Optimum and Projection ELM algorithm is described hereafter:

Algorithm 3.1. Optimum and Projection ELM

Suppose a training data set containing N size each with n features is given as $\{(x_i, t_i) | x_i \in R^n,\ t_i \in R^m,\ i = 1, \cdots, N\}$, i.e, $X \in R^{N\times n}$ and $T \in R^{N\times m}$ with any suitable activation function.

1. Using X and t_i, find $\hat{\gamma} = (X^TX)^{-1}X^T.t_i$ then calculate the fitted response $y_i =$ X.$\hat{\gamma}$ and the root mean error as $\varepsilon = \sqrt{\sum_{i=1}^{N}(t_i - y_i)^2}$, following by using equation 3.7 to specify $\propto$.
2. Performing equations 3.2 and 3.1 to calculate the best minimum number of hidden neurons L_{opt}.
3. Depend on either randomly assign input weight vectors w_i and hidden nodes bias $b_i, i = 1, \cdots, L_{opt}$, or the Projection principle of SVD as briefed below:

a. Find the singular value decomposition of input features (X) as $SVD(X) = UDV^T$.

b. Based on the SVD factorization, set input weights as: $w_1 = V\sqrt{D^T}$, $w_2 = VD^TU$, $w_3 = VD^TU^T$, $w_4 = V\sqrt{D^T}U$, and $w_5 = V\sqrt{D^T}U^T$.

c. Choose the bias as $b = D(1,1) * rand(L_{opt}, 1)$, or b= $rand(L_{opt}, 1)$.

d. Find $\sigma =$ eigen values $(w_i.X + b)$, $i = 1, \dots, 5$; then compute the ratio between the maximum and minimum eigen values as $c = \sqrt{\sigma_{max}/\sigma_{min}}$ where σ_{max} and σ_{min} are always positive.

e. Choose the w_i-form that corresponding to maximum c and select $w = w_i(:, 1: L_{opt})$.

4. Applying equation 2.6, calculate the hidden layer output matrix H $(w_1, \cdots, w_{L_{opt}}, b_1, \cdots, b_{L_{opt}}, x_1, \cdots, x_N)$.

5. Using equation 2.7, extent the target vector t to T as:

$$T_{ij} = \begin{cases} 1 \ for\ vectors\ of\ class_i = j \\ 0\ /-1\ for\ vectors\ of\ class_i \neq j \end{cases}$$

For vectors of $class_i \neq j$, we choose $T_{ij} = 0$ for calculating the regression (in order to get the real estimated value) and $T_{ij} = -1$ for evaluating the classification (with a view to enlarge the span between different classes).

6. Calculate the hidden output weight β: $\beta = H^{\dagger}T$, where $H^{\dagger} = (H^TH)^{-1}H^T$ is the pseudo inverse of hidden layer output matrix *H*.

   According to the perspective of evaluation, the samples are divided into training and testing sets. The training sets are adopted first to obtain the value of the output weight (β), then

7. The remaining testing set with the extracted β are used to fit or classify the test patterns using the following equation $y_{test} = H_{test}\beta$, or apply $L_{test} = arg_{max}^{row}(y_{test})$, where, $L_{test}(m \times 1)$ is the output label of *n* testing instances and the arg function returns the index of the maximum value for each row of y_{test}.

3.5 Regularized ELM (RELM)

The conventional ELM is essentially like a discriminative least square regression classifier. The only difference between ELM classifier and least square regression classifier is that ELM nonlinearly maps the data from the original space to hidden layer space. Therefore, both of these two models may encounter the singular problem when

calculating the inverse of normal equation. To deal with the singular problem, an L_p-norm is usually incorporated. Empirically, it can observe that the regularized ELM outperforms the standard ELM because it is more stable in computing the inverse of normal equation after adding an L_p-norm regularization. In addition, the regularized ELM algorithms allow small training errors and try to minimize the norm of the network output weights $\|\beta\|$. Several regularization methods have been used in the ELM, among those, the most common one is the L_2-ridge regression , in statistics, Tikhonov Regularization (TR) [109]. RELM allows addition of a small positive value called regularization parameter (λ) to the diagonal $H^T H \; or \; HH^T$ to gain more stability, robust solution, and good generalization performance, thus avoid model overfitting and improve overall prediction accuracy. It acts as a trade-off constant between maximizing marginal distance (minimize the norm of β) and minimizing least square error. It is probably the most successful regularization method of all time [110].

$$Minimize_\beta: f_{ELM} = \frac{1}{2}\|\beta\|_2^2 + \frac{1}{2\lambda}\sum_{i=1}^{N}\|\xi_i\|_2^2$$
$$\text{Subject to: } H\beta = t_i^T - \xi_i^T \;,\; \forall i \text{ , i.e, } i = 1, \dots, N \tag{3.8}$$

Where λ is the regularization parameter.

By substituting $\xi = T - H\beta$ and taking $\frac{\partial f_{ELM}}{\partial \beta} = 0$, this leads to:

$$H^\dagger = (H^T H \; + \; \lambda I)^{-1} H^T \quad \text{if } N \geq L \tag{3.9}$$

$$H^\dagger = H^T (HH^T \; + \; \lambda I)^{-1} \quad \text{if } L > N \tag{3.10}$$

Finally $\beta = H^\dagger T$ and $f(x) = h(x).\beta$ (3.11)

The RELM optimization function has a unique closed form solution as in equations 3.9 and 3.10.

In RELM, $H^\dagger$ is formed depending on both $N \& L$ dimensions. Equation 3.10 can be derived from equation 3.9 according to [111], in which the matrix inversion property is as equation 3.12:

$$(A + BCD)^{-1}BC = A^{-1}B(C^{-1} + DA^{-1}B)^{-1} \tag{3.12}$$

Let, $A = \lambda I, B = H^T, C = I, D = H$

Substitute above assumptions into equation 3.12 to produce equation 3.10 as:

$$H^\dagger = \frac{I}{\lambda}H^T(I + H\frac{I}{\lambda}H^T)^{-1} = H^T(\lambda I + HH^T)^{-1} = H^T(HH^T \; + \; \lambda I)^{-1}$$

RELM has a universal approximation capability [5] which can lead to unified extreme learning machine, thus enabling it to be applicable for different kinds of problems. For Kernel-based and $L \geq N$ [50, 65]:

$$f_L(x) = h(x).H^T(HH^T + \lambda I)^{-1}T = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{bmatrix}^T .(\Omega_{ELM} + \lambda I)^{-1}T$$

Where $\Omega_{ELM} \equiv$ ELM kernel $= h(x_i) . h(x_j) = K(x_i, x_j), i, j = 1, \dots, N$ (3.13)

The time consuming of the ELM and RELM methods is greatly depending on the inversion and multiplication manipulations of the matrices, therefore adopting efficient mathematical approaches like SVD or QR matrix factorization can yielding a significant key to reduce these operations.

As mentioned before, ELM is a very efficient and effective batch learning ANN algorithm that has been implemented in various methods and applications. The salient feature of ELM and its variants is the fast learning speed, which is achieved by non-tuning randomly generating of its hidden layer parameters. However, this randomness nature is also one of the ELM main problems.

The Bias-Variance trade-off of the ELM algorithms that use Least Square Error (LSE) are unbiased, and do not attempt to reduce the variance term. The ridge regression regularization effect is quite beneficial to ELM, because it forces output weights to smaller values with lower variance, and hence leads to a decrease in prediction error.

ELM and RELM share the same set of initial parameters. RELM is much less susceptible to the overfitting issue than the original ELM. With a suitable λ, RELM can achieve universal approximation learning ability on the training dataset if L is large enough (almost, several times those used for ELM) and hence RELM requires more operation time than ELM. The ELM is more sensitive than RELM to the changing of hidden nodes number L. For fair comparison, we were applying both ELM and RELM that share same initial parameters with λ = 0.001 on the breast cancer dataset [112] for classification purpose and λ = 0.1 on the wpbc_dataset [112] for regression purpose. Figures 3.4 and 3.5 show the perfomance (accuracy or RMSE) comparisons of both ELM and RELM for a single trial.

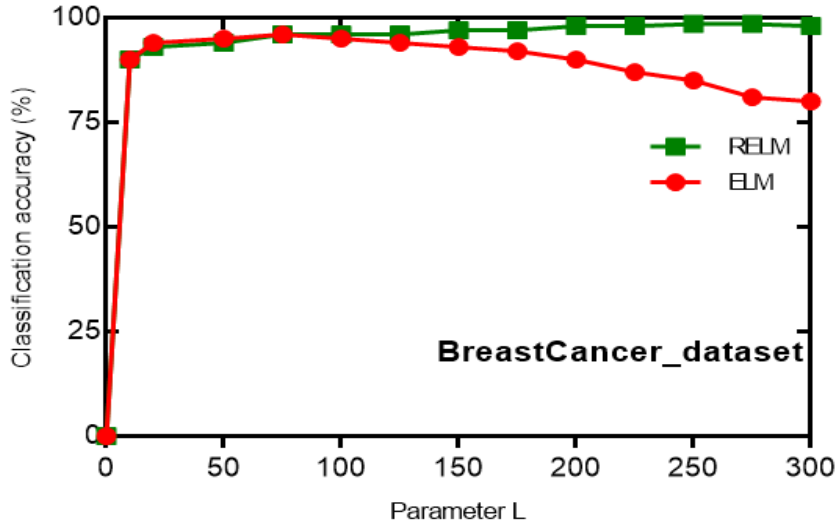


Figure 3.4 Accuracy vs. hidden nodes number relationship for the breast cancer dataset

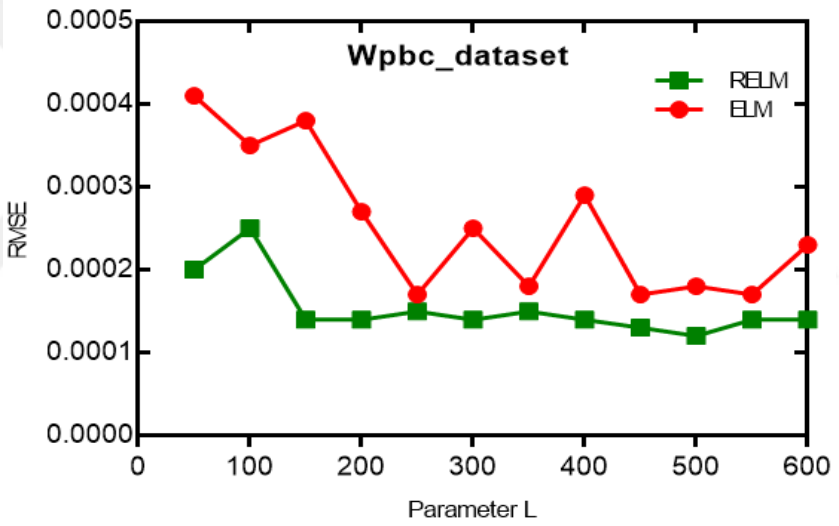


Figure 3.5 RMSE vs. hidden nodes number relationship for the wpbc_dataset

It can observe from Figures 3.4 and 3.5 that the parameter L has very limited impact on the generalization performance of RELM as compared to ELM. Therefore, RELM still retains the benefits of easy tuning than ELM and gains additional features like insensitivity to the overfitting issue, as L is large enough and λ is optimal.

3.6 ELM vs. RELM Comparison

- For suitable L range, RELM is accurate and indeed outperforms ELM.
- RELM is robust and less susceptible to the overfitting issue than the original ELM.
- RELM is less sensitive than ELM to the changing of hidden nodes number L.

- RELM in contrast to ELM provides very consistent results over many trials with varying L, where the regularization effect in the RELM can achieve low fluctuating performance. As a result, RELM has a smooth response.
- RELM has a universal approximation capability on the training dataset which can lead to unified extreme learning machine.

The training process of ELM and RELM are comparable, however RELM requires more operation time than ELM due to two reasons:

i. Producing good performance in RELM needs to adequate large L, which results in enlarge the working matrices and hence increase the required operations.

ii. Finding an optimum regularization parameter (λ_{opt}) in RELM adds extra time to overall operations.

3.7 L-curve with Regularization ELM

In ELM algorithm, if the matrix $H^T H$ or HH^T is close to singular, the pseudo inverse is prone to numerical instabilities and to avoid this problem, it should include a small regularization term (λ).

One may come to the conclusion that RELM is always better and much smoother than ELM, given the same hidden layer is shared among them. However, the ridge parameter (λ) may cause adverse effects to the RELM performance when chosen inappropriately. It needs to study the permission values of the λ that keep the accuracy of the ELM within the acceptable limits, it is clear as shown in Figures 4.14-4.16 (from chapter 4) and Figures 5.12-5.14 (from chapter 5) that applying any λ value is unbeneficial and may lead to reduce the accuracy, therefore finding the optimum λ is necessary to achieve best performance.

Different approaches for calculating suitable regularization parameter (λ) have been employed such as the Discrepancy Principle, Generalized Cross-Validation (GCV), and the L-Curve [110]. The L-curve is a log-log plot that seeks to determine a value of λ that balances between two error components: the solution norm (smallest training error) versus the residual norm of output weight whilst keeping our objective function as:

$$min_{\lambda,\beta} \{\|H\beta - T\|_2^2 + \lambda\|\beta\|_2^2\} \quad (3.14)$$

Like the GCV approach, L-curve method does not depend on specific or prior knowledge of the noise variance. One noted, the difficulty in GCV is that its function can have a very flat minimum, making it difficult to determine the optimal λ numerically. On the other hand, GCV rule may be unstable for correlated noise, resulting in under smoothing [113].

The L-curve method consists of the analysis of the piecewise linear curve whose break-points are:

$$(x_i, y_i) = (\log\|H\beta_i - T\|_2^2, \log\lambda\|\beta_i\|_2^2),\ i = 1\ to\ p \tag{3.15}$$

where p is the row dimension of the regularization matrix.

The L-curve basically consists of two parts: a "flat" part where the regularization errors dominate and a "steep" part where the perturbation error dominates. This curve, in most cases, exhibits a typical "L" shape, and the optimal value of the regularization parameter λ must lie on the corner of the "L", see Figure 3.6 which represents the L-curve graph for a random $H(20 \times 20), T(20 \times 1)$ and the $p = 10$ row dimension where the p-dimension is flexible and depending on the degree of regularization precision. The advantages of using L-curve are: easy and fast to present the set of the two axis points and it is flexible on choosing the number of the represented points, i.e, when accurate result is required large points are needed to represent the L-curve.

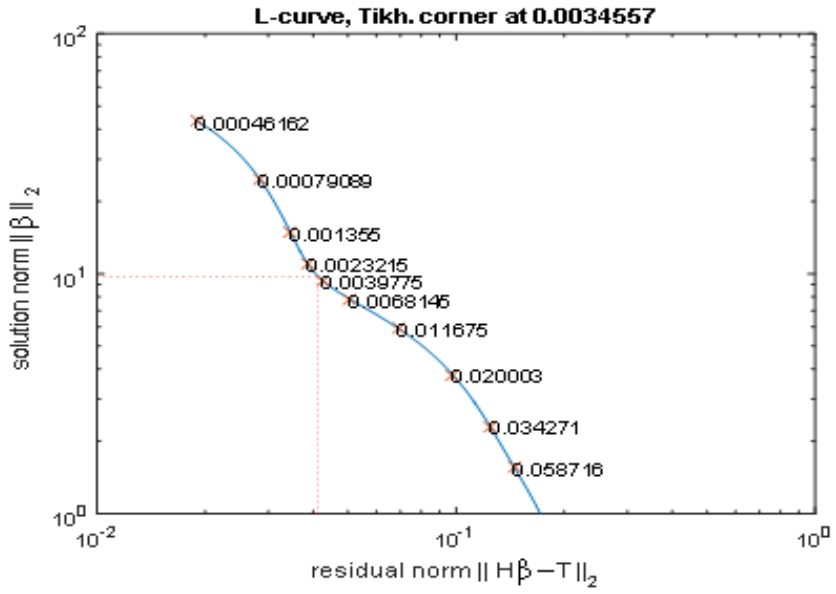


Figure 3.6 An L-curve

Numerically, the L-curve is usually more amenable due to its simplicity, accurately, ability to deal with large scale matrices and its cost is less or equal to other regularization methods [114]. Unlike other methods, the L-curve method doesn't need to predefined

limits of λ, i.e, λ_{min} and λ_{max} where choosing unsuitable predefined λ values will diverge the solution from the optimum λ_{opt}. Instead of manually selecting the hyper-parameters, worthy model selection and parameter optimization approaches can be achieved with high computational consuming LOO-CV, [28, 71, 72] have been adopted efficient computing method depending on the sum of residual errors which is equivalent to (PRESS) to calculate the cross validation minimum square error (MSE_{CV} or MSE^{PRESS}) utilizing equation 3.16:

$$MSE_{CV(K)}=MSE^{PRESS}=\frac{1}{K}\sum_{i=1}^{K}\left(\frac{t_i-\hat{t}_i}{1-(HAT_{ii})}\right)^2 \tag{3.16}$$

Where, $HAT = HH^{\dagger} = H.(H^TH)^{-1}H^T$ (3.17)

HAT_{ii} is the ith diagonal of HAT matrix that transform T into $\hat{T}$ and

$$\hat{T} = HAT.T \tag{3.18}$$

In LOO-CV, estimating the pseudo-inverse and MSE_{CV} by K-repetitive would be computationally expensive, that leads to missing the advantage of the ELM in fast predicting. Incorporating the regularization parameter and various recombination of H_i in the singular value decomposition (SVD) method with a suitable λ extraction approach (L-curve) is used to obtain the various $H^{\dagger}$'s with minimal re-computation and hence the optimum $\lambda_{opt}\ and\ \beta_{opt}$.

Due to variety of the experimental data types, the proposed algorithm can use different subsets H_i depending on random permutation or limited LOO-CV as a valuable approach to overcome the overfitting or using the bootstrapping method if the data samples are small.

Algorithm 3.2. Tikhonov Regularized ELM (TRELM) with L-curve and SVD

Suppose a training set of N samples each with n features is given as $\{(x_i, t_i)|\ x_i \in R^n,\ t_i \in R^m, i = 1, \cdots, N\}$, with any suitable activation function, using $X \in R^{Nxn}$ and $T \in R^{Nxm}$ with equations 2.6 and 2.7 to find the hidden layer output and target matrices $H_{NxL}\ and\ T_{Nxm}$, then permute H into different max random subsets H_i depending on the hypothesizes of different means using for example random permutation or analysis of variance (ANOVA) or Monte Carlo approaches, i.e, $H_i \in H \Longrightarrow \{H_1, H_2, \dots, H_{max}\}$.

Employing H_i with SVD matrix factorization and L-corner curve methods as:

1. For $i = 1{:}\,max$; we used $max = 20$

2. Take H_i and T utilizing L-curve regularization method by employing Hansen Regularization Toolbox Package [115] to find λ_i corresponding to H_i that satisfies $min_{\lambda_i}\{\|H_i\beta - T\|_2^2 + \lambda_i\|\beta\|_2^2\}$
3. loop
4. Since λ_{min} leads to β_{min}, select the minimum λ_i as λ_{opt} and corresponding H_i as the usable H
5. Based on the data dimensions and SVD factorizations, using
    a. H^TH if $N \geq L \Rightarrow SVD\ (H^TH) = VD^2V^T$
    b. HH^T if $L > N \Rightarrow SVD\ (HH^T) = UD^2U^T$ (3.19)
6. Calculating β_{opt} using minimized formula of section (3.10) as:
    a. for $N \geq L$:

$$\beta_{opt} = (H^TH + \lambda_{opt}I_L)^{-1}H^TT = V(D^2 + \lambda_{opt}I_L)^{-1}V^TH^TT$$
$$\beta_{opt} = \sum_{i=1}^{L} v_i \left(\frac{1}{d_{ii}^2+\lambda_{opt}}\right) v_i^T . H^TT \quad (3.20)$$

    b. for $L > N$:

$$\beta_{opt} = H^T(HH^T + \lambda_{opt}I_N)^{-1}T = H^TU(D^2 + \lambda_{opt}I_N)^{-1}U^TT$$
$$\beta_{opt} = H^T . \sum_{i=1}^{N} u_i \left(\frac{1}{d_{ii}^2+\lambda_{opt}}\right) u_i^T . T \quad (3.21)$$

3.8 Experimental Results

The classical ELM was tested with three situations: optimum hidden number, the SVD projection and TR regularization. The experimental results and runtime reported in this section were based on the average of 20 independent trials for regular size datasets. All simulations were implemented using MATLAB 8.1 (R2013a) environment and performed on an (Intel Core i5, 2.4 GHz CPU, 4GB RAM) computer. Eleven datasets from the University of California at Irvine (UCI) Machine Learning Repository [112] were tested: four for classification and seven for regression with 20 different random permutations taken without replacement. Two-thirds of the whole samples for training and one-third for testing and the whole samples were normalized with zero mean and unit standard deviation. Though the training and testing sets are randomly generated, they are kept the same for all comparing algorithms for fair comparison.

The large scale data was handled with extra two data sets [116]; AR face database with 700×300 training and 700×300 testing samples as 60×43 grayscale images, which, belong to 100 individual face recognitions, and USPS (US Postal Service) digits database consisting of handwritten digits from 0 to 9 with 7291×256 training and 2007×256 testing

samples as 16×16 grayscale images. The results showed that OELM, PELM, and TRELM can predict or fit the desired targets rapidly at low error rates even with large datasets.

3.8.1 Standard ELM with Optimum Hidden Nodes (OELM)

Most simple datasets are of root mean square error $\varepsilon \approx 0$, so $\propto = 2^{\varepsilon} \approx 1$ and hence the optimum L_{opt} will be $(2n + m)$ where $m = 1$ for regression. The performance (Accuracy or RMSE ± standard deviation over the computational time) of the primarily ELM learning algorithm was examined and elaborated first with optimum hidden node number (L_{opt}) in equations 3.1 and 3.2 on the eleven UCI datasets as presented in both Tables 3.1 and 3.2. All the random input weights were normalized within the range [-1, 1].

Table 3.1 OELM performance results for classification samples

Data Set/Types	n/m	α	L_{opt}	Train. Data	Test. Data	Train. accuracy (%)/ Time (sec)	Testing accuracy (%)/ Time (sec)
P. I. Diabetes/*	8/2	1.32	23	576	192	78.95±1.23/ 0.007	77.34±2.86/ 0.002
Wine/*	13/3	1.20	34	115	63	99.52±0.66/ 0.0044	96.04±1.89/ 0.003
Segment/**	19/7	4.31	194	1500	810	96.27±0.3/ 0.15	94.77±0.57/ 0.039
Satellite/**	36/7	2.32	183	4435	2000	93.97±0.49/ 0.35	89.44±0.58/ 0.065

* (small data type) and ** (moderate data type)

Table 3.2 OELM performance results for regression samples

Data Set/Type	n	α	L_{opt}	Train Data	Test. Data	Training RMSE/ Time (sec)	Testing RMSE/ Time (sec)
Stock/*	9	1.05	20	634	316	0.0993± 0.0118/ 0.002	0.0934±0.009/ 4.0e-04
Breast Cancer/*	32	1.07	69	130	64	0.0983± 0.0026/ 0.007	0.103± 0.005/ 3.8e-04
Bank/*	8	1.01	17	3000	1500	0.012±2.36e-04/ 0.017	0.012± 3.6e-04/ 0.004
Ailerons/**	40	2	162	4770	2384	2.0e-05± 2.9e-06/ 0.06	2.0e-05± 0.2e-06 /.005
D_Elevators/**	6	2.3	30	6345	3172	1.7e-04± 2.1e-05/ 0.04	1.8e-04± 2.3e-05/ 0.01
Elevators/**	18	2	74	5835	2917	1.0e-04± 1.8e-06/ 0.13	0.8e-04±2.7e-6/0.02
Kinematics/**	8	2.9	49	5462	2730	0.202± 0.0093/ 0.017	0.204±0.0086/ 0.0057

* (small data type) and ** (moderate data type)

Figures 3.7a, 3.8a, 3.9a show the relationship between hidden nodes number with average testing accuracy where each value of accuracy represents the average of 20 different random permutations, Figures 3.7b, 3.8b, 3.9b estimate the optimal hidden nodes size

using k-fold cross validation with k=10 inside the testing set. Figures 3.7(a & b) is for the 'Pima Indians Diabetes' classification data set, it is clear that the best testing accuracy is at $L_{opt} = 20$ from the average test and 23 from LOO-CV test. Figures 3.8(a & b) are for the 'Wine' data set that produces the highest accuracy with the lower optimum hidden node number ($L_{opt} = 34$ or 35) while Figures 3.9(a & b) are for the 'Segment' data set which generates the first minimum optimum hidden nodes at $L_{opt} = 200$.

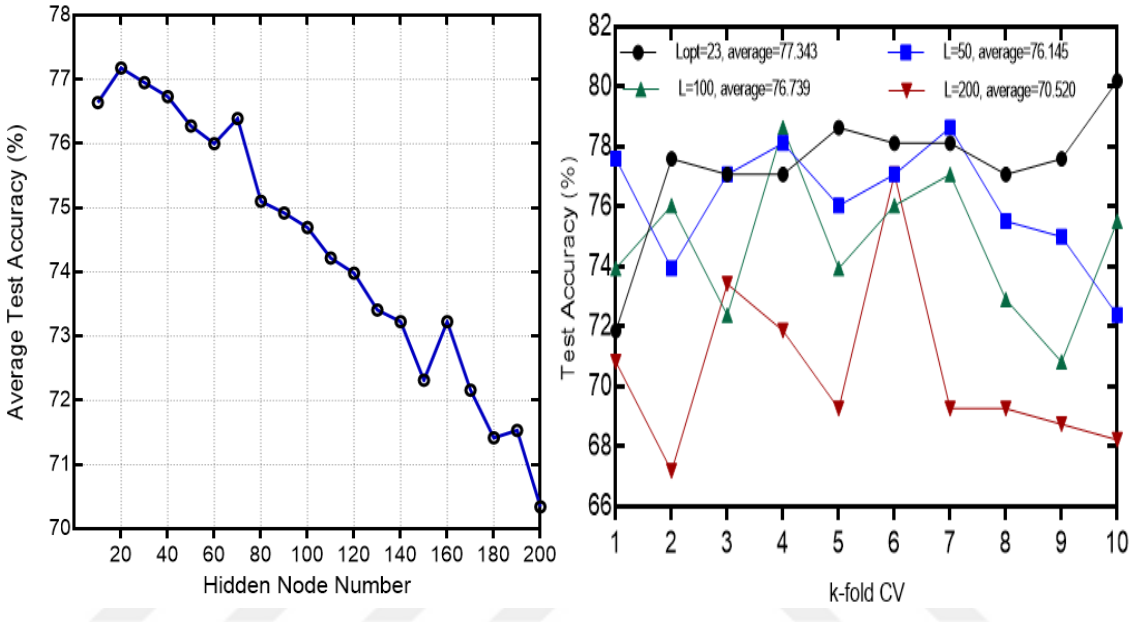


Figure 3.7 The relationship between testing accuracy with (a). hidden nodes number (b). the k-fold number for Pima Indians Diabetes data set

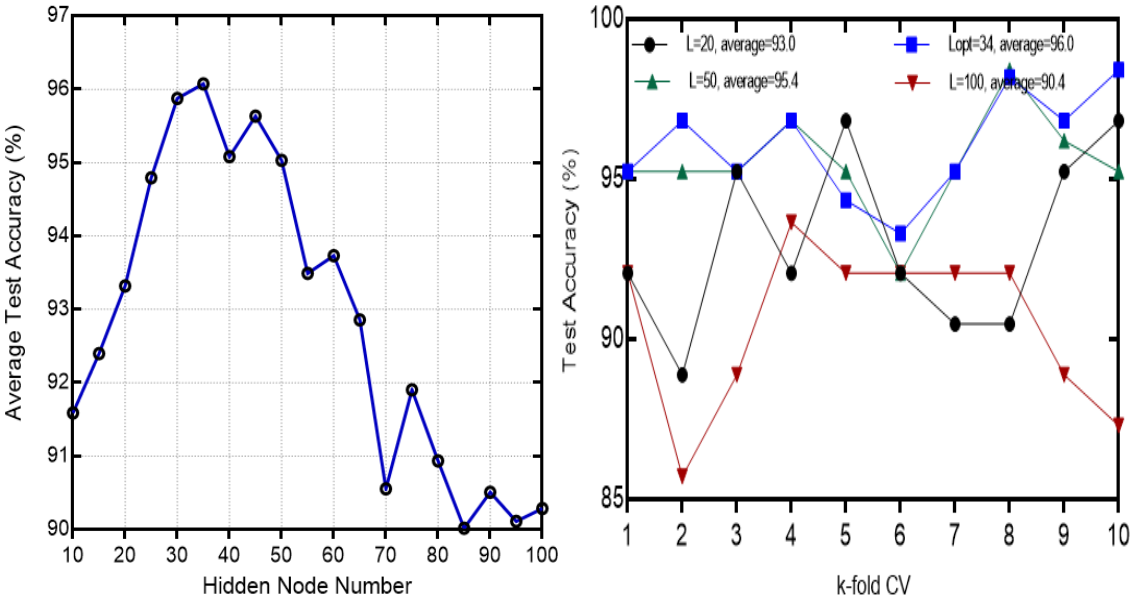


Figure 3.8 The relationship between testing accuracy with (a). hidden nodes number (b). the k-fold number for Wine data set

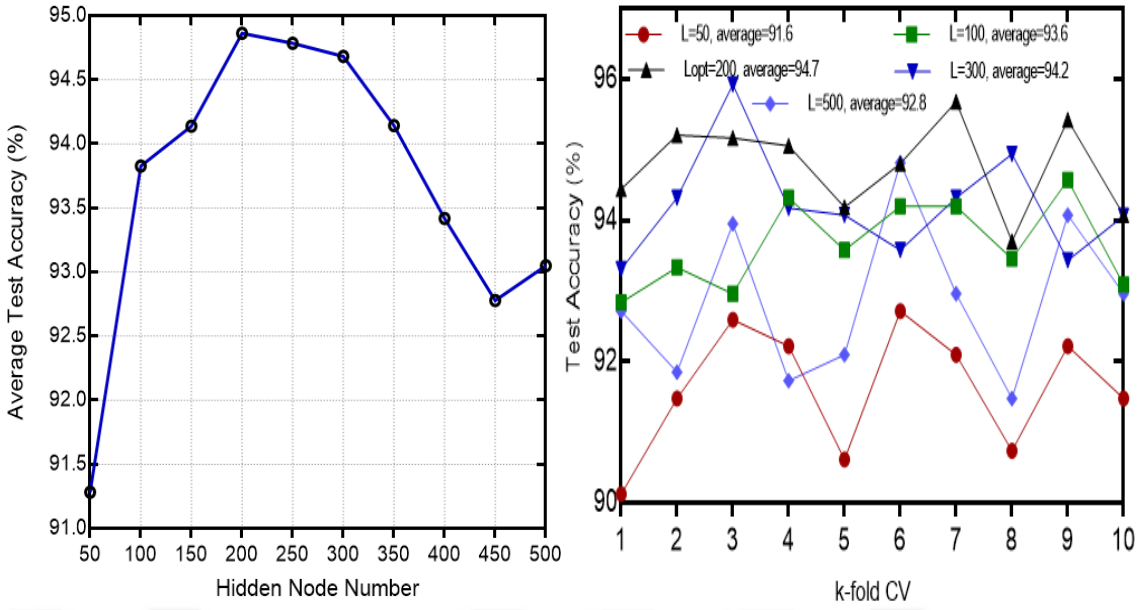


Figure 3.9 The relationship between testing accuracy with (a). hidden nodes number (b). the k-fold number for Segment data set

3.8.2 Projection ELM (PELM)

Mathematically, in ELM there are only matrices describing the projection between the two linear spaces: input $X.w$ and output $H.\beta$, with a nonlinear transformation between them $H = G(w.X + b)$. The number of hidden neurons (L) regulates the size of matrices w, H and β, where the network neurons are never treated separately. The goal of Projection ELM (PELM) is to analyze input orthogonal projection subspaces of $X.w$ feature mapping to extract and handle only significant features that can enhance the overall performance of classical ELM. PELM was tested on same previous data bases (classification type only), the simulation results are in line with earlier standard ELM which confirmed an increase in the accuracies with or without a slight increase in training time.

The PELM results for the accuracy and computational time are reported in the Table 3.3 which reveals an improvement of about 2% or more in accuracy even though the computational time has no impact.

Table 3.3 PELM performance results

Data Set/ Types	n /m	α	L_{opt}	Training Data	Testing Data	Train. accuracy (%) / Time(sec)	Test. accuracy (%) / Time(sec)
P.I.Diabetes/*	8/2	1.32	23	576	192	79.91±0.9/ 0.0063	78.7±2.12/ 0.002
Wine/*	13/3	1.20	34	115	63	99.7±0.39/ 0.0047	97.62±1.42/ 0.0045
Segment/**	19/7	4.31	194	1500	810	97.06±0.17/ 0.165	96.81±0.40/ 0.04
Satellite/**	36/7	2.32	183	4435	2000	95.38±0.37/ 2.145	91.73±0.47/ 0.1709

3.8.3 L-curve Tikhonov Regularization ELM (TRELM)

In ELM, choosing an appropriate or optimum regularization parameter (λ_{opt}) with a good efficiency and speed is crucial. Tikhonov Regularization of ELM approach (TRELM) which is examined numerically as a robust method by least square solution can be analytically derived.

The L-curve is a logarithmic calibration curve between the residual error norm and the solution norm with λ as the parameter ($\log \|H\beta - T\|_2^2, \log \lambda \|\beta\|_2^2$). As observed from Figure 3.10 for the segment image data base, the best regularization parameter was found to be ($\lambda_{opt} = 0.02817$) calculated from the corner of the curve that tries to suppress as much as possible the influence of both error norms at the same time. This gives a regularized solution β_λ that is as close as possible to the exact solution.

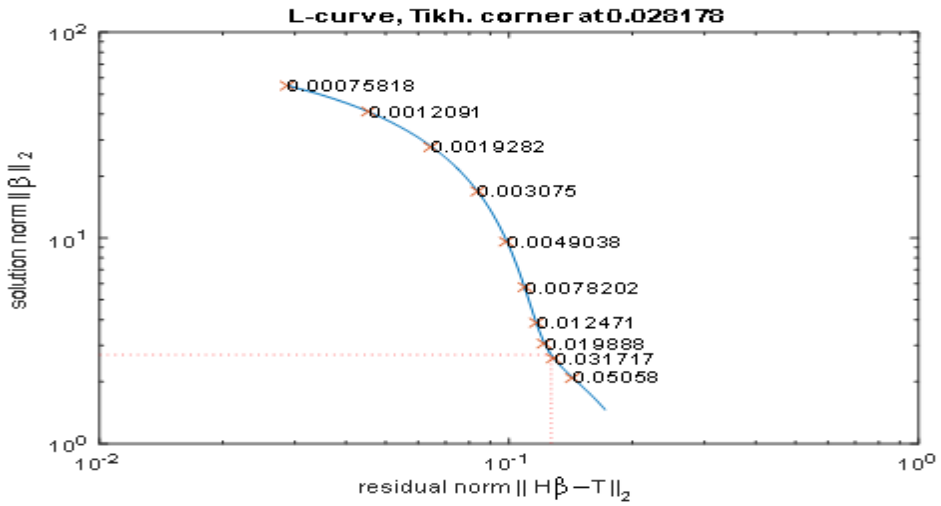


Figure 3.10 L-curve regularization method for Segment image data base

The performance of the proposed TRELM model was carried out on all types of classification/ regression data sizes (small, moderate, and large) problems as stated in Tables 3.4 and 3.5.

Table 3.4 TRELM performance results for classification data sets

Data Set	Type	λ_{opt}	L	Train.Acc./Time(sec)	Test. Acc. / Time(sec)
P. India Diabetes	Small	0.1108	1300	81.4±1.107/ 0.7970	75.5±1.97/ 0.0058
Wine	Small	0.0183	2100	97.4±0.48/ 0.0204	98.41±1.38/ 0.0033
Segment	Moderate	0.0281	1100	97.1±0.19/ 3.3280	95.43±0.43/ 0.0289
Satellite	Moderate	0.4493	1900	94±0.31/ 19.6933	90.1±0.37/ 0.2919
AR	Large	0.2466	2100	100±0.27/ 1.5643	93.78±0.36/ 0.0712
USPS	Large	54.598	2100	99.7±0.73/ 34.67	94.94±0.91/ 0.1733

Table 3.5 TRELM performance results for regression data sets

Data Set/ Types	λ_{opt}	L	Train. RMSE / Time (sec)	Test. RMSE/ Time (sec)
Stock/*	0.00034	250	0.0459± 0.0011/ 0.1535	0.0540±0.0025/ 0.0026
Breast Cancer/*	variable	>100	0.0982±0.0060/ 0.0279	0.1078±0.0078/ 0.0062
Ailerons/**	0.06081	150	8.16e-06±5.96e-07/ 0.7541	8.4e-06± 6.44e-07/ 0.0141
Bank/**	0.00248	120	6.448e-04±6.90e-05/ 0.384	0.002±1.486e-04/ 0.0058
D-Elevators/**	0.04979	120	1.39e-04±1.82e-06/ 0.8191	1.40e-04±2.42e-06/ 0.0133
Elevators/**	0.02237	260	9.68e-05± 1.41e-06 / 1.589	9.88e-5±2.90e-06/ 0.0258
Kinematics/**	0.07427	800	0.0952± 0.0022/ 5.8260	0.1134±0.0038 /0.0787

Tables 3.4 and 3.5 reveal that for both moderate and large data experiments, TRELM has lower error rates with largely hidden node number in classification and normally hidden node number in regression. The TRELM approach is applicable to moderate and big data matrices in order to provide robust and optimum solutions, also it can tackle a wide range of machine learning problems. Moreover, for the small dataset, the TRELM accuracy/RMSE is comparable to OELM approach but with longer time, i.e, TRELM isn't suitable to the small size datasets where it does not significantly improve the accuracy/RMSE or the time through increasing the hidden nodes size.

For all data types, it is obvious that ELM with regularization needs more time than without it because it operates properly with high hidden nodes, also TRELM has more stable response which means it has the lowest average standard deviations than with OELM and PELM. For 'USPS' and 'AR' data sets, Figures 3.11 and 3.12 show the average accuracy comparison over many runs among the three proposed approaches

(OELM, PELM, and TRELM). It is apparent that OELM and PELM do not have a linear relationship as in TRELM with respect to hidden nodes number.

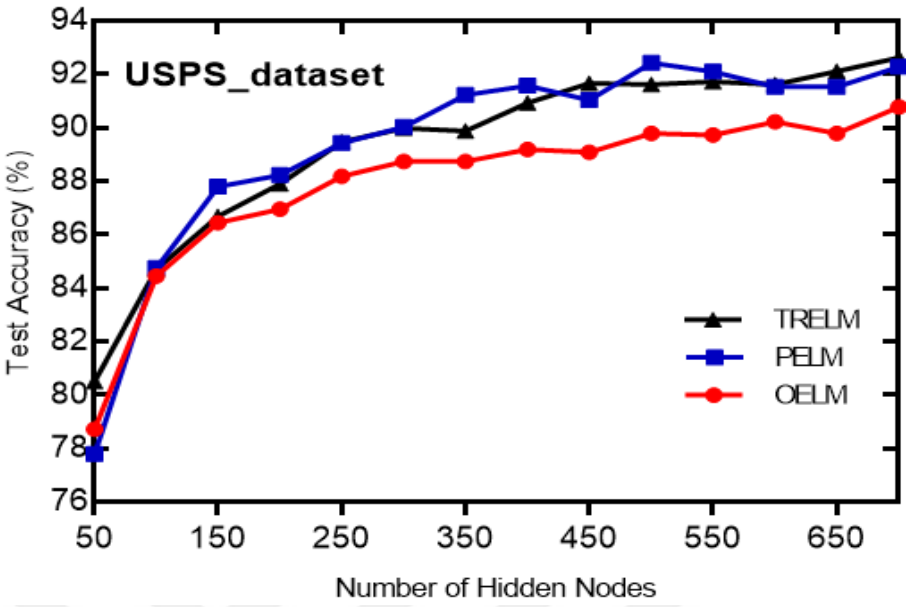


Figure 3.11 All ELM approaches accuracies for USPS dataset

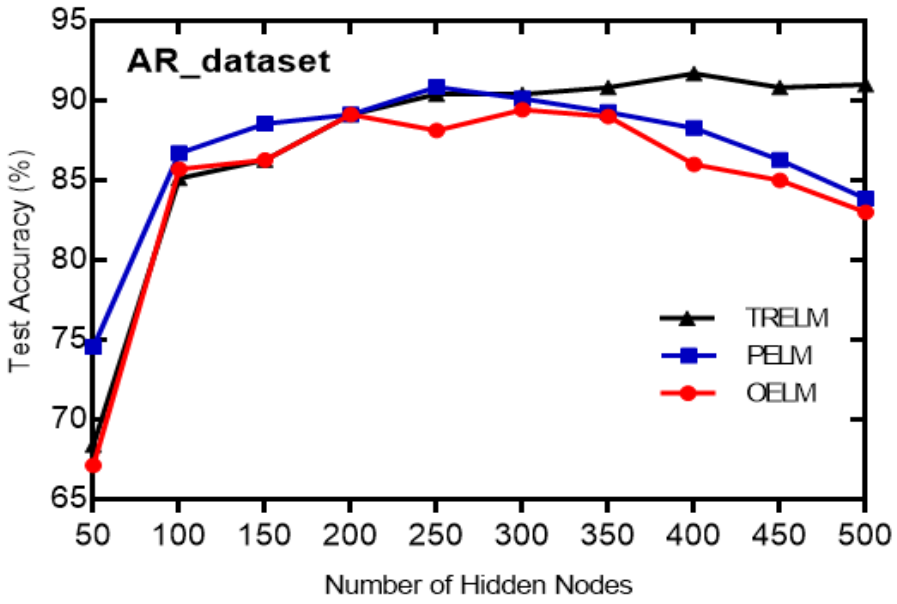


Figure 3.12 All ELM approaches accuracies for AR dataset

Many researches [16, 19, 105, 106] have demonstrated the superiority and effectiveness of the classical ELM and its improved variety approaches. It has been considered as the fastest nonlinear predication approach with better or comparable generalization and performance to the state-of-the-art learning algorithms, such as k-Nearest Neighbor

Classifier (k-NN), Linear Discriminant Analysis (LDA), linear and least square Support Vector Machine (LSVM and LS_SVM) [117], Logistic Regression Classifier (LR) [118], Collaborative Representation based Regularized Least Square (CRC_RLS) [119]. For fair comparison, we compare the TRELM experimental paradigm with above learning methods on AR and USPS face databases as in Tables 3.6 and 3.7.

Table 3.6 Performance comparison for AR data set

Classifier Method	**k-NN**	**LDA**	**LSVM**	**LR**	**CRC_RLS**	**TRELM**
Testing Accuracy (%)	71.57	84.71	75.85	77.42	**93.62**	**93.78**
Testing Time (sec)	0.193	0.421	3.987	0.144	NA	**0.071**

Table 3.7 Performance comparison for USPS data set

Classifier Method	**k-NN**	**LDA**	**LSVM**	**LR**	**CRC_RLS**	**TRELM**
Testing Accuracy (%)	92.82	80.61	93.37	91.18	93.78	**94.94**
Testing Time (sec)	4.86	0.373	8.102	0.282	NA	**0.1733**

As shown in the Tables 3.6 and 3.7, RELM has the best accuracy and testing time amongst all compared algorithms. However, in the case of noisy data, all SRC-associated classifiers consistently provide better recognition accuracy than RELM.

3.9 Minimized RELM

Because of the regularization effect in RELM, more regressors are employed to suppress the overfitting issue. Furthermore, the randomness problem that producing the fluctuations is also significantly reduced and the generalization performance of RELM is improved. With the introduction of RELM, it appears that two parameters need to be tuned L and λ, the L can be easily predicted, as long as it is large enough, it needs not to be tuned. Compared to ELM, the only drawback of RELM is the additional processing expended that slow down the learning speed. To estimate the additional required time, the output weights β for ELM and RELM having time complexity as equation 3.22:

$$(2L^2 + 4L - 1)N + L^3 + L^2 - 2L \tag{3.22}$$

Clearly, the number of operations required to train the ELM and RELM increases dramatically with larger L. In almost RELM researches, the L is uniformly selected to

1000, which is 10 times bigger than the one normally used in ELM. Therefore, the training procedure of RELM is much slower than ELM. As shown in equation 3.22, the most computationally operations that consume the most training time are $O(2L^2 \times N)$ and $O(L^3)$. To reduce the $2L^2 \times N$ part, we need to identify the non-relevant samples in deciding the decision boundary and hence reduce the number of training samples N without degrading the performance result. Also, to decrease the matrix inversion requirement part L^3, it can employ efficient matrix factorizations like LU-decomposition that can enhance mathematically this part.

3.9.1 Leave-One-Out Cross-Validation (LOO-CV)

Selecting the model parameters with good generalization performance using Leave-One-Out Cross-Validation (LOO-CV) is widely used. LOO-CV is usually unbiased and reliable as long as the training and testing data are drawn from the same source. The LOO-CV is known as a slow execution speed method (i.e, for *N* models, $N-1$ would need to be trained) therefore, it is rarely implemented in rapid algorithms and practical applications. The PREdiction Sum of Squares (PRESS) formula (i.e, the errors of the trained model on the training set) can greatly improve the efficiency of the LOO-CV procedure, it can be used for both the linear and nonlinear input variables.

PRESS has been used in many ELM algorithms [28, 71, 72, 120]. However, even though PRESS formula significantly speeds up the LOO-CV process, the computing time of valuing the ELM parameters and selecting the best one is still quite long, as a result, it still threatens the fast learning speed feature of ELM.

The PRESS [28, 120] as a local mean integrated squared error can be derived easily as:

$$PRESS = \frac{1}{N}\sum_{i=1}^{N}(t_i - \hat{t}_{i-1})^2 = \frac{1}{N}\sum_{i=1}^{N}(\epsilon_{i-1})^2 \tag{3.23}$$

where ϵ_{i-1} denotes the leave-one-out error when sample i is left out (also known residual), it is calculated as follow:

$$\epsilon_{i-1} = \frac{\epsilon_i}{1-h_i.(H^TH)^{-1}h_i{}^T} = \frac{t_i-\hat{t}_i}{1-h_i.(H^TH)^{-1}h_i{}^T} = \frac{t_i-\hat{t}_i}{1-h_{ii}}$$

$$PRESS = \frac{1}{N}\sum_{i=1}^{N}(\frac{t_i-\hat{t}_i}{1-h_{ii}})^2 = \frac{1}{N}\sum_{i=1}^{N}(\frac{t_i-\hat{t}_i}{1-HAT_{ii}})^2 = MSE^{PRESS} = error_{LOO} \tag{3.24}$$

where h_{ii} is the diagonal elements of the HAT matrix, i.e, HAT_{ii}.

Once the model is trained, all the PRESS residuals can easily be obtained using equation 3.24. Obviously, this involves a lot less computational than training the model for all N possible training set.

3.9.2 RELM with PRESS and SVD

Even though we successfully exploited the L-curve as an efficient approach for finding λ and practically in many applications λ is the only parameter that required to be tuned in RELM, it may extend the searching process to find the optimum L in addition to λ. Also, in many ELM variants, more than two parameters are needed to be tuned. For example in Graph ELM (GELM) [121, 122, 35] three parameters are varied (L, λ_1 and λ_2). In these cases, needing to efficient mathematical algorithm with fast tuning is vital and increased.

In this section, efficient LOO-CV formula based on PRESS and SVD is developed. It can evaluate the best parameters of RELM with fast learning speed without or with minimal user intervention. It can be easily used by the users and implemented in automatic processes as follow:

First, exploiting the algebraically productive properties of the ELM with SVD especially the core of the HAT matrix which is $H^T H$ or HH^T. Here by utilizing the SVD properties, HAT matrix can be altered to easier and fast mathematical form. Second, instead of repeatedly calculating the LOO-CV, it sequentially updates the result using the PRESS error formula. Then, HAT-matrix requires to compute $\mathrm{H}^{\dagger}$ during the training stage and its computation can be reused for estimating the LOO error (MSE^{PRESS}). Third, $PRESS$ only needs the diagonal of the HAT-matrix, which can be easily obtained by the row-wise dot-product between H and $\mathrm{H}^{\dagger}$.

According to the dimensionality of the applied data (N) and the hidden nodes number (L), there are two cases with either $N \geq L$ or $> N$:

❖ $N \geq L$

Instead of using the standard training procedure, it can utilize SVD to reduce the computing costs for all system parameters as:

➢ The output weight β:

$$\beta = (H^T H + \lambda I)^{-1} H^T T \text{ , let } H = UDV^T \tag{3.25}$$

$$\beta = (VD^T U^T UDV^T + \lambda VV^T)^{-1} H^T T$$

$$\beta = (V(D^T D + \lambda I)V^T)^{-1} H^T T = V(D^2 + \lambda I)^{-1} V^T H^T T$$

$$\beta = V. \begin{bmatrix} \frac{1}{d_{11}^2 + \lambda} & 0 & \cdots & 0 \\ 0 & \frac{1}{d_{22}^2 + \lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{d_{LL}^2 + \lambda} \end{bmatrix} . V^T H^T T$$

then β can be efficiently computed as:

$$\beta = V. \sum_{i=1}^{L} \left(\frac{1}{d_{ii}^2 + \lambda} \right) . V^T H^T T \quad (3.26)$$

It can also modify equation 3.27 to more concise form as:

$$\beta = V. \sum_{i=1}^{L} \left(\frac{1}{d_{ii}^2 + \lambda} \right) . V^T \mathrm{V} D^T U^T T = V. \sum_{i=1}^{L} \left(\frac{d_{ii}}{d_{ii}^2 + \lambda} \right) . U^T T \quad (3.27)$$

- The estimated output $\hat{T}$ and HAT:

$$\hat{T} = H\beta = HV. \sum_{i=1}^{L} \left(\frac{1}{d_{ii}^2 + \lambda} \right) . V^T H^T T = HAT. T \quad (3.28)$$

So, the HAT matrix that used in PRESS formula can be expressed as:

$$HAT = HV. \sum_{i=1}^{L} \left(\frac{1}{d_{ii}^2 + \lambda} \right) . V^T H^T = UDV^T V. \sum_{i=1}^{L} \left(\frac{1}{d_{ii}^2 + \lambda} \right) . V^T \mathrm{V} D^T U^T$$

$$HAT = U. \sum_{i=1}^{L} \left(\frac{d_{ii}^2}{d_{ii}^2 + \lambda} \right) . U^T \quad (3.29)$$

PRESS formula needs only to the diagonal elements of HAT, i.e,

$$HAT_{ii} = \sum_{i=1}^{L} \left(\frac{d_{ii}^2}{d_{ii}^2 + \lambda} \right) . u_{ii}^2 \quad (3.30)$$

After computing the $\hat{T}$ and HAT from equations 3.29 and 3.30, it can apply the PRESS formula and selecting the optimal λ that corresponding to the minimum MSE^{PRESS} error.

❖ $L > N$

- The output weights β:

$$\beta = H^T (HH^T + \lambda I)^{-1} T \text{ , let } H = UDV^T \quad (3.31)$$

$$\beta = H^T (UDV^T VD^T U^T + \lambda UU^T)^{-1} T$$

$$\beta = H^T (U(DD^T + \lambda I)U^T)^{-1} T = H^T U(D^2 + \lambda I)^{-1} U^T T$$

$$\beta = H^T U. \begin{bmatrix} \frac{1}{d_{11}^2+\lambda} & 0 & \cdots & 0 \\ 0 & \frac{1}{d_{22}^2+\lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{d_{NN}^2+\lambda} \end{bmatrix} U^T.T$$

then β can be efficiently computed as:

$$\beta = H^T U. \sum_{i=1}^{N} \left(\frac{1}{d_{ii}^2+\lambda}\right). U^T T \tag{3.32}$$

Equation 3.33 can be abbreviated as:

$$\beta = VD^T U^T U. \sum_{i=1}^{N} \left(\frac{1}{d_{ii}^2+\lambda}\right). U^T T = V. \sum_{i=1}^{N} \left(\frac{d_{ii}}{d_{ii}^2+\lambda}\right). U^T T \tag{3.33}$$

- The estimated output $\hat{T}$ and HAT:

$$\hat{T} = H\beta = HH^T U. \sum_{i=1}^{N} \left(\frac{1}{d_{ii}^2+\lambda}\right). U^T T = HAT.T \tag{3.34}$$

So, the HAT that used in PRESS formula can be expressed as:

$$HAT = HH^T U. \sum_{i=1}^{N} \left(\frac{1}{d_{ii}^2+\lambda}\right). U^T = UDV^T VD^T U^T U. \sum_{i=1}^{N} \left(\frac{1}{d_{ii}^2+\lambda}\right). U^T$$

$$HAT = U. \sum_{i=1}^{N} \left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right). U^T \tag{3.35}$$

Also, $HAT_{ii} = \sum_{i=1}^{N} \left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right). u_{ii}^2$ (3.36)

The above steps and equations can be illustrated briefly by the following algorithm.

Algorithm 3.3. Regularized ELM (RELM) with PRESS and SVD

Suppose a training set of N samples each with n features is given as $\{(x_i, t_i) | \, x_i \in R^n, t_i \in R^m, \; i = 1, \cdots, N\}$, with any suitable activation function, using $X \in R^{Nxn}$ and $T \in R^{Nxm}$ with equations 2.6 and 2.7 to find the hidden layer output and target matrices H_{NxL} *and* T_{Nxm},

1. Factorize H using SVD with $H = UDV^T$,
2. For $\lambda = \lambda_{min} : \lambda_{max}$; choosing a proper range for λ
3. Find HAT, HAT_{ii}, and $\hat{T}$ as:

Either $\geq L$: $HAT = U.\sum_{i=1}^{L}\left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right).U^T$, $HAT_{ii} = \sum_{i=1}^{L}\left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right).u_{ii}^2$,

Or $L > N$: $HAT = U.\sum_{i=1}^{N}\left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right).U^T$, $HAT_{ii} = \sum_{i=1}^{N}\left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right).u_{ii}^2$,

then $\hat{T} = HAT.T$

4. Find the error using $MSE^{PRESS} = \frac{1}{L}\sum_{i=1}^{L}(\frac{t_i-\hat{t}_i}{1-(HAT_{ii})})^2$ or $\frac{1}{N}\sum_{i=1}^{N}(\frac{t_i-\hat{t}_i}{1-(HAT_{ii})})^2$
5. loop
6. $\lambda_{opt} = argmin_{\lambda}[MSE^{PRESS}]$, i.e, select the λ_{opt} that corresponding to min. MSE^{PRESS}.
7. Find β with:

   Either $\beta = V.\sum_{i=1}^{L}\left(\frac{d_{ii}}{d_{ii}^2+\lambda}\right).U^T T$ for $N \geq L$

   Or $\beta = V.\sum_{i=1}^{N}\left(\frac{d_{ii}}{d_{ii}^2+\lambda}\right).U^T T$ for $L > N$

CHAPTER 4

GRAPH BASED EXTREME LEARNING MACHINE WITH L21-NORM REGULARIZATION

4.1 Introduction

Recently, the topic of improving the compactness of ELM has attracted great interest. Extreme learning machine (ELM) is a single hidden layer feedforward neural network (SLFN) with a sufficient number of hidden neurons and almost any nonlinear activation function. The ELM algorithm is a learning approach formed by two processing steps. The first step is the mapping process that converts the input space R^n or reduced input space R^d to a usually high-dimensional feature space R^L (noted as ELM space) with preserving some training data properties, i.e, ($X \in R^{N\times n}$ or $X \in R^{N\times d} \Rightarrow H \in R^{N\times L}$). The second step is the optimization scheme process that projects high-dimensional feature space R^L to a low projection linear space R^m, i.e, ($H \in R^{N\times L} \Rightarrow T \in R^{N\times m}$). Here, two problems are arisen; the first is how to choose the optimum L that determine the proper dimensionality of the feature space (this problem and solution were described in detail previously in chapter 3). The second problem is how to preserve the discriminative features of the input data from losing after inserting to the non-linear activation functions.

Due to the nonlinear mapping of the activation functions and most Subspace Learning (SL) approaches that may destroy the local consistency properties in the dataset, the graph Laplacian was intervened into the optimization function as a demand to exploit the class similarity advantages of intrinsic construction that is certainly increasing the overall model accuracy. Mathematical approaches including subspace learning, graph Laplacian, and L_{21}-norm regularization is combined effectively with baseline ELM to produce a novel extension of ELM termed Graph based ELM with L_{21}-norm Regularization (L21GELM) that can improve the robustness and compactness of ELM. L21GELM

exploits jointly the discriminative information, geometric structure in manifold and local neighborhood around each data point and merges efficient approaches like graph Laplacian and L_{21}-norm regularization to improve the robustness and compactness of the Graph Extreme Learning Machine (GELM). The main purpose of L21GELM is to enhance the label consistency which enforces training samples from the same class to generate similar ELM network outputs.

4.2 Related Works

4.2.1 Feature Extraction

Feature Extraction is investigated as good way of reducing the number of features under consideration. In other words, the data with a high dimensions feature space can be transformed to a fewer dimensions feature space. The transformation methods can be linear, as in the case of Principal Component Analysis (PCA) [123, 124], or nonlinear techniques as exist in [125, 126]. In widely used images, adjacent spatial neighbourhood pixels consist of different patches have a local or global embedding structure. Therefore, many subspace learning methods take advantage of the geometrical structure of input data by minimizing the nearby points that are close to each other in transformation space. The recognition precision and speed could be enhanced significantly by constructing a low-dimension space with discriminative features.

The popular linear and kernel dimensionality reduction algorithms that are employed in this work including: Principal Component Analysis (PCA) and Kernel Principal Component Analysis (KPCA) [127], Linear Discriminant Analysis (LDA) [128], Generalized or Kernel Discriminant Analysis (GDA/KDA) [129, 130], Linearity Preserving Projection (LPP) [131–133], Neighbourhood Preserving Embedding (NPE) [134, 132], and Linear Graph Embedding (LGE) [132], which can extract distinct features by projecting the origin high dimensionality of the source image into low dimensionality spaces.

Although feature extraction is recognized as a flexible tactic, it attempts to construct some new joint features, which may be not invariant under transformation, i.e, resulting in inconsistent feature space. For this reason, there are variants in the resultant accuracies when working with different SL methods. Feature selection, on the other hand, concentrates on finding a small feature subset among the original feature vectors.

4.2.2 Feature Selection

Real applications like image retrieval, text mining and bioinformatics usually contain a lot of features which incur very high computational costs for analysis. Pruning noisy or non-informative features or/and select the most significant features can usually improve their prediction performances. Moreover, a small set of features is beneficial to visualize or interpret the prediction results. Beside to the feature extraction methods that can be used as dimensionality reduction methods, there are feature selection which extensively used to obtain the suitable or optimum features among the all features without altering the data or used any transformation.

The available feature selection methodologies in the literature are often classified into the *filter*, *wrapper* and *embedded* methods. Filter methods such as Minimum Redundancy Maximum Relevance (MRMR) [135], Fast Correlation-Based Filter (FCBF) [136] and Redundancy Constrained Feature Selection (RCFS) [137] are depended on feature correlation and mutual information metrics and based on statistical tests ; thereby, small-significant feature subsets are rapidly produced with trustworthy accuracy performance. However, when big dimensional datasets are processed, filter methods are not a good choice and indeed less attractive in comparison to others due to the scale shortcomings of feature correlation or class dependency computations when dimensions increase. Both the wrapper and embedded methods are not dependent on the correlation among the features in the search. Wrapper methods evaluate feature subsets based on some search strategy with respect to the classification algorithm; therefore, determines the performance based on classification accuracy [138]. In other words, they can identify the feature subset that produces the optimum true classification rate. Genetic Algorithm (GA) [139] and its variants, Support Vector Machine based on Recursive Feature Elimination (SVM-RFE) [140–143] are widely used among the wrapper algorithms. The computations in wrapper algorithm are much complex in comparison to filter methods; nevertheless, the efficiency of the selected subsets are higher even if they are not optimum. Embedded Methods, on the contrary, are different from other former methods where the searching mechanism is straight embedded within the classifier model used. Incorporating previous information regarding the regression function or structure of the classification plays a crucial role in finding the best feature set. The L1-Sparse Support Vector Machines (L1-SVM) [144–146] are well known wrapper methods that effectively combine L_1 regularized risk functions with the SVM optimization function. The efficiency of the L_1-regularization is often low when large-scale and high dimensional

datasets are handled. Therefore, Feature Generating Machine (FGM) [147] was introduced to tackle the above problems. Embedded methods have been demonstrated to deal effectively with very high dimensional datasets. Nowdays, many studies have concentrated on introducing novel feature selection approaches to minimize the redundancies as much as possible between the selected features.

Mainly in ELM, two scenarios can be employed to produce slightly feature selecting: the first which belongs to the embedded class is to build efficient regularization or adopt mathematical model within the ELM optimization function like the L_{21}-norm [148] that can select the robust features. While the second approach, which belongs to wrapper class, is to weight the hidden nodes and neglect those have little contributions to the performance. Dynamic adjustment of hidden nodes parameter using Orthogonal Least-Squares (OLS) was adopted in [149], it enables for a better selection of most appropriate hidden nodes from a large set of candidates results in reduction in the residual error.

4.2.3 The Regularization

To attain better generalization, a lot of theoretical supports in the traditional supervised learning can be adopted to minimize the empirical loss of training data. It will be beneficial to take into account different penalties for different errors. These penalties are based on the L_P-norm which can be described using the formula below [150]:

$$\|x\|_{L_P} = (\sum_{i=1}^{N} |x_i|^P)^{\frac{1}{P}} \tag{4.1}$$

where x is a N-dimension vector. By the definition of the L_P-norm, it can easily define the L_0-norm, L_1-norm and L_2-norm.

The L_2-norm is equivalent to the Euclidean distance with $\|x\|_{L_2} = \sqrt{< x, x >}$.

Also, L_0-norm is not really a norm because it does not have some properties of the norm, it is used to count the nonzero elements number of vector , it is defined as follows:

$$\|x\|_{L_0} = \lim_{P \to 0} \|x\|_P^P \tag{4.2}$$

The L_P-norm with $P = 1/2$, $P = 1$, and $P = 2$ can be seen in Figure 4.1

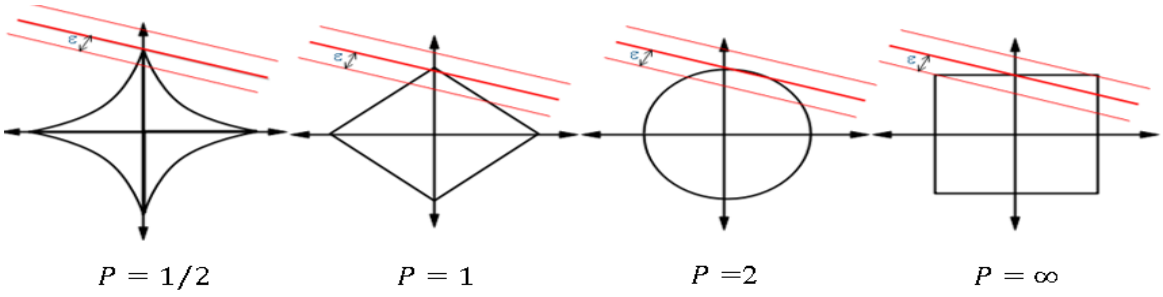


Figure 4.1 Different regularization types

From the Figure 4.1, (ε) is the empirical loss which represents the area under the curve, it can also be seen that the L_P-norm is not smooth within the range of $0 < p < 1$, and has vertices at the coordinate axes; so, it is employed as the regularization term to get the sparse solution. However, it is difficult to present the area under the curve ε since it is nonconvex; thereby, it is not appropriate for obtaining the global optimum solution. On the other side, when $p > 1$, the L_P-norm is convex and smooth (easy to present the area under the curve ε), subsequently, it is employed as the regularization term to obtain the global optimal solution. Nevertheless, it is smooth and not appropriate to determine the sparse solution. The situation where $p = 1$ can be considered the alone case for L_P-norm to keep the convex but in hardly manner and sparse properties.

It can summarize all the regularization norms as: L_2 regularization is considered as important regularized criteria for learning, it is easy to achieve a remarkable generalization performance. However, L_2 represents least square regression solution that suffers from noisy data and cannot detect or exclude the outliers, this is due to the capability of L_2-norm to minimize the summation of squared weights as much as possible without force them zeroes. As a result, L_2 provides no additional advantage in terms of sparsity. On the other hand, L_1 regularization can reject noisy features efficiently with the advantage of sparsity but with complex convex solution. Empirically, the L_1-norm regularization is suitable for regression and binary classification, it is computationally cost for multi-class classification. L_{21} or $L_{2,1}$-norm is another regularization and known as the rotation invariant L_1-norm. This regularization is often applied to tackle and overtake the difficulty of robustness in outliers and the large variations within given data. In this work, ELM with L_{21}-norm algorithm with iterative solution is proposed. For an arbitrary matrix $\beta = \beta_{ij}$, L_{21}-norm is defined as:

$$\|\beta\|_{2,1} = \|\beta\|_{21} = \sum_i (\sum_j \beta_{ij}^2)^{1/2} \quad (4.3)$$

L_{21}-norm is suitable for both supervised and semi-supervised multi-task learning and regression purposes [151, 152]. L_{21}-norm is a sparse algorithm where some of its rows shrink to zero therefore it can jointly select the most useful features and solve the sparse problem with convex solution [153]. As a consequence, this norm is perfect for the noisy classification learning tasks and hence suitable to classify face and object recognitions that include variations like rotation, illumination, pose and complex facial expressions. Analytically, L_{21}-norm can be represented in closed form and it is suitable for yielding a unified extreme learning system [153, 154].

4.2.4 Graph Laplacian Regularization

Graph Laplacian is a mathematical representation that could fully characterize the manifold structure of a graph as a matrix representation. Given a graph $G(V, E)$, with V nodes and E edges, with an adjacency or weight matrix $W = [w_{ij}] \in R^{N\times N}$, the key point in graph embedding representation is the similarity weight matrix W, it helps to store information in a $N\times N$ symmetric matrix W where each entry w_{ij} denotes the edge strength between vertices (or data points) i and j. W is one type of adjacency graph, where $w_{ij} = 1$ if samples x_i and x_j are connected to the same query and $w_{ij} = 0$ otherwise. It can represent w_{ij} entries of the weight matrix W as:

$$w_{ij} = \begin{cases} 1 & if\ x_i = x_j \\ 0 & otherwise \end{cases} \tag{4.4}$$

Depending on the nature of data and the application types, there are different ways for determining the similarity and the weight matrix, one of the common and effective technique is the Radial Base Function (RBF) or heat kernel function [155] as follow:

- For classification

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) & if\ t_i = t_j \\ 0 & otherwise \end{cases} \tag{4.5}$$

- For regression

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right).\exp\left(-\frac{\|t_i - t_j\|^2}{2\sigma^2}\right) \tag{4.6}$$

Where σ is the width parameter and it is set equal to the mean Euclidean distance between the training data.

- For Kernel

The similarity measure $s(.,.)$ that can be exploited to generate the weight matrix W as:

$$s_{ij} = s(h_i, h_j) = \exp\left(-\frac{\|h_i - h_j\|^2}{2\sigma^2}\right) \text{ and } w_{ij} = \begin{cases} s_{ij} & if\ x_i = x_j \\ 0 & otherwise \end{cases} \tag{4.7}$$

The calculation of the weight matrix in the kernel space R^L rather than in the input space R^N has the advantages that the kernel covers all input data and the nonlinear relationships between input data x_i and x_j (which are h_i and h_j) can be used to enhance the performance. The superior performance of the kernel comes with a higher computation cost and memory requirements.

The corresponding graph Laplacian matrix Ł [35] is then defined as $Ł = D - W$, where D is a diagonal matrix with the degree of each vertex, i.e., the column (or row) sums of W,

$$D_{ii} = \sum_j w_{ij} = \text{diag}\left\{\sum_{j=1}^{N} w_{1,j}, \sum_{j=1}^{N} w_{2,j}, \dots, \sum_{j=1}^{N} w_{N,j}\right\} \quad (4.8)$$

Laplacian Ł is a symmetric and positive semi-definite matrix. A simple example can illustrate the above concept as follows:

Suppose we are given five samples $\{x_1, x_2, x_3, x_4, x_5\}$, x_1, x_2 and x_3 are with query 1 while x_4 and x_5 are with query 2. We have the following:

$$W = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}; Ł = D - W$$

$$Ł = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

The resultant Ł matrix can describe the whole situation, the first element of Ł which is 2 means that x_1 is connected to the next points which are referred by -1 value, i.e, points x_2 and x_3 and x_1 is not connected to the other points which are referred by 0 value, i.e, points x_4 and x_5.

The objective function that needs to be minimized is the combination of both the distance between same or near classes by exploiting the output of the activation functions (y_i & y_j) and the embedding adjacent weight matrix (W) where it can be considered as a regularizer that penalizes large distances during the embedding and preserves local information optimally [156].

$$\sum_{ij} \|y_i - y_j\|^2 w_{ij} = \sum_{ij} y_i^T y_i w_{ij} + \sum_{ij} y_j^T y_j w_{ij} - 2\sum_{ij} y_i^T y_j w_{ij}$$

$$= \sum_i y_i^T y_i \sum_j w_{ij} + \sum_j y_j^T y_j \sum_i w_{ij} - 2\sum_{ij} y_i^T y_j w_{ij}$$

$$= \sum_i y_i^T y_i D_{ii} + \sum_j y_j^T y_j D_{jj} - 2\sum_{ij} y_i^T y_j w_{ij} = 2\sum_i y_i^T y_i D_{ii} - 2\sum_{ij} y_i^T y_j w_{ij}$$

$$= 2Tr(Y^T D Y) - 2Tr(Y^T W Y) = 2Tr(Y^T Ł Y) \quad (4.9)$$

Where $Y \in R^{n \times N}$ or $Y \in R^{n \times d}$ (reduced dimension), since the sum of weighted dot products of columns to lines of Y can be written with respect to the matrix-product trace $Tr(.)$, i.e

$$\sum_{ij} {y_i}^T y_i w_{ij} = Tr(Y^T Y D) = Tr(Y^T D Y) \tag{4.10}$$

The derived equivalence of the distance objective function and the trace arises in many methods is trying to preserve manifold distances for subspace learning like the original Graph-Regularized Factorization [156], Laplacian Eigenmaps [157], or Laplacian Regularization [158].

Instead of simply using the graph Laplacian with L_2 regularization on high dimensionality data structure, we propose jointly combine the graph Laplacian with the L_{21} regularization on manifold low dimensionality data.

4.3 Graph Regularized ELM (GELM)

The first representation of the Laplacian regularization with ELM was called pairwise ELM [159] which is claimed to be more consistent with the nature of ranking, the pairwise comparison between the feature input-label relationships instead of the pointwise approximation is considered. The ranking function is searched in pairwise approaches to reveal the similarity preference between two samples. The aim of the pairwise ELM is to minimize the difference effects between similar or nearly samples and to find a relationship that keeping the information from altering after many stages working. Within the ELM framework and traditional least square loss function, the problem can be mathematically written as:

$$\begin{aligned} &\text{minimize: } \tfrac{1}{2}\|\beta\|^2 + \tfrac{1}{2\lambda}\sum_{i=1}^{N}\|\xi_i\|^2 \\ &\text{Subject to: } \left(h\left(x_u^i\right) - h\left(x_v^i\right)\right)\beta = \left(y_u^i - y_v^i\right) - \xi_i^T, \quad i = 1, \dots, N \end{aligned} \tag{4.11}$$

Therefore, with incorporating W within equation 4.11, it can be as:

$$\begin{aligned} &\min \tfrac{1}{2}\|\beta\|^2 + \tfrac{1}{2\lambda}\, W \sum_{i,j=1}^{N}\left\|\left(y_i - y_j\right) - \left(f(x_i) - f\left(x_j\right)\right\|^2 \right. \\ &= \min \tfrac{1}{2}\|\beta\|^2 + \tfrac{1}{2\lambda}\left(\left(y_i - f(x_i)\right)^T Ł \left(y_j - f\left(x_j\right)\right)\right) \\ &= \min \tfrac{1}{2}\beta^T\beta + \tfrac{1}{2\lambda}(T - H(x)\beta)^T Ł \,(T - H(x)\beta) \end{aligned} \tag{4.12}$$

Setting the derivative of equation 4.12 to zero yields:

$$\beta = (H^T Ł H + \lambda I)^{-1} H^T Ł T \tag{4.13}$$

GELM [121, 122] algorithm with graph Laplacian matrices in the ELM space R^L rather in the input space R^n has the advantage that local and nonlinear relationships between the input samples x_i can be formulated and exploited to improve the performance of the learning models and hence further enhancing its classification performance.

Denote $X = [x_1, x_2, \ldots, x_N]$ as the training data matrix where $x_i \in R^n$, define a graph adjacent matrix W, whose element W_{ij} reflects the similarity between x_i and x_j as:

$$w_{ij} = \begin{cases} 1 & x_i \; and \; x_j \; are \; of \; same \; class \\ 0 & otherwise \end{cases} \quad (4.14.a)$$

The graph Laplacian is then constructed through $Ł = D - W$ (4.14.b)

where Ł is the Laplacian matrix, $Ł \in R^{N \times N}$ and D is a diagonal matrix with

$$D_{ii} = \sum_{j=1}^{N} w_{ij} \quad (4.14.c)$$

Since the outputs y_i and y_j should be similar to each other when the inputs x_i and x_j are from the same class, it is necessary to minimize the following objective function which is called graph regularization term:

$$min \sum_{ij} \|y_i - y_j\|_2^2 w_{ij} = Tr(YŁY^T), \text{ where } Y = H.\beta \quad (4.15)$$

By combining the previous L_2 and graph regularization terms with suitable regularization parameters ($\lambda_1$and λ_2) which balance the impacts of different terms, it can generalize the objective function as:

$$\begin{aligned} &min_\beta \|H.\beta - T\|_2^2 + \lambda_1 Tr(YŁY^T) + \lambda_2 \|\beta\|_2^2 \\ &= min_\beta \|H.\beta - T\|_2^2 + \lambda_1 Tr(H\beta Ł\beta^T H^T) + \lambda_2 \|\beta\|_2^2 \end{aligned} \quad (4.16)$$

Derivating equation 4.16 w.r.t β and setting to zero leads to:

$$H^\dagger = (H^T H + \lambda_1 H^T ŁH + \lambda_2 I)^{-1} H^T \text{ and hence } \beta = H^\dagger T \quad (4.17)$$

The local consistency property of similar or neighbor samples that share similar properties can be formulated as adjacent graph regularization term ($H^T ŁH$) and inserted into the main objective function. It can learn similar outputs for samples from the same class.

Algorithm 4.1. Graph Regularized ELM

Input: A training dataset containing N samples each with n features is given as $\{(x_i, t_i) | \ x_i \in R^n, t_i \in R^m, \ i = 1, \cdots, N \}$, i.e, $X \in R^{N \times n}$, and $T \in R^{N \times m}$ where

$T_{ij} = \begin{cases} 1 & for\ vectors\ of\ class_i \ = j \\ -1 & for\ vectors\ of\ class_i \ \neq j \end{cases}$ with any suitable activation function $G(x)$.

Depend on either random or optimized base, assign input weights w_i, bias b_i, hidden nodes number L and the regularization parameters $\lambda_1$and λ_2.

Output: Output weight matrix β

Compute the hidden layer output matrix H using equation 2.6;

Calculate the Laplacian matrix Ł using equations 4.14;

Find $H^{\dagger}$, and the output weight matrix β according to equation 4.17.

4.4 The Proposed L21GELM

To promote feature selection while removing outliers issue, a good way is to consider the sparsity of a decision weight vector β by replacing L_2-norm regularization in the structural risk function with L_{21}-norm. The objective function of the ELM classification problem with graph Laplacian and L_{21}-norm minimization is formulated as:

$$F_{opt} = min_{\beta} \|H.\beta - \ T\|_2^2 + \lambda_1 Tr(Y Ł Y^T) + \lambda_2 \|\beta\|_{21}^2 \tag{4.18}$$

where $\lambda_1$and λ_2 are the penalty terms

$$\text{As } Y = H.\beta \Longrightarrow min_{\beta} \|Y - \ T\|_2^2 + \lambda_1 Tr(Y Ł Y^T) + \lambda_2 \|\beta\|_{21}^2 \tag{4.19}$$

Using Karush-Kuhn-Tucker (K.K.T) principle [160], derive equation 4.19 w.r.t Y yields

$$2(Y - T) + 2\lambda_1 (Y Ł) = 0 \Longrightarrow Y \left(\frac{I}{\lambda_1} + Ł\right) = T \Longrightarrow H.\beta = (c.I + Ł)^{-1} T$$

where the constant $c = 1/\lambda_1$, the inital hidden output weight β_o

$$\beta_o = \beta = H^{\dagger} \ (Ł + c.I)^{-1} T \tag{4.20}$$

where $H^{\dagger} = (H^T H \ )^{-1} H^T$ is Moore-Penrose generalized pseudo inverse.

Now, to solve the equation 4.18, we have

$$F_{opt} = min_{\beta} \|H.\beta - \ T\|_2^2 + \lambda_1 Tr(Y Ł Y^T) + \lambda_2 \|\beta\|_{21}^2$$
$$F_{opt} \ = min_{\beta} \|H.\beta - \ T\|_2^2 + \lambda_1 Tr(H\beta . Ł . \beta^T H^T) + \lambda_2 \|\beta\|_{21}^2 \tag{4.21}$$

Taking $\frac{\partial F_{opt}}{\partial \beta} = 0 \Longrightarrow 2\beta(H^T H) - 2(H^T T) + 2\lambda_1 \beta(H^T Ł H) + 2\lambda_2 \beta D = 0 \Longrightarrow$

$$\beta = (H^T H + \lambda_1 H^T Ł H + \lambda_2 D)^{-1}.H^T T \qquad \text{for } N \geq L \tag{4.22}$$

Based on the inversion basis of [111], equation 4.22 can be formulated to

$$\beta = H^T(HH^T + \lambda_1 Ł + \lambda_2 HDH^T)^{-1}.T \qquad \text{for } L > N \tag{4.23}$$

where $D \in R^{L \times L}$ is a diagonal matrix whose ith diagonal element is

$$D_{ii} = \frac{1}{2\|\beta_i\|_2} \tag{4.24}$$

The algorithm with an iterative solution can be represented as follow:

Algorithm 4.2. Embedded Graph with L21-norm Regularized ELM

Input: A training dataset containing N samples each with n features is given as $\{(x_i, t_i) | \, x_i \in R^n, \, t_i \in R^m, \, i = 1, \cdots, N\}$, i.e , $X \in R^{N \times n}$, and $T \in R^{N \times m}$ where

$T_{ij} = \begin{cases} 1 & for\ vectors\ of\ class_i = j \\ -1 & for\ vectors\ of\ class_i \neq j \end{cases}$ with any suitable activation function $G(x)$.

Depend on DR method with reduced dimension $d < n$, i.e, $X \in R^{N \times d}$

Depend on either random or optimized base [34], assign input weights w_i, bias b_i, hidden nodes number L and the regularization parameters $\lambda_1$and λ_2.

Output: Output weight matrix β

Initial

- Calculate the hidden layer output matrix H and graph Laplacian matrix Ł using equation 2.6 and equations 4.14;
- Based on the dimension of the input data (N) and the hidden nodes number (L) either $N \geq L$, calculate $A = H^T H + \lambda_1 H^T Ł H$
  or $L > N$, calculate $A = HH^T + \lambda_1 Ł$
- Set t=0, $c = {}^1/_{\lambda_1}$ calculate the initial β using equation 4.20 with
  $\beta^t = H^{\dagger}\,(Ł + c.I)^{-1}T$

Repeat

- Update the diagonal matrix D with

$$D^{t+1} = \begin{bmatrix} \frac{1}{2\|\beta_1^t\|_2} & & \\ & \ddots & \\ & & \frac{1}{2\|\beta_L^t\|_2} \end{bmatrix}$$

- Update the output weight matrix β according to equation 4.22 or equation 4.23 as

  $\beta = (A + \lambda_2 D)^{-1}.H^T T$ for $N \geq L$,

  $\beta = H^T(A + \lambda_2 HDH^T)^{-1}.T$ for $L > N$

  $t = t + 1$

After the first iteration, the diagonal D of L_{21}-norm selection stage is introduced and the resultant insignificant distances are removed from the β equation at the second iteration. It can reduce the effect of the outliers becuase the outliers have a large norm distances, so its reciprocal is shrinked to zero.

4.5 Algorithm Headlines

Clearly, L21GELM retains the easy features of ELM, RELM and GELM and offers the following additional worthy merits:

- Generalization performance is improved with nearly immune to the overfitting and noisy issue.
- In order to achieve significant fast and stable classification approach including high dimensionality data of face and object images, dealing with efficient reduced input features is needed, therefore, linear and kernel subspace learnings with graph Laplacian were proposed in the current approach. Various well-known dimensionality reduction methods are employed to reduce the input data that can generate efficient features.
- As in equation 4.22, the learned output weights (β) can regulate different spaces to exploit the advantages of each space, it properly compacts three spaces including: the input space using graph Laplacian with (H^TŁH), feature space (ELM space) using ($H^T H$) and L_{21}-norm sparse space (D).
- The discriminative graph Laplacian with (Ł) is used to avail the consistency in the input databases that can adjust the samples in the same class to similar outputs. The graph Laplacian matrix was calculated based on the training data labels only.
- The ELM space with $H^T H$ maps the data from original space to nonlinear hidden layer space which is useful for smoothed classification. This smoothness property

has been verified to be valuable with local and manifold subspace learning for discriminative tasks.

- The last part of output hidden β equation is the diagonal matrix D that contains diagonal elements with $D_{ii} = 1/2\|\beta_i\|_2$, it is useful for variant and noisy input data. The noisy samples have high $\|\beta_i\|_2$ distance, hence low D, and β depends on the remaining parts (Ł & $H^T H$) which include valuable features while for noise-free data, β generally depends on all parts which make L21GELM robust and less sensitive to noisy data.
- As L21GELM is converged monotonically, it can consider the β result for the large iteration is the global solution to the optimization function.
- The L21GELM substantially merges various ELM algorithms together (RELM, L1ELM, and GELM) with time less than overall algorithm's processing time, it can also achieve some feature selection and sparse advantage with good generalization and performance.

4.6 Experimental Study

Various well-known face with facial expression and object recognition benchmark datasets are examined and state-of-the-art learning approaches are compared with the proposed approach. Consequently, the data representations handled by various feature spaces in training data enhanced significantly the classification efficiency as compared to other related popular algorithms. The outcomes demonstrated that the L21GELM was more general and accurate (more than 3% advance in average accuracy) than other ELM models, also it can deal with large scale data problems with good feature selection and less sensitive to noisy environments. We present experiments with different aspects in order to evaluate the proposed approach in image classification with face and object recognitions. These aspects include analyzing learning capacity, studying the algorithm with employing subspace learnings, performance on pruning irrelevant data or functions, the convergence and sensitivity of the proposed algorithm with various parameters. We have employed two categories of benchmark datasets. The first category comprises face recognition datasets that including ORL, YALE, extended YALE-B, and AR. The second category is object recognition datasets where COIL-20 and USPS are employed, these datasets are briefly described below:

4.6.1 Face recognition datasets

1) The ORL dataset [3]: It contains 400 facial images depicting 40 persons, the images were captured at different times, varying the lighting, facial expressions (open/closed eyes, smiling/not smiling) and facial details (glasses/no glasses). Facial images were taken in a frontal position with a movement tolerance for face rotation up to 20 degrees.

2) Yale Database [161]: consists of 165 gray scale images of 15 individuals. Each subject has 11 images, one per different facial expression or configuration: normal,(center, right and left)-light, w/glasses, w/no glasses, happy, sad, sleepy, surprised, and wink.

3) The Extended YALE-B dataset [162]: includes facial images that depicting 38 persons in 9 poses, under 64 illuminations per individual.

4) The AR dataset [163]: consists of over 4000 facial grayscale images depicting 50 male and 50 female faces. These images have a frontal facial pose, several expressions (anger, smiling and screaming), different illumination conditions (left and/or right light) and with some occlusions (sun glasses and scarf).

Figure 4.2 illustrates four rows with fourteen image samples each from the ORL, Yale, Extended Yale-B and AR databases, respectively.

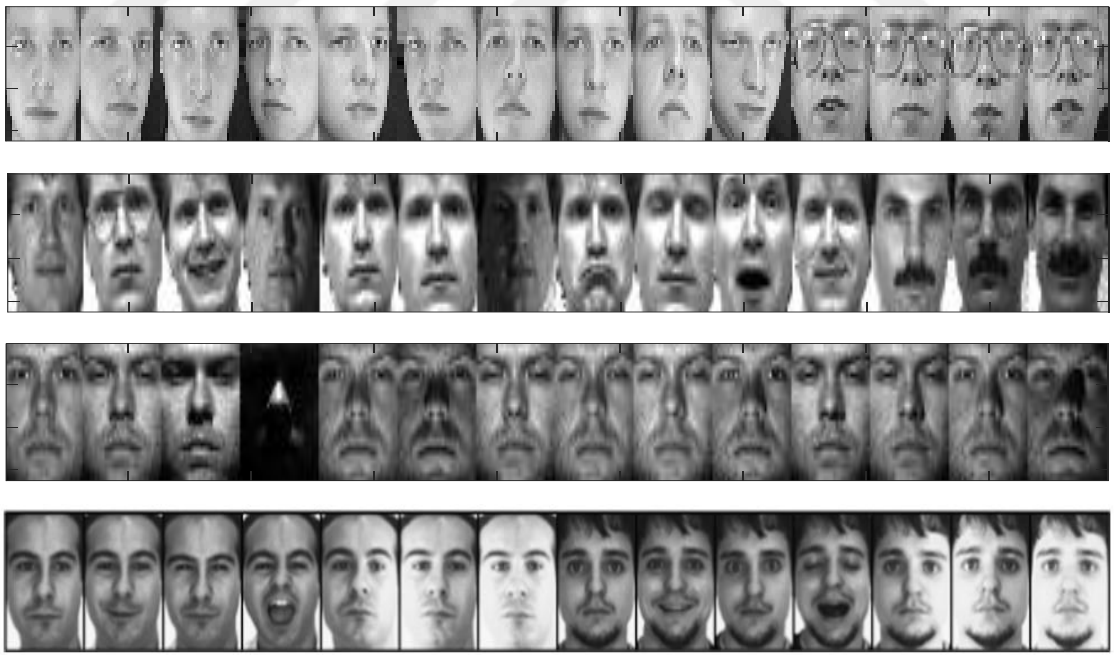


Figure 4.2 ORL, Yale, Extended Yale-B and AR - face recognition dataset samples

[3] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

4.6.2 Object recognition datasets

1) COIL-20 [164]: contains 1440 images of 20 objects, each object has 72 images. The images of each object were taken 5 degrees rotation apart. Each image is represented by a 1024-dimensional vector, with 256 gray levels per pixel.

2) USPS [4]: includes gray-scale images database representing handwritten digits from 0 to 9. The original USPS handwritten digit database contains 9298 images.

Figure 4.3 shows four rows images samples of 20 objects in COIL20 and 10 digits in USPS databases, respectively.

Figure 4.3 An example of images of two-object recognition (COIL20 and USPS) datasets

Table 4.1 presents briefly the information concerning the datasets used in the experiments where the training and testing samples (4 for training and the remaining of 20 for testing) were chosen randomly.

Table 4.1 Details of the face and object datasets

Dataset	Size	# of Features	# of Classes	Dimension	Type
ORL	400	1024	40	32×32	Image, Face
YALE	165	1024	15	32×32	Image, Face
YALE-B	2414	1024	38	32×32	Image, Face
AR	700	300	100	60×43	Image, Face
COIL20	1440	1024	20	32×32	Image, Object
USPS	9298	256	10	16×16	Image, Handwritten

[4] http://www-i6.informatik.rwth-aachen.de/keysers/usps.html, or http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html

To get a better estimation of recognition accuracy, the experimental results obtained by applying the competing algorithms are based on the average of 20 independent trials for regular size datasets, the samples are projected to d number of training samples using different subspace learning approaches.

All simulation programs were realized using MATLAB 8.6 (R2015b) software tool and performed on an (Intel Core i5, 2.4 GHz CPU, 4GB RAM) computer. For all datasets, we chose randomly [2, 3, 4, 5, 10, 20, 30] samples per subject for training and the rest for testing. The variations of the samples per subject, i.e, splitings of training and testing data can provide various circumstances, which are necessary to evaluate the machine learning and computational intelligence methods precisely. It can be considered implicitly as a kind of cross-validation that can give a better estimation of recognition accuracy, Table 4.2 shows the variations in ORL_data size for both training and testing. All images are converted to vector based representation. Our proposed L21GELM algorithm was tested and compared with three ELM well-known approaches: standard ELM (ELM) [19], L2-Regularized ELM (RELM) [5], and Graph ELM (GELM) [121, 122]. It can easily found that the L21GELM achieves consistently better performance (both accuracy and stability) on all the data sets than conventional ELM, RELM and GELM. The main reason is that L21GELM simultaneously takes the classification error term, discriminative graph regularization, and rejecting the undesired samples terms into consideration.

Table 4.2 Variation of some samples per subject for the ORL_database

Samples per subject	**Trained data size**	**Tested data size**
2	80	320
3	120	280
4	160	240
5	200	200

The L21GELM algorithm has four hyper-parameters: number of hidden neurons L, regularization parameters $\lambda_1 \& \lambda_2$ and max iteration t_{max}, all these parameters can be optimized and selected by applying cross validation in the training set. Here, we chose:

- L= [100 200 500 1000 2000 3000], based on the obtained results for almost cases, the highest accuracy was achieved with high L that requires large execution time.

- $\lambda_1 = \lambda_2 = [2^{-10}\, 2^{-9}$ ... $2^{10}]$, regularization parameters that balanced the terms of the objective function, they take wide range. Optimum $\lambda_1 \& \lambda_2$ can be primarily attained

through employing efficient approaches like cross validation with PRESS [28, 113] or L-curve method [110], however these approaches may require large time.

- $t_{max} = 20$, in experiments the learning procedure converged within only 20 iterations which was a fast convergence rate.

In spite of the ability in employing all activation function with L21GELM, the only hidden layer activation function used for all experiments was the sigmoid function. For a fair comparison, same ELM spaces and assumptions are used for all ELM models.

Tables 4.3-4.7 show the mean of the classification accuracy with/without standard deviation for different trains per subject on the ORL, YALE, YALE-B, COIL-20 and USPS datasets respectively using ELM, RELM, GELM, and L21GELM.

Table 4.3 Performance (mean accuracy± standard deviation) % for ELM methods on ORL dataset

ORL	**2 Train**	**3 Train**	**4 Train**	**5 Train**
ELM	78.72±2.61	84.71±2.21	89.87±1.82	93.56±1.43
RELM	82.12±2.34	87.90±1.89	93.23±1.61	95.06±1.50
GELM	86.76±2.30	90.42±1.81	95.01±1.48	96.03±1.12
L21GELM	**88.42±2.28**	**91.81±1.77**	**96.82±1.36**	**97.73±1.10**

Table 4.4 Performance (mean accuracy ± standard deviation) % for ELM methods on YALE dataset

YALE	**2 Train**	**3 Train**	**4 Train**	**5 Train**
ELM	58.12±4.45	67.65±4.42	74.24±3.54	78.78±3.21
RELM	58.86±4.12	68.86±4.12	75.56±3.42	79.83±3.06
GELM	60.21±4.10	70.38±3.89	77.72±3.28	81.87±3.03
L21GELM	**62.86±4.07**	**72.63±3.64**	**79.52±3.20**	**84.24±2.81**

Table 4.5 Performance (mean accuracy ± standard deviation) % for ELM methods on YALE-B dataset

YALE_B	**5 Train**	**10 Train**	**20 Train**	**30 Train**
ELM	75.82±1.95	87.38±1.23	93.57±0.84	95.59±0.74
RELM	76.36±1.63	88.13±1.36	95.02±0.67	97.02±0.56
GELM	79.16±1.64	89.62±1.17	95.93±0.65	97.62±0.59
L21GELM	**81.04±1.43**	**91.24±1.03**	**96.52±0.62**	**98.77±0.47**

Table 4.6 Performance (mean accuracy) % for ELM methods on COIL-20 dataset

COIL-20	2 Train	4 Train	6 Train	8 Train
ELM	70.8	83.81	86.62	88.92
RELM	72.19	84.19	87.02	89.42
GELM	73.52	85.22	87.85	90.31
L21GELM	**75.37**	**87.72**	**89.33**	**92.10**

Table 4.7 Performance (mean accuracy) % for ELM methods on USPS dataset

USPS	3 Train	5 Train	10 Train	15 Train
ELM	71.33	80.74	83.81	87.87
RELM	72.09	82.02	84.74	88.32
GELM	72.30	82.44	84.84	88.82
L21GELM	**74.04**	**84.39**	**86.85**	**89.63**

Figures 4.4-4.8 present the comparisons of the accuracy in histogram shape of the corresponding Tables 4.3-4.7 between ELM, L2-RELM, GELM, and L21GELM on the ORL, YALE, YALE-B, COIL-20 and USPS datasets respectively.

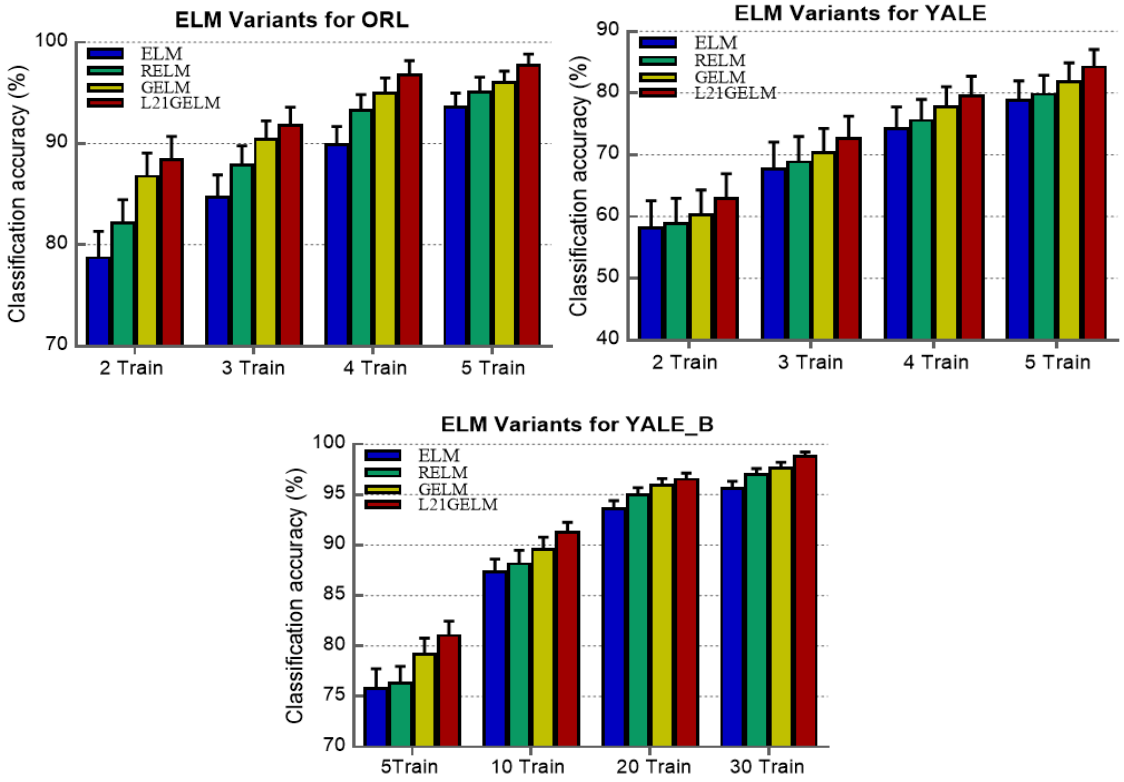


Figure 4.4-4.6 ELM methods comparison testing accuracies for face recognition on datasets ORL, YALE, and YALE-B

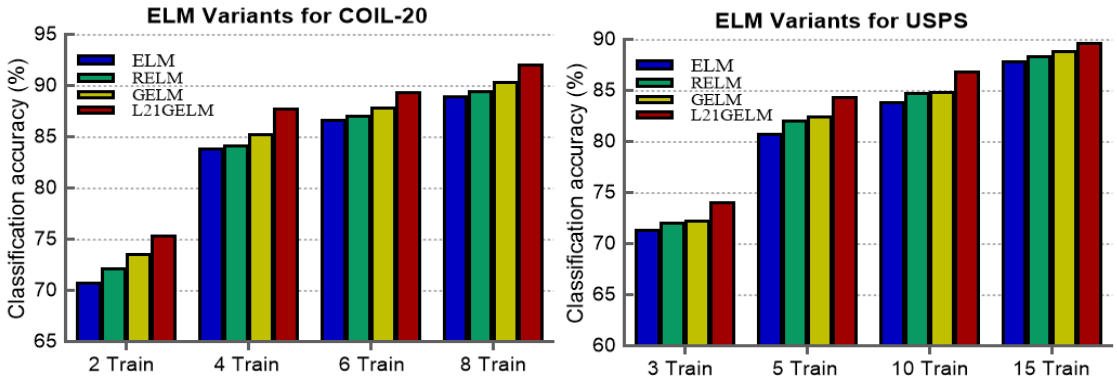
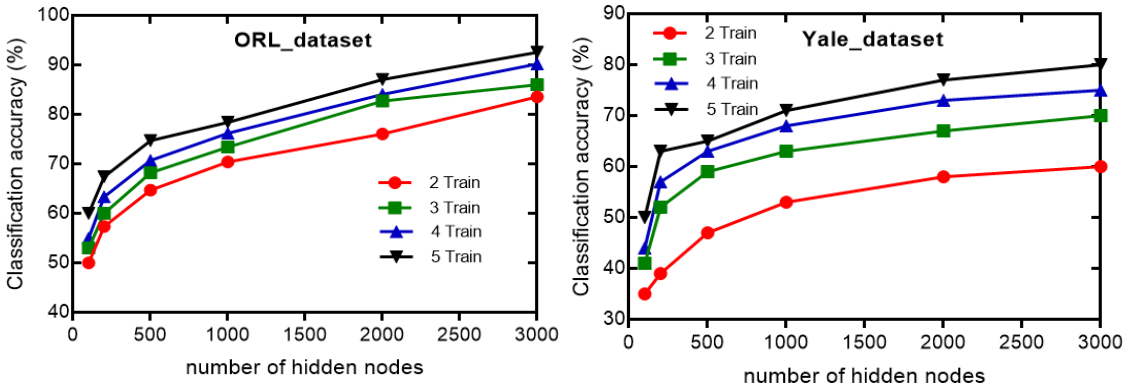
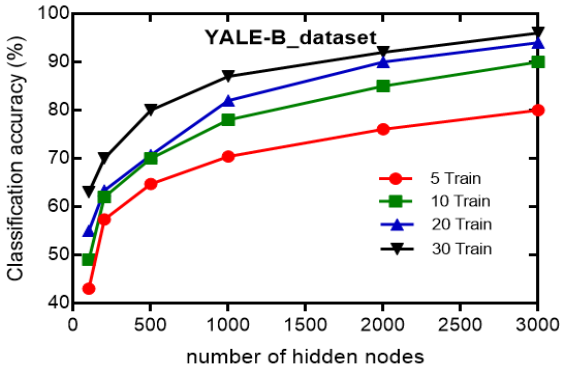


Figure 4.7-4.8 ELM methods comparison testing accuracies for object recognition on dataset COIL-20 and USPS

From the results, it can be seen that the L21GELM is outperformed over all other ELM variants for various train samples. This means that the hidden layer generated with the L21GELM method is more effective due to the including of the most discriminative features. The YALE dataset is more complicated than other face datasets because it has fewer classes with more feature variations (various facial expressions and lighting illuminations) that can increase the difficulty of classification. However, the L21GELM has a distinct accuracy in YALE than other ELM variants due to its immunity against the variants and noisy data.

The suitable selection of the hidden nodes is a key factor in determining the overall ELM network architecture and performance. Figures 4.9-4.11 explain the classification accuracies of L21GELM at different trains per subject against the incremental of hidden nodes [100 200 500 1000 2000 3000] for ORL, YALE, YALE-B datasets respectively. It is clear from the figures that the L21GELM is pretty sensitive, the sharp transitions in all trends below 1000 hidden nodes, to hidden nodes at low number ranges of hidden nodes. The classification accuracies become less sensitive when the numbers of hidden nodes increase.



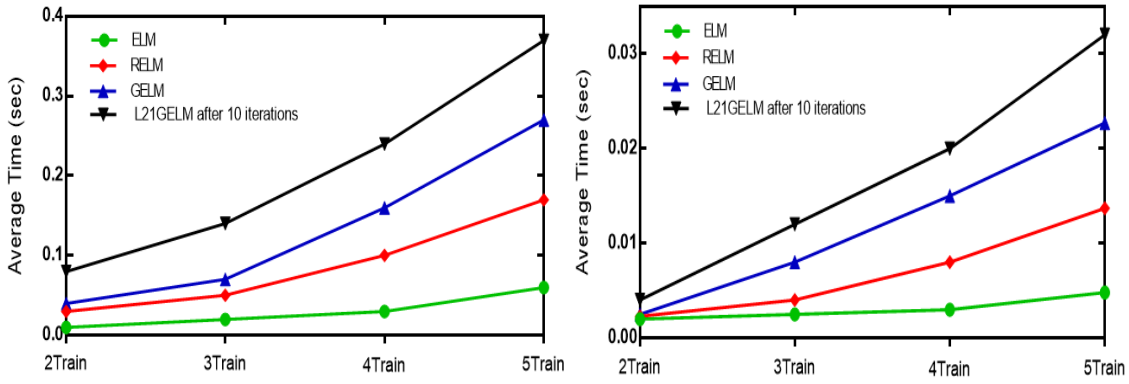


Figures 4.9-4.11 L21GELM performance for different trains per subject against the incremental of hidden nodes on datasets ORL, YALE, and YALE-B

4.6.3 Computing Complexity Analysis

Clearly, the highest computational time operation on computing β is the inversion of $L \times L$ matrices H^TŁH and $H^T H$ which is $O(L^3)$. As in most cases, the number of hidden nodes L can be much smaller than the number of training samples N: $L \ll N$ and thus the computational cost reduces dramatically. Similarly, the RELM, GELM and the proposed L21GELM have similar complexity as conventional ELM, they are based on computing the inverse of $L \times L$ matrices, except L21GELM needs to recursive processing that will increase the time requirement.

Figures 4.12-4.13 reveal a comprehensive comparison on the time elapsed for different training samples of ELM models on the ORL and YALE datasets. In general, the L21GELM algorithm is based on an iterative solution, thereby it has a higher execution time than other ELM methods as shown in the Figures 4.12-4.13. However, L21GELM still has a high convergence training rate compared to other efficient machine learning approaches such as support vector machine (SVM), sparse representation based classification (SRC), and many other.

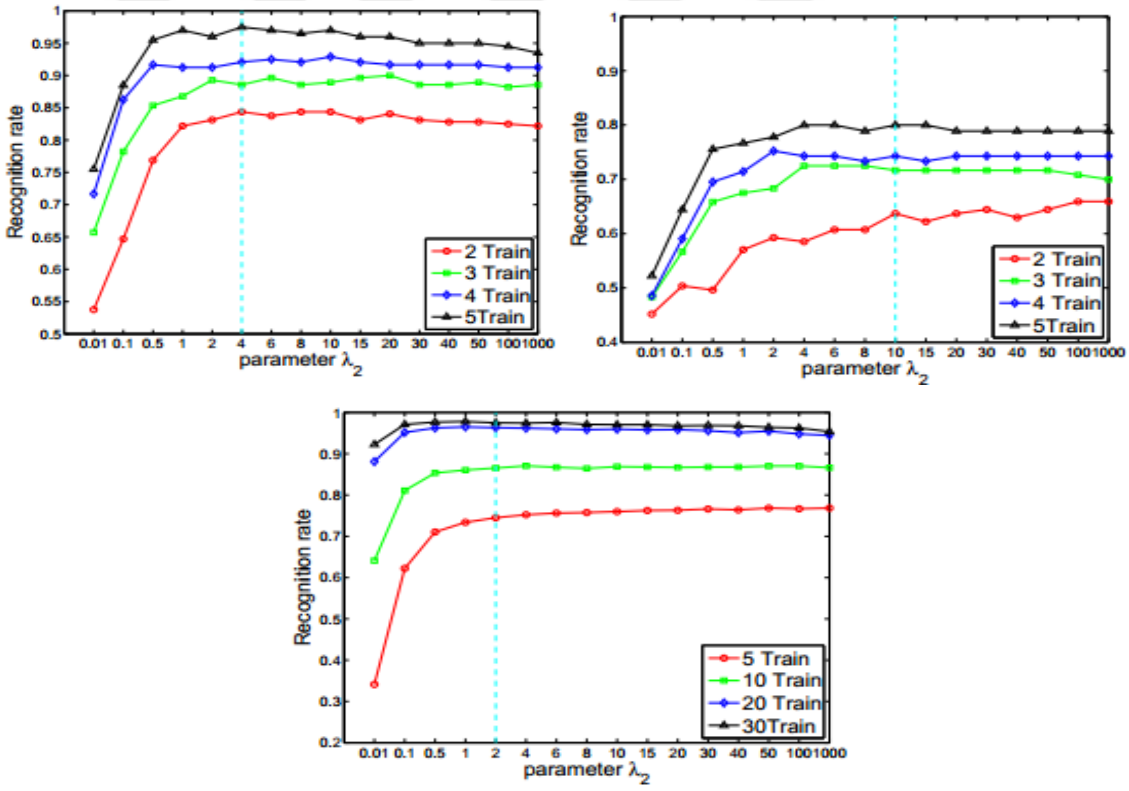


Figures 4.12-4.13 Comparison of time for different training samples of ELM models on the ORL and YALE datasets respectively

4.6.4 Parameters Sensitivity Analysis

Four hyper-parameters are used in L21GELM that proved a main role in the model speed and accuracy: L, t_{max}, $\lambda_1 \& \lambda_2$. The ELM performance (accuracy and stability) is not very sensitive to both of the L, t_{max} as they are large enough, i.e, $L > 1000$, and $t_{max} \geq 10$.

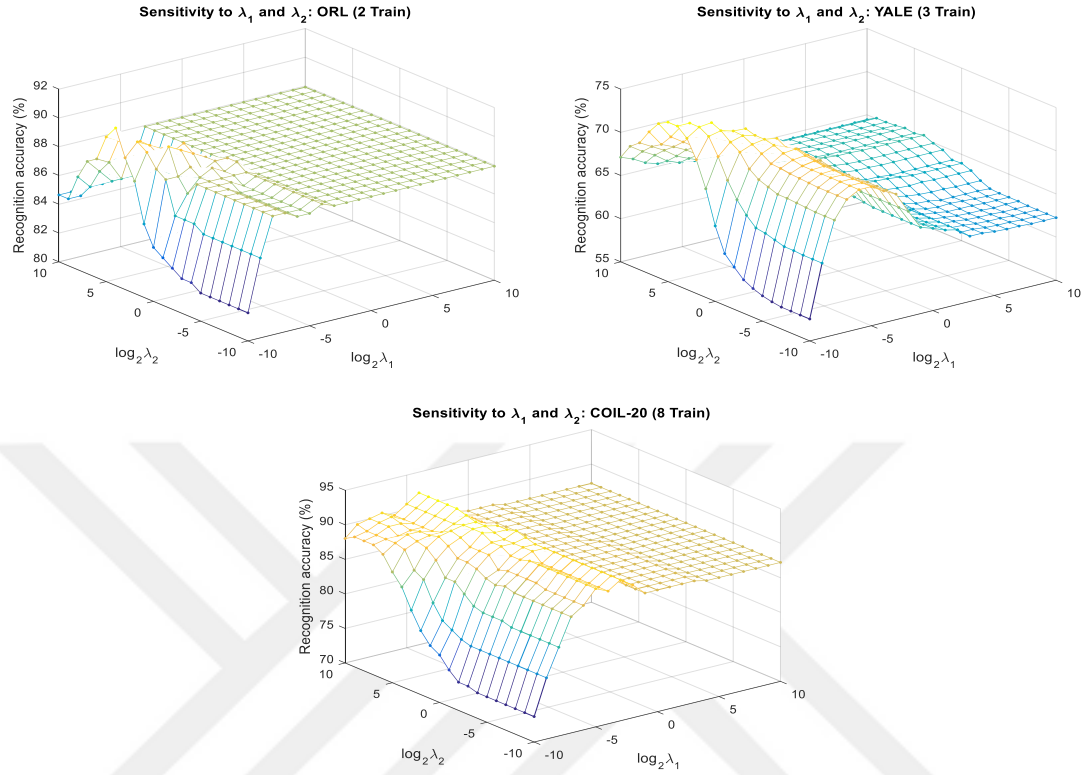
We firstly explain the parameter sensitivity analysis for regularized ELM (RELM) by fixing the number of hidden nodes to large value as 2000 and then analyzing the sensitivity of RELM with λ. Figures 4.14-4.16 show the experimental results of RELM to parameter λ for ORL, Yale and Extended Yale-B database with different number of training samples per subject. The vertical line shown in cyan in each figure allocates the best value of RELM. Experimental results have shown that RELM performs well in ELM training (the input weights and the bias of hidden layer are randomly generated).



Figures 4.14-4.16 RELM recognition rate to the variations of the λ parameter for ORL, YALE, and YALE-B respectively

Measuring the sensitivity of the L21GELM performance with respect to the combination of regularization parameters $\lambda_1 \& \lambda_2$ needs firstly fixing other hyper-parameters L, t_{max}. Figures 4.17-4.19 show the accuracy for different trains and various datasets as a function

to the combination of $\lambda_1 \& \lambda_2$ in the $\lambda_1 \& \lambda_2 = [2^{-10}\, 2^{-9}\, \ldots\, 2^{10}]$ range with $L = 2000$, $t_{max} = 10$.



Figures 4.17-4.19 $\lambda_1 \& \lambda_2$ parameters sensitivity analysis of L21GELM on different trains and various database

The accuracies in the grid graphs are well tuned and flat in wide surface area under the range of $\lambda_1 \& \lambda_2$ that ensuring stable performance, the L21GELM is quite sensitive on some parts of regularization parameters. Moreover, in spite of analyzing different trains and datasets, the best performance is bounded with almost unique range of $\lambda_1 \& \lambda_2$. In order to confirm the optimum generalization performance with minimum execution time, the four hyper-parameters (L, λ_1, λ_2, t_{max}) firstly need to be optimized and smoothly tuned by adopting the cross validation in the training part.

The L21GELM is an iterative approach; it reaches the desired solution with fast convergence rate. The testing accuracies of the L21GELM algorithm for the AR and USPS datasets, as illustrated in Figure 4.20, are better than those for other ELM methods and their significant values are distinct after 10 iterations. The starting accuracy at initial iteration $t = 1$ is comparable to that of the baseline ELM and after several iterations it increases quickly. The monotonic increase of the convergence curve proves the optimality

solution of the β where the output weights of the irrelevant hidden neurons always shrink to zero.

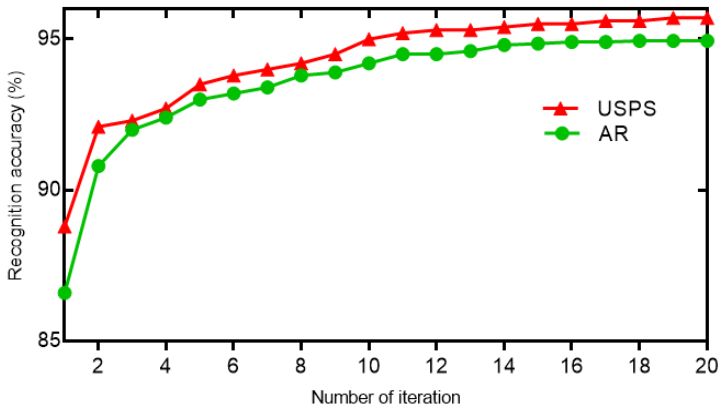


Figure 4.20 Testing accuracies of AR and USPS datasets with t-iterations.

β is a global optimum solution to the problem. Moreover, if the proposed Algorithm 4.2 converges, β^t and hence D^t will meet the optimal condition, thus β^t is the global optimum. So the problem of proving that the solution of algorithm is global optimal can be transformed into proving the convergence of the proposed algorithm. Fortunately, the monotonic decrease of the objective value according to the updating scheme is guaranteed by the proof in [165]. Since the objective value is monotonic increase or decrease, it should be convergent. On the basis of the above discussion, the global optimality of the solution induced by the algorithm has been guaranteed.

4.6.5 L21GELM with Local and Manifold Subspace Learning

Generally, the number of hidden nodes (L) of the baseline ELM without regularization is small because it works on the R^n input space only and it uses the number of hidden nodes to control the model complexity. GELM and L21GELM are working on both R^n input and R^L feature spaces, as almost ($L > n$), it needs to enlarge hidden nodes number for providing global solution and hence, enhancing performance. Employing subspace learning (SL), input data with R^n space can be reduced to R^d space ($d << n$) by yielding only discriminative input features for improving the accuracy and minimizing the operation time through decreasing the requirement for large L. In this work, L21GELM exploits different feature spaces determined by applying many non-supervised and supervised approaches with linear and non-linear feature mapping as PCA, LDA, KPCA, KDA, LPP, NPE, and LGE.

Tables 4.8-4.10 provide a comparison that reveals L21GELM classification accuracies for subspace learning DR methods (PCA, LDA, KPCA, KDA, LPP, NPE, and LGE) with

$L = 500$ on ORL, YALE, and YALE-B datasets respectively. Since all methods are of comparable standard deviations, the accuracies are written only as:

Table 4.8 L21GELM accuracy of different training sampling on ORL dataset for $L = 500$ hidden nodes

Train	(PCA)	(LDA)	(KPCA)	(KDA)	(LPP)	(NPE)	(LGE)
2	79.41	**84.46**	43.93	53.91	77.75	78.18	76.63
3	85.11	**89.64**	48.54	62.61	84.96	83.85	83.18
4	89.25	**94.85**	45.5	71.83	89.68	85.41	87.79
5	91.0	**96.02**	47.05	77.2	91.5	86.3	91.0

Table 4.9 Accuracy of different training sampling on YALE dataset for $L = 500$ hidden nodes

Train	(PCA)	(LDA)	(KPCA)	(KDA)	(LPP)	(NPE)	(LGE)
2	54.66	**58**	10.0	10.44	56.51	56.51	57.25
3	64.58	**68.41**	10.41	12.91	68.41	68.25	68.58
4	71.90	**76.95**	10.19	13.90	74.76	76.19	76.0
5	75.88	**82.44**	10.33	14.55	79.11	80.22	79.22

Table 4.10 Accuracy of different training sampling on YALE-B dataset for $L = 500$ hidden nodes

Train	(PCA)	(LDA)	(KPCA)	(KDA)	(LPP)	(NPE)	(LGE)
5	48.06	**73.72**	3.7	19.39	70.5	67.02	63.91
10	60.6	**84.46**	5.4	32.25	72.23	70.43	70.39
20	72.92	**85.19**	8.9	44.54	74.29	71.58	74.66
30	75.63	**90.45**	12.46	57.08	82.09	73.09	81.30

It is clear from the above tables that utilization of the data representations in different feature spaces in training data can impact the ELM performance (both accuracy and stability). In addition, the best-reduced feature space used with the L21GELM is that obtained from LDA. This is because of the distribution of the data in the face and object recognition images are normally distributed that meet the LDA properties. In addition, LDA is a supervised technique that analytically depends on the labels information availability and determines the best separating hyperplane between the classes by minimizing within-classes and maximizing between-classes variances. Furthermore, it appears from the theory and results of L21GELM (LDA) that it is close to the works [121, 166] where they formulate the embedded graph Laplacian based on both within-class

(intrinsic graph) and between-class (penalty graph) with the advantage of L21GELM regarding its suitability to noisy data.

We follow approximate kernel learning approaches, which are suited to large-scale learning problems for fast calculations. The Kernel based methods (KPCA and KDA) convert the input data to kernel feature space then the transformed data is inputted to ELM that has nonlinear activation functions, the consequence nonlinearity conversions alter the data to non-suitable features for classification, results in both KPCA and KDA give low accuracies. Under such conditions, non-linear classifiers give very poor accuracy results while linear classifiers give better results.

4.6.6 L21GELM with Noisy Environment

For further showing the efficiency of proposed L21GELM, it is necessary to measure the performance, examine the immunity and robustness of L21GELM against noisy data (same previous datasets with adding a certain noise level). The aim of this experiment is to investigate the ability of L21GELM as compared to other ELM models to classify well under noisy circumstances. Since the L21GELM is proposed for classification purposes that surely have integer labels, it is difficult or impossible to add real noise values or convert the classes to other values. Therefore, the only potential way is to add the noise to the attribute data (training and/or testing features). The variety of the datasets prevent us from adding randomly distributed noise to attributes, so uniformly distributed noise is added depend on the standard deviation (σ) of the input data and the noise level (k) as in equation 4.25:

$$feature_{noisy} = feature_{free} + k \times \sigma \times random(size(feature_{free})) \quad (4.25)$$

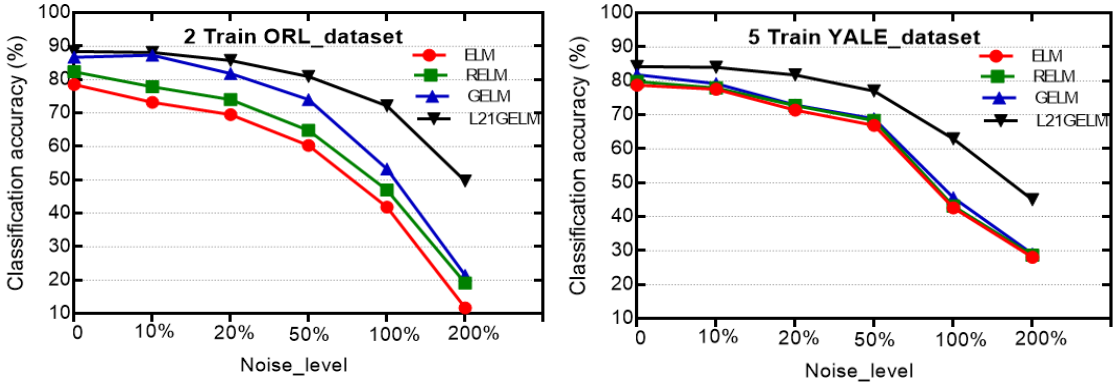
Tables 4.11-4.12 and corresponding Figures 4.21-4.22 show the classification accuracy of randomly choice on ORL (2 Train) and YALE (5 Train) datasets with noise level equal to [0 10 20 50 100 200] % of the standard deviation of the input test data. It is clear that all methods accuracies decrease dramatically with increasing the noise level. However, our L21GELM is less sensitive to the noise than others due to the jointly considering of both L21-norm and different feature spaces including low-rank which has been proved to be robust to outliers and noise [153].

Table 4.11 Testing accuracy for different noise levels on ORL (2 Train) dataset

ORL	Noise level (k%) /Testing accuracy (%)					
Algorithm	**0**	**10%**	**20%**	**50%**	**100%**	**200%**
ELM	78.5	73.2	69.5	60.3	41.8	11.7
RELM	82.3	77.8	74.0	64.8	46.9	19.1
GELM	86.7	87.3	81.8	74.0	53.2	21.4
L21GELM	**88.4**	**88.1**	**85.7**	**80.9**	**72.1**	**49.6**

Table 4.12 Testing accuracy for different noise levels on YALE (5 Train) dataset

YALE	Noise level (k%) /Testing accuracy (%)					
Algorithm	**0**	**10%**	**20%**	**50%**	**100%**	**200%**
ELM	78.7	77.5	71.4	66.9	42.6	28
RELM	79.8	77.9	72.7	68.3	43.1	28.6
GELM	81.8	79.2	72.9	68.8	45.6	29
L21GELM	**84.2**	**84.0**	**81.7**	**77.0**	**62.9**	**45**



Figures 4.21-4.22 Testing accuracy for different noise levels on ORL (2 Train) and YALE (5 Train) datasets

4.7 Comparison with other Classifiers

Finally, the L21GELM approach is compared with the state of the art classifier algorithms which are widely used in the face recognition: Nearest Neighbor Classifier (k-NN), Linear Regression Classifier (LRC) [118], Support Vector Machine (SVM) [117], Sparse Representation-based Classification (SRC) [116], Collaborative Representation-based Regularized Least Square (CRC_RLS) [119], GELM [122]. As in the Table 4.13, for the AR dataset and under different dimensions the classification accuracies can be seen, the L21GELM achieves satisfactory performance in all dimension cases, demonstrating the superiority and effectiveness of the proposed paradigm over other methods even for dataset under different conditions. Furthermore, L21GELM was proved to need less execution time among other state of the art algorithms but larger than well-known ELM approaches.

Table 4.13 Different state of the art classifier accuracies (%) on AR face recognition dataset for different dimension

AR	**Testing accuracy (%)**		
Algorithm	**Dim d=50**	**Dim d=150**	**Dim d=n=300**
1NN	67.8	70.2	71.3
LRC	71.1	74.6	76.0
SVM	68.3	74.7	75.4
SRC	81.9	90.1	**93.3**
CRC_RLS	80.7	**91.0**	**93.7**
GELM	**84.0**	**91.4**	**93.8**
L21GELM	**85.2**	**92.3**	**94.4**

4.8 Unified ELM with L21GELM Classifier

In addition to better performance of L21GELM on different evaluation criteria, including efficiency, accuracy, and consumption time. L21GELM can be considered as a unified ELM model that can achieve different ELM approaches as in equation 4.26 where for $N \geq L$:

$$\beta = (H^T H + \lambda_1 H^T Ł H + \lambda_2 D)^{-1}. H^T T \tag{4.26}$$

it can acheive the following models:

- ELM, if $\lambda_1 = \lambda_2 = 0,\ \beta = (H^T H)^{-1}. H^T T$
- RELM, if $\lambda_1 = 0, D = I, \beta = (H^T H + \lambda_2 I)^{-1}. H^T T$
- GELM, if $D = I, \beta = (H^T H + \lambda_1 H^T Ł H + \lambda_2 I)^{-1}. H^T T$
- L21GELM, $\beta = (H^T H + \lambda_1 H^T Ł H + \lambda_2 D)^{-1}. H^T T$

CHAPTER 5

EXTREME LEARNING MACHINE WITH ENSEMBLE

5.1 Introduction

In the previous chapters, the properties of ELM has been extensively reviewed. The ELM like other machine learning methods may suffer from the overfitting and underfitting issues. The RELM and GELM have been proposed to deal with the mentioned issues in ELM.

In the ELM method, the user can set two parameters which are the number of hidden layer neurons (L) and the hidden layer input weights (w). Consequently, the improper initialization of (L) and (w) may impair the performance of the ELM model.

The random initialization of parameters especially weights and biases perhaps increases the complexity of hidden layer and cause ill-condition. It can affect model generalization ability and may weaken robustness of ELM to encounter variations in the input data [167]. The ensemble and regularization algorithms can be presented with ELM to reduce the randomness. Both network ensemble and regularization can be performed with ELM and it's variants not to alter the network topology but to provide another solution that can tackle efficiently the parameters uncertainty problems of ELM architecture and to improve the performance and networks generalization ability. Moreover, transforming to non-linear mapping with different activation functions can reduce the convergence ability of the algorithms [168, 169], it can improve this convergence by utilizing the ensemble, regularization, or may kernelization principle.

The L2 (Ridge regression) Regularized ELM version (RELM) is a well-known variant of ELM. Experimentally, RELM outperforms the standard ELM because it is more stable, robust against variations, and has good generalization performance, thus avoid model overfitting and improve overall prediction accuracy. RELM has two vital parameters that

effect the performance which are the hidden nodes number (L) and the regularization parameter (λ). L doesn't need to be tuned as long as it is large enough while λ is variable and needs to be optimized.

Although optimizing the regularization parameter of the RELM using Leave-One-Out Cross Validation (LOO-CV) method was reported to be efficient, it is a costly approach.

In the current work, an ensemble ELM approach based on homogeneous RELM base classifiers and weighted majority voting for making final decision is investigated. The base learners generates different decision boundaries and different errors. Moreover, the weights are assigned according to their expected minimum errors on the training data. Combining multiple RELMs in an ensemble model with suitable weights can greatly improve the test accuracy rates. It can also overcome some ELM limits and difficulties that significantly restrict the reliable performance and hence provide some ELM solution keys too.

To perform well-organized version of RELM that can search and evaluate the optimal network parameters and keep the speed advantage of the ELM, PREdiction Sum of Squares (PRESS) formula and Singular Value Decomposition (SVD) are combined together as computational approaches. Efficient LOO-CV with PRESS formula and SVD factorization is proposed. SVD is utilized implicitly within the RELM approach to retain the fast ELM learning advantage and to reduce dramatically the complexity of the LOO-CV.

The proposed algorithm is examined with various datasets types, the experimental results have demonstrated that this approach has better performance on several evaluation criteria, including accuracy, stability, and speedup. It can enhance the performance by increasing the accuracy probably more than 3% and minimizing the required operations through speed up the RELM algorithm with factor (2.8) for the average of 20 trials and factor (6.8) for the average of 50 trials.

5.2 Ensemble Classifier Schemes

Typically, there are differences among the ensemble systems in terms of the selection of training data for individual classifiers, the certain protocol employed to generate ensemble members, and/or the combination rule or standard to obtain the ensemble decision.

In most cases, ensemble members are used in one of two schemes: classifier selection or classifier fusion. In the classifier selection, each individual classifier is trained in some local neighborhood of the total feature space, and then the final decision is chosen depending on the highest contribution weight to the final decision. In the classifier fusion, all available classifiers are independently trained at the same time over the entire feature space, and then combined to obtain a composite classifier which has lower variance (and consequently lower error). Some examples of these approaches are available in literature including bagging [170], random forests [171], arc-x4 [172], and boosting/AdaBoost [173]. Different ways for combining the individual classifiers can be used, they almost based on the labels only, or on class-specific continuous valued outputs, for which classifier outputs are initially normalized to the interval of $(0,\ 1)$ and considered as the support provided by the classifier to each class. The algebraic combination rules such as simple or weighted majority voting, maximum/minimum/sum/product, or other combinations class-specific outputs can be achieved by the above interpretation in addition to the Dempster–Shafer-based classifier fusion, or decision templates. However, for such classifiers, simple or weighted majority voting followed at a distant second by the Borda count are the most commonly applied combination rules. On the other side, there are several classifiers, that provide continuous valued class-specific outputs such as multilayer perceptron or (naive) Bayes classifier; in which, each class will receive support by the classifier. In such classifiers, a wide range of combiners can be used such as voting-based approaches, arithmetic (sum, product, mean, etc.) combiners in addition to more sophisticated decision templates. Some combiners are simple and available to be used promptly when the training is finished, whereas others are more complex and may need an extra training steps (for example, those used with hierarchical mixture of experts and stacked generalization). Several approaches will be briefly discussed in the next sections.

5.3 Ensemble Combiner Schemes [170]

The last step in any ensemble-based system is the mechanism used to combine the individual classifiers. The type of classifiers is very important to select the strategy used in this step.

The decision of the kth classifier is defined as $f_k \in \{0, 1\}$, $k = 1, \dots, K$ where K is the number of the base classifiers. Many ways of combining ensemble members are listed below as:

5.3.1 Algebraic combiners

Algebraic combiners are non-trainable combiners. In those combiners, an algebraic expression such as sum, product, mean, minimum, maximum is used to combine the continuous valued outputs from classifiers. In each case, the largest support is received by the final ensemble decision (class j) after an algebraic expression is applied to individual supports $\mu_j(x)$ obtained by each class. Specifically $h_{final}(x) = arg\ max_j\ \mu_j(x)$ where the final class supports are determine as follows:

- Mean rule: $\mu_j(x) = \frac{1}{K}\sum_{k=1}^{K} f_k(x)$
- Harmonic mean $\mu_j(x) = \frac{K}{\sum_{k=1}^{K}\frac{1}{f_k(x)}}$
- Sum rule: $\mu_j(x) = \sum_{k=1}^{K} f_k(x)$
- Weighted sum rule : $\mu_j(x) = \sum_{k=1}^{K} \alpha_k f_k(x)$ where (α_k is the weight assigned to the kth classifier f_k according to some measure of performance), if α_k are not normalized, i.e, $\alpha_k \notin \{0, 1\}$, the weighted sum rule will be: $\mu_j(x) = \frac{\sum_{k=1}^{K} \alpha_k f_k(x)}{\sum_{k=1}^{K} \alpha_k}$
- Product rule: $\mu_j(x) = (\prod_{k=1}^{K} f_k(x))^{1/K}$
- Maximum rule: $\mu_j(x) = max_{k=1,\dots,K}\{f_k(x)\}$
- Minimum rule: $\mu_j(x) = min_{k=1,\dots,K}\{f_k(x)\}$
- Median rule: $\mu_j(x) = med_{k=1,\dots,K}\{f_k(x)\}$
- Generalized mean rule: $\mu_{j,\delta}(x) = (\frac{1}{K}\sum_{k=1}^{K} f_k(x)^{\delta})^{1/\delta}$

  Where, $\delta \to -\infty$ (Minimum rule), $\delta \to \infty$ (Maximum rule), $\delta = 0$ (Geometric mean rule), or $\delta = 1$ (Mean rule).

5.3.2 Voting based methods

Voting based methods operate on labels only. In such methods, $f_k(x)$ is either 1 or 0 depending on whether classifier k chooses j among C classes, or not, respectively. The ensemble then chooses class J that receives the largest total vote:

- Majority (plurality) voting

  $\sum_{k=1}^{K} f_k(x) = max_{j=1,\dots,C}\{\sum_{k=1}^{K} f_k(x)\}$

When the outputs of the classifier are independent, the performance improvement always being achieved by the majority voting combination.

- Weighted majority voting

  $\sum_{k=1}^{K} \alpha_k f_k(x) = max_{j=1,\dots,C}\{\sum_{k=1}^{K} \alpha_k f_k(x)\}$, where ($\alpha_k$ is the classifier weight).

5.3.3 Other combination rules

More sophisticated combiners than those mentioned above are also available including Borda count [174], behavior knowledge space [174], decision templates [175, 176], and Dempster Schafer rule [177]. In Borda count method, the rankings of the class supports are usually taken into consideration whereas in the behavior knowledge space combiner, the most true classes for every possible class combinations provided by the classifiers are listed in a lookup table. In decision templates, the current decision profile of the unknown instance and the average decision profiles of instances from each class are computed and compared for similarity, and lastly, the Dempster Schafer rule which computes the plausibility based belief measures for each class. However, simple rules such as sum or weighted majority voting have almost demonstrated better empirical work than others [178]. More details regarding the all available combination rules in the literature can be found in [179].

5.4 Ensemble Learning

Ensemble learning is a technique that combines multiple classifiers or diverse models where a finite number of learners is trained for the same task. The ensemble learning preliminaries can be summarized as:

- Learn multiple alternative definitions of a concept using different training data or different learning algorithms.
- Then, decisions obtained from the multiple definitions are combined, by using for example weighted voting.

It can enhance the performance, lowered the instability and the variance, and also useful to deal with lack of adequate data or large and complex databases [180–183]. If proper diversity is achieved, each classifier makes a diverse error; so, involving strategical combination of this ensemble can minimize the overall error. This concept is graphically clarified in Figure 5.1 (a, b, c), it was adopted in V-ELM [184] where several K independent networks are trained individually by the ELM that uses same number of hidden nodes as well as similar activation functions. Each ELM is randomly initialized independently and all these individual ELMs are trained with the same learning parameters and dataset. The final class label is then determined by one method of ensemble combiner schemes that mentioned previously in section 5.3 for example in V-ELM, a majority voting has been used on all the results gained by these K-independent ELMs.

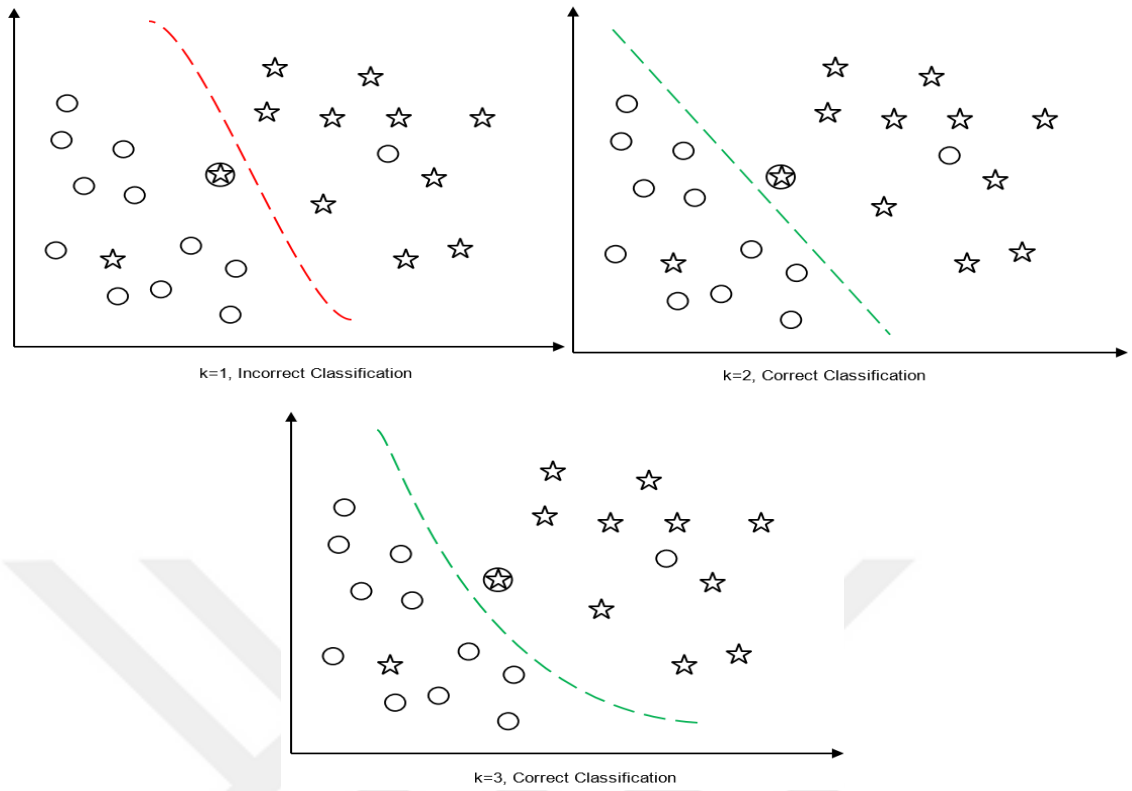


Figure 5.1 (a, b, c) ELM classifiers using three different random parameters

As can be seen in the ensemble system shown in Figure 5.2, multiple *independent* and *diverse* decisions are combined which are more precise compared to random guessing; random errors eliminate each other and the correct decisions are reinforced.

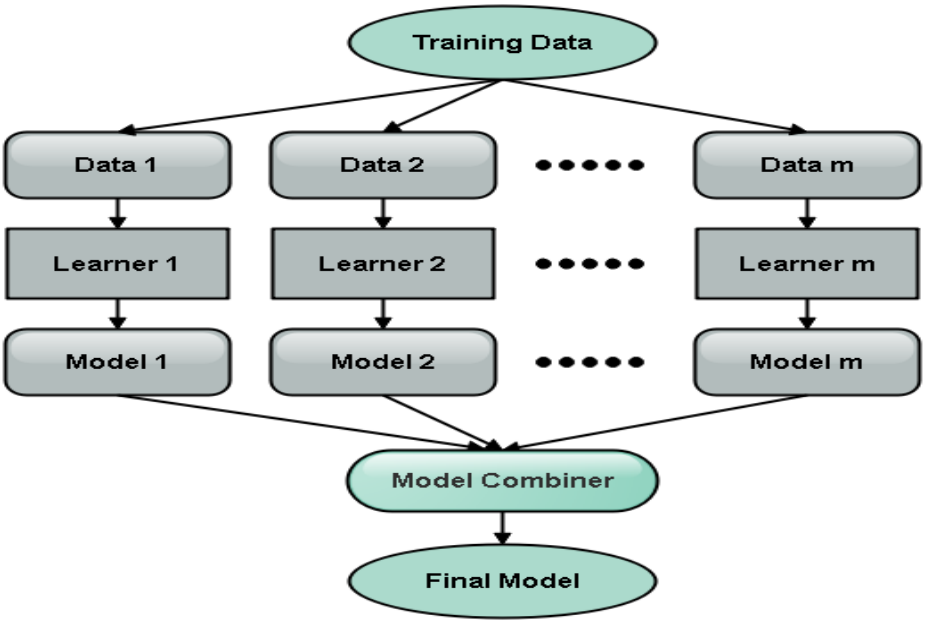


Figure 5.2 Ensemble system of m-multiple classifiers/models

The ensemble system has two main categories:

- Homogeneous base classifier: a single learning algorithm is used; however, various training data is manipulated to enable it learn multiple data models.
    - ✓ Training data is not equal (i.e. data1 ≠ data2 ≠ … ≠ data m).
    - ✓ Learning algorithms are similar (i.e. learner1 = learner2 = … = learner m).
- Heterogeneous base classifiers: Use multiple arbitrary learning models that manipulate same and almost diverse training data to make it learn multiple models.
    - ✓ Training data is not equal (i.e. data1 ≠ data2 ≠ … ≠ data m).
    - ✓ Learning algorithms are not similar (i.e. learner1 ≠ learner2 ≠… ≠ learner m).

There are different methods for changing training data, the well-known and most common methods are:

- ➢ Bagging: Resample training data with repeatedly and randomly replacement.
- ➢ Boosting: Reweight training data, that is to say, a new hypothesis is learned at each iteration and then the examples are reweighted to make the system focusing on examples until the last learned classifier became wrong.
- ➢ DECORATE: Add additional artificial training data to increase diversity in the obtaining ensemble.

5.5 Review of ELM Ensemble

Many ELM ensemble techniques have been proposed, Lan, Y., et al. [185] reported a multiple online sequential ELM ensemble (EOS-ELM). This ensemble picks the mean value of the multiple test outputs as the predictive output.Voting based ELM (V-ELM) [184] was proposed as an ensemble algorithm to reduce the number of misclassified samples and to lower the variance amongst diverse realizations. Multiple trainings of independent ELMs are usually performed in V-ELM instead of a single ELM training. Then, the final decision is made based on the majority voting algorithm. Instead of simple voting that assumes all individual classifiers contribute equally to the decision ensemble, Cao, J., et al. [186] adopted weighted voting method in ELM (WV-ELM), the weights assignment of WV-ELM is based on a minimum square error (MSE) metric to optimize the linear combination of multiple ELMs.

The assumption of ensemble several independent models may be better than ensemble all of them, Liu, Y., et al. [187] proposed selective ensemble algorithm (ROS-ELM) to improve the robustness and stability of OS-ELM, they adopt particle swarm optimization

(PSO) method to select individual learner to form the ensemble. Liu, N., et al. [188] used the genetic algorithm (GA) as an evolutionary algorithm (EV-ELM) for optimizing these weights by separating them to distinguish the importance of each individual ELM classifier in decision making.

Instead of using all component classifiers, ensemble pruning techniques such as Order based, Clustering based and Optimization based [174] are employed to select the accurate and diverse classifiers subsets that may give equal or better performance. Two robust algorithms of ensemble extreme learning machines E2LMs (Bagging-based and AdaBoost-based ELMs) [183] were proposed for the classification task of the high-dimensional data such as hyperspectral images. Van Heeswijk, M., et al. [33] built GPU-accelerated ELM ensemble with structure selection model to enhance the overall ensemble ELM predication error for large scale regression tasks. It first trains multiple ELMs independently through an efficient leave-one-out cross validation (LOO-CV) method to compute the ensemble weights where the LOO error criteria is employed to calibrate these weights, and then combines the individual ELMs models through weighted voting.

Escandell-montero, P., et al. [189] suggested an ensemble ELM using a regularized committee for regression problems. A regularized version of least squares fitting is employed to calculate the ELM model parameter while elastic net regularization is used to select the ensemble committee members.

Zhang, P., et al. [190] exploited Robust AdaBoost.RT based Ensemble ELM (RAE-ELM) as a quick learner with high regression performance, which makes it a good candidate of "weak" learners. Therefore, RAE-ELM approach could output the final hypotheses in optimally weighted ensemble of weak learners. To meet the challenge of the big data, [180] introduced a Distributed ELM (DELM) as an optimized version of ELM to deal with large matrix operations over large datasets, then developed an algorithm called Weighted Ensemble Classifier based on Distributed ELM (WE-DELM) for efficiently classifying uncertain streaming data with drift concept. The weights of the base classifiers are updated dynamically according to classification results. Also [181] suggested a parallelized ELM ensemble that based on the Min–Max Modular network (M3-network) to increase its capability in handling large-scale classification problems. The proposed M3-ELM first decomposes classification problems into smaller subproblems which are easier to be trained by ELM and then ensembles these ELMs with

the M3-network that can leverage the network to derive the prediction instead of combining the component outputs of linearly.

Cao, J., et al. [191] recommended two weights optimization based ensemble methods (CSSV-ELM and SpaCSSV-ELM) under the framework of Class-Specific Soft Voting Scheme for multiple ELMs. The two types of CSSV are based on accuracy and sparsity aspects to regulate the weights for multiple ensemble models. CSSV can lower memory requirement and speed up the classification process. Cao, J., et al. [116] exploited speed advantage of ELM and the robust behavior of Sparse Representation Classifier (SRC) against noisy data to adapt a hybrid ensemble method combining these models. Consequently, it can produce better classification performance with a lower computational complexity.

It is worth noting that involving the Evolutionary Algorithm EA [102, 188] or the Sparse Representation SRC [85, 96, 98] as auxiliary tools to the ELM will enhance the classification rates, but the salient advanced property of the ELM which is the extreme speed may be lost.

5.6 The Proposed Weighted Majority Voting

In many applications, not all the classifiers contribute equally to the decision making. The overall performance of the ensemble system can be improved by weighing the decisions prior to combination. Moreover, in almost voting ensemble methods, ensuring good generalization needs to enlarge the number of ensemble base classifiers. Weighted majority voting scheme can reduce the number of the required ensemble to get the final decision.

In the proposed method, the weight which represents the importance of each individual RELM base learner is extracted using the PRESS LOO error metric (e_i). The weight is inversely related to the lowest error in the PRESS and formed with min-max normalization. The weight interval $\xi_i \in [0, 1]$ where $\xi_i = 1$ for the best classifier, while $\xi_i = 0$ for the worst classifier, and the remaining between them.

$$\xi_i = 1 - \frac{e_i - \min(e_i)}{\max(e_i) - \min(e_i)} \tag{5.1}$$

where e_i is the min. PRESS error for each RELM base classifier. It can be evaluated for *N* number of samples using minimum normalized mean square error as:

$$e_i = \frac{MSE}{var(y)} = \frac{{}^1/_N \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{var(Y)} \tag{5.2}$$

Equation 5.2 is modified to PRESS form as equation 5.6 in section (5.8). The most commonly used procedure choosing the classifiers with the smallest error on training data is unfortunately a defective one. Regardless of the training data types and for the sake of reducing the misleading classification performance, a pretty and active combination of the ensemble ELM with the PRESS approach is generalized. Furthermore, to enhance the evaluation of weights, an additional factor called stability factor (SF) is considered within the weights assignment. We can qualify the stability of the output weight vector (β) solution and hence measure the variation between the train and test responses. The proposed SF is calculated as follow:

$$z = \frac{{}^{1}/_{L}\|\beta\|_2^2}{var(\beta)} \text{ and } SF = 1/(1+\exp(-z)) \tag{5.3}$$

Where $\|\beta\|_2^2 = \sum_{i=1}^{L}\|\beta_i\|_2^2$ is L_2-regularization term of vectors β_i. Here, not only the norm-2 is suitable, it can depend on another normalizations $\|\beta\|_n$ such as $n = 1, n = \infty$.

A low stability factor means that a large variations in the solution of the (β), it implies a large change between hidden layer output vectors that generated from train data and test data $\Delta h = h_{test} - h_{train}$, i.e, $var(\beta)$ is large, z is low and SF is less than 1, whereas, if the $var(\beta)$ is low, z becomes high and SF will be approximatlly 1. The low stability factor can reduce the weight assignment to low value, i.e, the corresponding base classifier has specific minor effect in the final decision.

Now, the assignment weight is defined by combining two factors, which can evaluate the accuracy and the stability of each base classifier.

$$\xi_i = \left(1 - \frac{e_i - \min(e_i)}{\max(e_i) - \min(e_i)}\right) \times SF \tag{5.4}$$

The final weights assignment (α_i) for ensemble can be normalized and assessed as in equation 5.5 as follow:

$$\alpha_i = \frac{\xi_i}{\sum_{i=1}^{N}\xi_i} \quad \text{where } \sum_{i=1}^{N}\alpha_i = 1 \tag{5.5}$$

5.7 The Proposed Ensemble Pruning

In ensemble classification system, it needs to trade-off between two facts: many is better than all [192] and the misclassification rates for ensemble algorithms could go to 0% if the independent training number K is sufficiently large. In this work, instead of using all the generated input weight components in ELM, it can select only the subset of accurate and diverse weights. The generated input weights are evaluated by the eigenvalues

projection metric. We firstly initiate (nK) number of random input weights (w_i), where K is an odd number represents the number of the ensemble classifiers, K is selected as odd number to avoid the tie condition during voting and n is any integer multiplication factor, we chose $n = 2$ and $K = 7$ or 9 for all experiments, then finding $\sigma = \text{eigenvalues}\,(w_i)$ and compute the ratio between the maximum and minimum eigenvalues as $c = \sqrt{\sigma_{max}/\sigma_{min}}$, sorting c in descending order, and choose only the K number of input weight vector (w_i) that are corresponding to the highest c.

Choosing the largest c value leads to the largest span between maximum and minimum eigenvalues that ensures generating discriminative feature mapping which can approximate all or almost functions distinctly, it can confirm the matrix $H^{\dagger}$ which is the core of the ELM approach nonsingular with a minimum row or column dimension, stable and allowable for fast calculations. In other words, small c means the feature distributions are weak and correlated and the chance of generating a compactness feature mapping that predicates the desired function is low.

Ensemble Pruning is utilized here in order to produce the discriminative parameters and to minimize the need for large number of base classifiers K required for the ensemble method.

5.8 The Proposed Fast RELM with Weighted Voting Ensemble (WV-RELM)

The RELM in contrast to ELM can provide very consistent results over many trials with varying L, it can also achieve low fluctuating performance and be nearly unaffected by the over-fitting issue. RELM has two hyper-parameters that needs to be tuned which are the hidden nodes number L and the regularization parameter λ. Generally, in RELM, L needs not to be tuned as long as it is large enough.

In this section, efficient LOO-CV formula based on PRESS and SVD is investigated to evaluate the best regularization parameter λ with fast learning speed and minimal user intervention.

The leave-one-out estimation of the local mean integrated squared error is:

$$error_{LOO} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{t_i - \hat{t}_i}{1-(HAT_{ii})}\right)^2 \tag{5.6}$$

where, $HAT = HH^{\dagger}$matrix that requires to compute $H^{\dagger}$ during the training stage and its computation can be reused for estimating the $error_{LOO}$ [193, 194]. HAT can be modified to simpler form utilizing the SVD properties, also equation 5.6 needs the diagonal of the

HAT-matrix, which can be easily obtained by the row-wise dot-product between H and $H^{\dagger}$.

5.8.1 Generation Approach of the Fast HAT-matrix

According to the dimensionality of the applied data (N) and the hidden nodes number (L), there are two cases to factorize the SVD depending on either H or H^T as UDV^T:

❖ $N \geq L$

$$\hat{T} = H\beta = H(H^TH + \lambda I)^{-1}H^TT = HAT.T \tag{5.7}$$

Equation 5.7 can be rewritten as:

$$\hat{T} = H(H^TH + \lambda I)^{-1}H^T.T, \text{ let } H = UDV^T$$

$$\hat{T} = UDV^T(VD^TU^TUDV^T + \lambda VV^T)^{-1}VD^TU^T.T$$

$$\hat{T} = UDV^T(V(D^TD + \lambda I)V^T)^{-1}VD^TU^T.T$$

$$\hat{T} = UDV^T(V^T)^{-1}(D^2 + \lambda I)^{-1}V^{-1}VD^TU^T.T$$

$$\hat{T} = UDV^TV(D^2 + \lambda I)^{-1}V^TVD^TU^T.T = UD(D^2 + \lambda I)^{-1}D^TU^T.T = HAT.T$$

So, $HAT = UD(D^2 + \lambda I)^{-1}D^TU^T$ (5.8)

Since $D, D^T, and\ D^2$ are diagonal matrices, it can simplify the

$D(D^2 + \lambda I)^{-1}D^T = \frac{DD^T}{(D^2+\lambda I)} = \frac{D^2}{(D^2+\lambda I)}$ as a square diagonal matrix with

$$D(D^2 + \lambda I)^{-1}D^T = \begin{bmatrix} \frac{d_{11}^2}{d_{11}^2 + \lambda} & 0 & \cdots & 0 \\ 0 & \frac{d_{22}^2}{d_{22}^2 + \lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{d_{LL}^2}{d_{LL}^2 + \lambda} \end{bmatrix}$$

where d_{ii} is the ith diagonal element of D and $d_{ii} = 0$ for $i > L$. Let $\delta_i = \frac{d_{ii}^2}{d_{ii}^2+\lambda}$, the HAT will be as: $HAT = U\frac{D^2}{(D^2+\lambda I)}U^T$

$$HAT = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1L} \\ u_{21} & u_{22} & \cdots & u_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NL} \end{bmatrix} \cdot \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \\ 0 & \delta_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_L \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{21} & \cdots & u_{N1} \\ u_{12} & u_{22} & \cdots & u_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ u_{1L} & u_{2L} & \cdots & u_{NL} \end{bmatrix}$$

$$HAT = \begin{bmatrix} \sum_{i=1}^{L} u_{1i}u_{1i}\,\delta_i & \sum_{i=1}^{L} u_{1i}u_{2i}\,\delta_i & \cdots & \sum_{i=1}^{L} u_{1i}u_{Ni}\,\delta_i \\ \sum_{i=1}^{L} u_{2i}u_{1i}\,\delta_i & \sum_{i=1}^{L} u_{2i}u_{2i}\,\delta_i & \cdots & \sum_{i=1}^{L} u_{2i}u_{Ni}\,\delta_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{L} u_{Ni}u_{1i}\,\delta_i & \sum_{i=1}^{L} u_{Ni}u_{2i}\,\delta_i & \cdots & \sum_{i=1}^{L} u_{Ni}u_{Ni}\,\delta_i \end{bmatrix} \quad (5.9)$$

In fact, the PRESS method does not need all matrix elements of equation 5.9, it needs only diagonal elements, which are:

$$HAT_{diag} = \begin{bmatrix} \sum_{i=1}^{L} u_{1i}^2\,\delta_i & \sum_{i=1}^{L} u_{2i}^2\,\delta_i & \cdots & \sum_{i=1}^{L} u_{Ni}^2\,\delta_i \end{bmatrix} \quad (5.10)$$

The computational complexity for the HAT_{diag} is $3NL$.

- $L > N$

$\hat{T} = H\beta = HH^T(HH^T + \lambda I)^{-1}T = HAT.T$, let $H^T = UDV^T$

$\hat{T} = VD^TU^TUDV^T(VD^TU^TUDV^T + \lambda VV^T)^{-1}.T$

$\hat{T} = VD^TDV^T(V(D^TD + \lambda I)V^T)^{-1}.T = VD^T\mathrm{D}(D^2 + \lambda I)^{-1}V^T.T$

$\hat{T} = VD^T\mathrm{D}(D^2 + \lambda I)^{-1}V^T.T$, So, $HAT = VD^T\mathrm{D}(D^2 + \lambda I)^{-1}V^T$ (5.11)

Since $D, D^T, and\ D^2$ matrices are diagonal matrices, it can simplify the $D^TD(D^2 + \lambda I)^{-1} = \frac{D^TD}{(D^2+\lambda I)} = \frac{D^2}{(D^2+\lambda I)}$ as a square diagonal matrix with

$$D^TD(D^2 + \lambda I)^{-1} = \begin{bmatrix} \frac{d_{11}^2}{d_{11}^2 + \lambda} & 0 & \cdots & 0 \\ 0 & \frac{d_{22}^2}{d_{22}^2 + \lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{d_{NN}^2}{d_{NN}^2 + \lambda} \end{bmatrix}$$

where d_{ii} is the ith diagonal element of D and $d_{ii} = 0$ for $i > L$. Let $\delta_i = \frac{d_{ii}^2}{d_{ii}^2+\lambda}$, the HAT is as: $HAT = V\frac{D^2}{(D^2+\lambda I)}V^T$

$$HAT = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1N} \\ v_{21} & v_{22} & \cdots & v_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ v_{N1} & v_{N2} & \cdots & v_{NN} \end{bmatrix} \cdot \begin{bmatrix} \delta_1 & 0 & \cdots & 0 \\ 0 & \delta_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_N \end{bmatrix} \cdot \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{N1} \\ v_{12} & v_{22} & \cdots & v_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ v_{1N} & v_{2N} & \cdots & v_{NN} \end{bmatrix}$$

$$HAT = \begin{bmatrix} \sum_{i=1}^{N} v_{1i}v_{1i}\,\delta_i & \sum_{i=1}^{N} v_{1i}v_{2i}\,\delta_i & \cdots & \sum_{i=1}^{n} v_{1i}v_{Ni}\,\delta_i \\ \sum_{i=1}^{N} v_{2i}v_{1i}\,\delta_i & \sum_{i=1}^{N} v_{2i}v_{2i}\,\delta_i & \cdots & \sum_{i=1}^{N} v_{2i}v_{Ni}\,\delta_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} v_{Ni}v_{1i}\,\delta_i & \sum_{i=1}^{N} v_{Ni}v_{2i}\,\delta_i & \cdots & \sum_{i=1}^{N} v_{Ni}v_{Ni}\,\delta_i \end{bmatrix} \quad (5.12)$$

For PRESS, only diagonal elements are needed, which are:

$$HAT_{diag} = \begin{bmatrix} \sum_{i=1}^{N} v_{1i}^2\,\delta_i & \sum_{i=1}^{N} v_{2i}^2\,\delta_i & \cdots & \sum_{i=1}^{N} v_{Ni}^2\,\delta_i \end{bmatrix} \quad (5.13)$$

The computational complexity for the HAT_{diag} is $3N^2$.

5.8.2 Generation Approach of the Fast Output Weights β

Alternatively, instead of using the standard training procedure, it can utilize SVD to reduce the computing costs. After applying PRESS formula and selecting the optimal λ, the trained output weights β can be computed efficiently as:

- $N \geq L$

  $\beta = (H^T H + \lambda I)^{-1} H^T T$, since $H = UDV^T$

  $\beta = (VD^T U^T UDV^T + \lambda VV^T)^{-1} VD^T U^T.T = (V(D^T D + \lambda I)V^T)^{-1} VD^T U^T.T$

  $\beta = (V^T)^{-1}(D^2 + \lambda I)^{-1} V^{-1} VD^T U^T.T = V(D^2 + \lambda I)^{-1} D^T U^T.T$

$$\beta = V.\begin{bmatrix} \frac{d_{11}}{d_{11}^2+\lambda} & 0 & \cdots & 0 \\ 0 & \frac{d_{22}}{d_{22}^2+\lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{d_{LL}}{d_{LL}^2+\lambda} \end{bmatrix}.U^T T$$

Let $\Omega_i = \frac{d_{ii}}{d_{ii}^2+\lambda}$, then β can be efficiently computed as:

$$\beta = V\{[\Omega_1 \quad \Omega_2 \quad \cdots \quad \Omega_L] \circ (U^T T)\} \tag{5.14}$$

❖ $L > N$

$\beta = H^T(HH^T + \lambda I)^{-1}T$, since $H^T = UDV^T$

$\beta = UDV^T(VD^TU^TUDV^T + \lambda VV^T)^{-1}.T = UD^TV^T(V(D^TD + \lambda I)V^T)^{-1}.T$

$\beta = UD^T(D^2 + \lambda I)^{-1}V^T.T = U\mathrm{D}(D^2 + \lambda I)^{-1}V^T.T$

$$\beta = U.\begin{bmatrix} \frac{d_{11}}{d_{11}^2+\lambda} & 0 & \cdots & 0 \\ 0 & \frac{d_{22}}{d_{22}^2+\lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{d_{NN}}{d_{NN}^2+\lambda} \end{bmatrix}.V^T T$$

Let $\Omega_i = \frac{d_{ii}}{d_{ii}^2+\lambda}$, then β can be precisely computed as:

$$\beta = U\{[\Omega_1 \quad \Omega_2 \quad \cdots \quad \Omega_N] \circ (V^T T)\} \tag{5.15}$$

where ∘ is the element wise multiplication between two matrices.

The overall WV-RELM ensemble algorithm is as follow:

Algorithm 5.1. Ensemble WV-RELM

Suppose a training set of N samples each with n features is given as $\{(x_i, t_i) | \, x_i \in R^n, \, t_i \in R^m, \, i = 1, \cdots, N\}$, with any suitable activation function $G(x)$, hidden nodes number L, independent training odd number K, using $X \in R^{N \times n}$ and $T \in R^{N \times m}$ with equations 2.6 and 2.7 to find the hidden layer output and target matrices $H_{N \times L}$ and $T_{N \times m}$,

a. Training phase:

1. For $k = 1 \ldots nK$ where n=any small integer number, it can choose $n = 2, 3, \ldots$
2. Randomly assign the input weight and bias $w_i^k \; and \; b_i^k$ $(i = 1, \ldots, L)$
3. Find the $\sigma = \text{eigenvalues}\,(w_i^k)$, then calculate $c^k = \sqrt{\frac{\sigma_{max}}{\sigma_{min}}}$ where σ_{max} and σ_{min} are the maximum and minimum eigenvalues
4. loop k
5. Sort c^k in descending order, choose the corresponding w_i^k for the first K-values of c^k and discard the other.

b. Selection phase:

1. For $k = 1 \ldots K$, do
2. Calculate the $H^k = G(w_i^k . X + b_i^k)$
3. Select $\lambda = [\lambda_{min} : \lambda_{max}]$, do
4. Depending on L and N dimensions, factorize H^k or $(H^k)^T$ using SVD with H^k or $(H^k)^T = UDV^T$ to calculate HAT as:

Either $N \geq L$, $HAT = \; UD(D^2 + \lambda I)^{-1} D^T U^T$

$$HAT = \begin{bmatrix} \sum_{i=1}^{L} u_{1i} u_{1i} \, \delta_i & \sum_{i=1}^{L} u_{1i} u_{2i} \, \delta_i & \cdots & \sum_{i=1}^{L} u_{1i} u_{Ni} \, \delta_i \\ \sum_{i=1}^{L} u_{2i} u_{1i} \, \delta_i & \sum_{i=1}^{L} u_{2i} u_{2i} \, \delta_i & \cdots & \sum_{i=1}^{L} u_{2i} u_{Ni} \, \delta_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{L} u_{Ni} u_{1i} \, \delta_i & \sum_{i=1}^{L} u_{Ni} u_{2i} \, \delta_i & \cdots & \sum_{i=1}^{L} u_{Ni} u_{Ni} \, \delta_i \end{bmatrix}$$

Or $L > \; N$, $HAT = \; VD^T \mathrm{D}(D^2 + \lambda I)^{-1} V^T$

$$HAT = \begin{bmatrix} \sum_{i=1}^{N} v_{1i} v_{1i}\, \delta_i & \sum_{i=1}^{N} v_{1i} v_{2i}\, \delta_i & \cdots & \sum_{i=1}^{N} v_{1i} v_{Ni}\, \delta_i \\ \sum_{i=1}^{N} v_{2i} v_{1i}\, \delta_i & \sum_{i=1}^{N} v_{2i} v_{2i}\, \delta_i & \cdots & \sum_{i=1}^{N} v_{2i} v_{Ni}\, \delta_i \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} v_{Ni} v_{1i}\, \delta_i & \sum_{i=1}^{N} v_{Ni} v_{2i}\, \delta_i & \cdots & \sum_{i=1}^{N} v_{Ni} v_{Ni}\, \delta_i \end{bmatrix}$$

then $\hat{T} = HAT.T$

5. Find the error using $error_{LOO} = \frac{1}{N}\sum_{i=1}^{N}(\frac{t_i - \hat{t}_i}{1-(HAT_{ii})})^2$
6. Loop λ
7. $\lambda_{opt} = argmin_{\lambda}[error_{LOO}]$, i.e, select the λ_{opt} that corresponding to min. LOO error, also
8. Choose $e^k = \frac{MSE}{variance} = \frac{\sum(error_{LOO})}{var(error_{LOO})}$
9. Calculate β^k as:
   a. $\beta^k = ((H^k)^T H^k + \lambda_{opt} I)^{-1}(H^k)^T T$ for $N \geq L$

      $= V\{[\Omega_1 \quad \Omega_2 \quad \cdots \quad \Omega_L] \circ (U^T T)\}$ as equation 5.14
   b. $\beta^k = (H^k)^T (H^k (H^k)^T + \lambda_{opt} I)^{-1} T$ for $L > N$

      $= U\{[\Omega_1 \quad \Omega_2 \quad \cdots \quad \Omega_N] \circ (V^T T)\}$ as equation 5.15
10. loop k
11. Construct the weights for the base classifiers using Weighted Majority Voting as
    a. For $k = 1, \ldots, K$
    b. To increase the weights consideration and dependency on the resultant β^k and to enlarge the span between comparable weights, we added the stability factor SF to the calculated weights as:
    c. $z = \frac{1/L \|\beta^k\|_2^2}{var(\beta^k)}$ and the stability factor $SF = 1/(1 + \exp(-z))$
    d. $\xi^k = \left(1 - \frac{e^k - \min(e^k)}{\max(e^k) - \min(e^k)}\right) \times SF$ and the ensemble weights are $\alpha^k = \frac{\xi^k}{\sum_{k=1}^{N} \xi^k}$
    e. Loop k

c. Testing phase:

Calculate the output o_{test} with respect to each query x_{test} using all base classifiers and finally evaluate the estimated output $\hat{y} = \hat{t}$ as:

1. For $k = 1, \dots, K$ do
2. Using the k-th trained parameters $(w_i^k,\ b_i^k,\ \beta^k)$ to predict the $H_{test}^k = G(w_i^k . x_{test} + b_i^k)$
3. $o_{test}^k = H_{test}^k . \beta^k$
4. Loop k
5. The estimated ensemble output:
   $\hat{y} = argmax_k\{(\alpha^k)^T . o_{test}^k\} = argmax_k \sum_{k=1} \alpha^k . o_{test}^k$

Here, the HAT and β^k in the selection phase of the Algorithm 5.1 can be reformulated depending on the minimized RELM of section (3.9) as follow:

Either $N \geq L$, $HAT = U . \sum_{i=1}^{L} \left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right) . U^T$ as equation 3.29

or $L > N$, $HAT = U . \sum_{i=1}^{N} \left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right) . U^T$ as equation 3.35,

then $\hat{T} = HAT . T$, also

a. $\beta^k = ((H^k)^T H^k + \lambda_{opt} I)^{-1} (H^k)^T T$ for $N \geq L$

$= V . \sum_{i=1}^{L} \left(\frac{d_{ii}}{d_{ii}^2+\lambda}\right) . U^T T$ as equation 3.27

b. $\beta^k = (H^k)^T (H^k (H^k)^T + \lambda_{opt} I)^{-1} T$ for $L > N$

$= V . \sum_{i=1}^{N} \left(\frac{d_{ii}}{d_{ii}^2+\lambda}\right) . U^T T$ as equation 3.33

5.9 Computational Complexity of the RELM with Fast PRESS

The computational complexity of the proposed algorithm is focused on solving the RELM base learner to obtain it's regularization and error parameters (λ and $LOO\ error$). These parameters can be found with the normal costly LOO-CV and the enhanced PRESS, Tables 5.1-5.2 compare the computational time complexity between LOO-CV and Fast PRESS for 20 and 50 permutations.

Table 5.1: The computational complexity of the RELM with normal LOO-CV and fast PRESS algorithms for 20 trails

Algorithm	Step	Computational Complexity	Speed-up
LOO-CV	$H^T H$	$2L^2N - L^2$	For example, N=1000, L=1000, the speed up ratio=2.81
	LOO×20	$\{(2L^2 + 5L + 4)N + L^3 + 2L^2 - 2L\}$×20	
	Total	$(42L^2 + 100L + 80)N + 20L^3 + 39L^2 - 40L$	
Fast PRESS	SVD	$14L^2N + 8L^3$	
	LOO×20	$\{(7L + 4)N - L\}$×20	
	β	$2LN - 2L^2 + L$	
	Total	$(14L^2 + 142L + 80)N + 8L^3 - 2L^2 - 19L$	

Table 5.2: The computational complexity of the RELM with normal LOO-CV and fast PRESS algorithms for 50 trails

Algorithm	Step	Computational Complexity	Speed-up
LOO-CV	$H^T H$	$2L^2N - L^2$	For example, N=1000, L=1000, the speed up ratio=6.81
	LOO×50	$\{(2L^2 + 5L + 4)N + L^3 + 2L^2 - 2L\}\times 50$	
	Total	$(102L^2 + 250L + 200)N + 50L^3 + 99L^2 - 100L$	
Fast PRESS	SVD	$14L^2N + 8L^3$	
	LOO×50	$\{(7L + 4)N - L\}\times 50$	
	β	$2LN - 2L^2 + L$	
	Total	$(14L^2 + 352L + 200)N + 8L^3 - 2L^2 - 49L$	

For more illustration to the computational complexity of the proposed method, Figures 5.3 and 5.4 show the computational cost for 20 trials and 50 trials with N=1000 and different hidden nodes range (L).

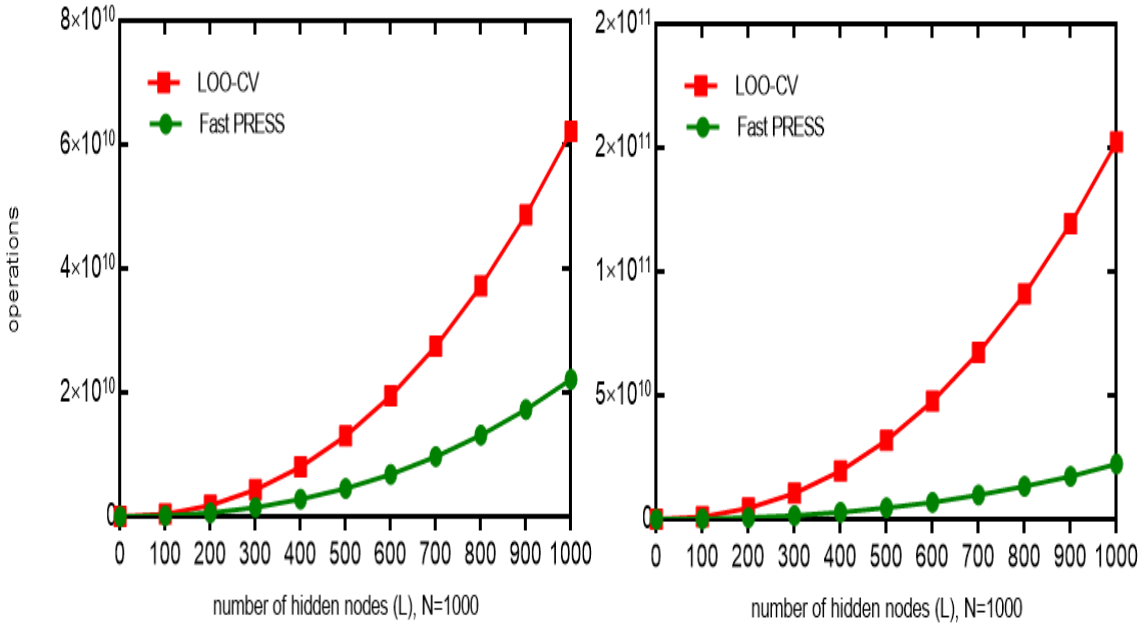


Figure 5.3-5.4 Computational complexity comparison for 20 trails, N=1000 and 50 trails, N=1000 respectively

It is clear from the previous tables and figures that the computational cost of the proposed Fast PRESS algorithm in the training phase is significantly lower than the one of the LOO-CV algorithm.

5.10 Experimental Study

In this section, different aspects of experiments are presented in order to gain insight and evaluate the proposed approach in data and image classification with face and object recognitions. Three categories of benchmark datasets are employed. The first category contains various UCI (University of California at Irvine) databases [112] that includes binary and multiple classifications types. The second category comprises face images datasets that including ORL, YALE, extended YALE-B, and AR, these images were captured at different times, various lighting and poses, facial expressions and facial details. The third category is object images datasets where COIL-20, and USPS are employed, these images belong to different object recognition, these datasets are briefly described in Table 5.3:

Table 5.3 Details of the UCI, Face, and Object datasets

Type	Database	Size	# Attributes	# Classes	Initialization detail
UCI	Balance	625	4	3	Two-thirds of the whole samples for training and the remaining one-third for testing. The experimental results are based on the average of 20 independent trials.
	Hayes	160	4	3	
	Monk1	556	6	2	
	Monk2	601	6	2	
	Monk3	554	6	2	
	Satellite	6435	36	7	
	Segment	2310	19	7	
	Sonar	208	60	2	
	Waveform	5000	21	3	
	Wine	178	13	3	
Face	ORL	400	1024	40	Randomly [2, 3, 4, 5, 10, 20, 30] samples per subject for training are chosen and the rest for testing. They are based on the average of 50 independent trials.
	YALE	165	1024	15	
	YALE-B	2414	1024	38	
	AR	700	300	100	
Object	COIL20	1440	1024	20	
	USPS	9298	256	10	

All simulations were implemented using MATLAB 8.6 (R2015b) environment and performed on an (Intel Core i5, 2.4 GHz CPU, 4GB RAM) computer. Our proposed algorithm WV-RELM was tested and compared with four ELM well-known approaches: standard ELM [19], VELM [184], EV-ELM [188] and RELM [5, 94]. The decision of the WV-RELM algorithm is based on $K = 9$ homogeneous learners, if the weights of these base learners are comparable, we reduces the ensemble number to $K = 7$. In spite of the ability to employ all activation functions with WV-RELM, the only hidden layer activation functions used for all experiments were the sigmoid and radial base (RBF) functions. For a fair comparison, same ELM spaces and assumptions were used for all ELM models.

Tables 5.4 shows the performance comparisons of ELM, V-ELM, EV-ELM, RELM and WV-RELM with sigmoid (s) and RBF (r) activation functions including train times, average testing accuracies, and standard deviations.

Table 5.4 Performance comparisons with different ELM methods

Database	**Algorithms**	***L***	**# of Ens_ embles**	**Training Time (s)**	**Testing Acc. (%)**	**Std. devi_ ation (%)**
Balance	ELM	100	NA	0.05	89.60	1.34
	V-ELM		20	1.19	90.27	0.68
	EV-ELM		20	3.14	90.84	0.66
	RELM		NA	0.18	91.56	0.78
	WV-RELM(s)		9	0.57	**93.12**	**0.62**
	WV-RELM(r)		9	0.52	**92.87**	**0.67**
Hayes	ELM	50	NA	< 0.01	74.64	5.56
	V-ELM		20	0.40	78.50	3.62
	EV-ELM		20	2.53	79.91	3.22
	RELM		NA	0.05	80.20	3.27
	WV-RELM(s)		7	0.14	**81.38**	**3.19**
	WV-RELM(r)		7	0.10	**81.52**	**3.16**
Monk1	ELM	100	NA	0.06	78.40	2.61
	V-ELM	100	20	0.79	85.89	1.34
	EV-ELM	100	20	2.81	87.13	1.13
	RELM	250	NA	0.26	86.10	1.23
	WV-RELM(s)	250	7	0.64	**88.76**	**0.72**
	WV-RELM(r)	250	7	0.83	**86.62**	**1.02**
Monk2	ELM	100	NA	0.07	78.85	2.07
	V-ELM	100	20	0.92	83.53	0.81
	EV-ELM	100	20	3.23	83.85	0.73
	RELM	250	NA	0.29	83.12	0.97
	WV-RELM(s)	250	7	0.71	**84.42**	**0.74**
	WV-RELM(r)	250	7	0.62	**85.38**	**0.70**

Monk3	ELM	100	NA	0.05	80.67	3.11
	V-ELM	100	20	0.68	89.39	1.07
	EV-ELM	100	20	2.67	89.53	0.94
	RELM	250	NA	0.24	91.03	0.88
	WV-RELM(s)	250	7	0.60	**93.58**	**0.76**
	WV-RELM(r)	250	7	0.63	**91.83**	**0.86**
Satellite	ELM	200	NA	0.35	89.44	0.58
	V-ELM	200	20	15.11	91.56	0.43
	EV-ELM	200	20	17.09	**92.10**	**0.36**
	RELM	1000	NA	5.32	90.1	0.37
	WV-RELM(s)	1000	9	19.35	**92.89**	**0.33**
	WV-RELM(r)	1000	9	18.18	**93.78**	**0.30**
Segment	ELM	200	NA	0.15	94.77	0.57
	V-ELM	200	20	6.13	95.84	0.46
	EV-ELM	200	20	7.78	96.32	0.42
	RELM	1000	NA	2.78	95.41	0.43
	WV-RELM(s)	1000	9	6.41	**97.87**	**0.40**
	WV-RELM(r)	1000	9	6.92	**96.47**	**0.42**
Sonar	ELM	50	NA	< 0.01	77.79	3.42
	V-ELM	50	20	0.26	87.02	1.94
	EV-ELM	50	20	2.20	87.67	1.74
	RELM	500	NA	0.11	85.98	2.32
	WV-RELM(s)	500	7	0.27	**88.52**	**1.62**
	WV-RELM(r)	500	7	0.32	**89.73**	**1.34**
Waveform	ELM	200	NA	0.42	84.82	0.51
	V-ELM		20	12.62	86.15	0.23
	EV-ELM		20	14.56	86.53	0.18
	RELM		NA	1.93	86.27	0.21
	WV-RELM(s)		7	4.70	**87.61**	**0.16**
	WV-RELM(r)		7	3.82	**87.84**	**0.15**
Wine	ELM	30 or 50	NA	< 0.01	97.52	1.50
	V-ELM		20	0.12	98.90	0.85
	EV-ELM		20	2.23	**99.23**	**0.69**
	RELM		NA	0.02	98.13	1.36
	WV-RELM(s)		9	0.07	**99.20**	**0.70**
	WV-RELM(r)		9	0.03	**98.66**	0.94

It is found from the Table 5.4 that, WV-RELM consistently achieves the best performance over all the data sets. WV-RELM enhances the overall performance (accuracy, deviation and consuming time) by improving both the efficiency in learning the model and the accuracy in performing classification. For similar or near number of hidden nodes (L), it is clear that the training time of the WV-RELM is comparable or lower than other ensemble methods V-ELM and EV-ELM. This is because of employing the ensemble pruning that can yield efficient input parameters for minimal number of the ensemble (7 or 9) while other methods need to higher ensemble number (20) to produce success rates.

Table 5.4 reveals that, for almost data sets, the performance of the WV-RELM with sigmoid function (s) is comparable to those for RBF function (r).

The training time cost of WV-RELM is linearly increased with a proportion to the number of independent training (K), input data size (N) and the times used for weighted voting. However, the necessity of searching for an optimum λ and large L in RELM leads to increase the training time significantly. The dimensionality of some data for face and object images is high, therefore to prevent increasing the hidden layer nodes, it needs to employ subspace learning approaches like PCA, LDA, etc.

Typically, the techniques of combining multiple classifiers are widely adopted to improve both reliability and accuracy of a single classifier.

In all above ELM models, the hidden layer weights are initialized randomly without ensemble pruning except the WV-RELM which can use the eigen values for weight assigning refinement. Due to random initialization of input layer weights, the performance of ELM fluctuates. The performance fluctuation due to any change in the parameters of the classification algorithm or training dataset composition is known as error due to variance.

For the rest of experiments, we were used the sigmoid activation function only with all ELM models. Tables 5.5-5.9 show the mean of the classification accuracies for different trains per subject on the ORL, YALE, YALE-B, COIL-20 and USPS datasets respectively using ELM, V-ELM, EV-ELM, and WV-RELM with $L \geq 1000$. Consequently, the data representations handled by suitable feature subspace learning (we used LDA) in the training data enhanced the classification efficiency as compared to other related popular algorithms.

Table 5.5 Performance (mean accuracy± standard deviation) % for ELM methods on ORL dataset

ORL	**2 Train**	**3 Train**	**4 Train**	**5 Train**
ELM	78.72±2.61	84.71±2.21	89.87±1.82	93.56±1.43
V-ELM	81.04±2.44	85.36±1.87	92.85±1.64	95.35±1.51
EV-ELM	83.66±2.40	86.95±1.87	93.87±1.43	**96.03±1.29**
WV-RELM	**85.07±2.24**	**88.67±1.73**	**94.81±1.29**	**96.93±1.11**

Table 5.6 Performance (mean accuracy ± standard deviation) % for ELM methods on YALE dataset

YALE	2 Train	3 Train	4 Train	5 Train
ELM	58.12±4.45	67.65±4.42	74.24±3.54	78.78±3.21
V-ELM	58.76±4.23	68.26±4.30	75.28±3.52	79.45±3.06
EV-ELM	59.09±4.15	69.71±3.87	76.20±3.26	80.12±3.04
WV-RELM	**61.28±4.10**	**71.39±3.84**	**81.49±3.25**	**84.92±2.93**

Table 5.7 Performance (mean accuracy ± standard deviation) % for ELM methods on YALE-B dataset

YALE_B	5 Train	10 Train	20 Train	30 Train
ELM	75.82±1.95	87.38±1.23	93.57±0.84	95.59±0.74
V-ELM	77.84±1.80	88.24±1.43	94.12±0.65	96.19±0.52
EV-ELM	79.03±1.73	89.55±1.38	**95.26±0.61**	**97.00±0.60**
WV-RELM	**80.23±1.52**	**90.53±1.23**	**95.42±0.61**	**98.37±0.57**

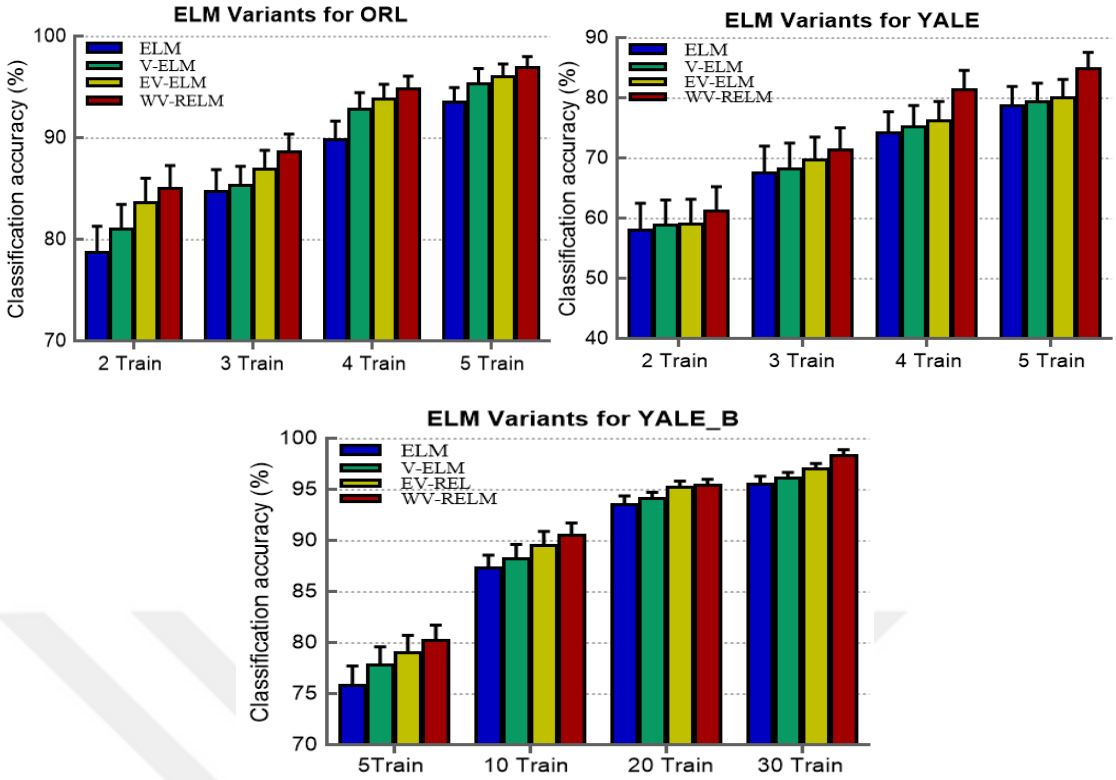
Table 5.8 Performance (mean accuracy) % for ELM methods on COIL-20 dataset

COIL-20	2 Train	4 Train	6 Train	8 Train
ELM	70.8	83.81	86.62	88.92
V-ELM	72.28	84.66	87.23	89.53
EV-ELM	73.82	85.80	87.90	90.72
WV-RELM	**74.77**	**86.57**	**88.49**	**92.34**

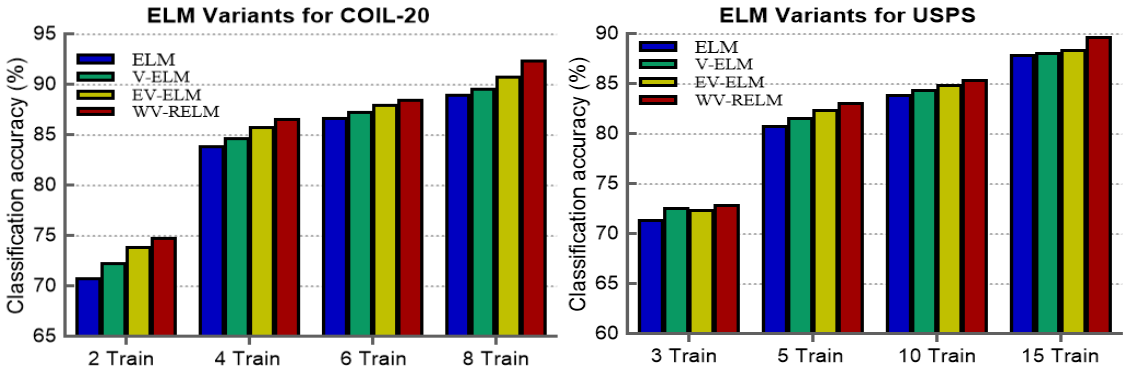
Table 5.9 Performance (mean accuracy) % for ELM methods on USPS dataset

USPS	3 Train	5 Train	10 Train	15 Train
ELM	71.33	80.74	83.13	87.87
V-ELM	72.59	81.58	84.34	88.10
EV-ELM	72.31	82.31	84.84	88.39
WV-RELM	**72.84**	**83.04**	**85.38**	**89.63**

Figures 5.5-5.9 reveal a comprehensive comparison on the accuracies in histogram plots of the corresponding Tables 5.5-5.9 between ELM, V-ELM, EV-ELM, and WV-RELM on the ORL, YALE, YALE-B, COIL-20 and USPS datasets respectively.



Figures 5.5-5.7 ELM methods comparison testing accuracies for face recognition on datasets ORL, YALE, and YALE-B

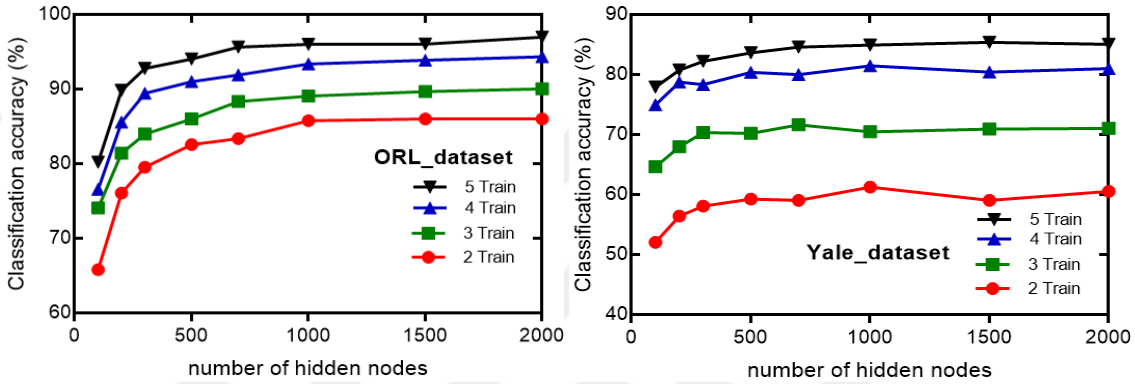


Figures 5.8-5.9 ELM methods comparison testing accuracies for object recognition on dataset COIL-20 and USPS

From the results, it can be seen that the accuracy of the WV-RELM is outperformed over all other ELM models under all train samples, also WV-RELM is more stable where it has the lowest variation in standard deviation. This means that the hidden layer generated with the WV-RELM method is more effective due to the implication of the most discriminative features. The YALE dataset has fewer classes with more feature variations (various facial expressions and illuminations) that can increase the difficulty of

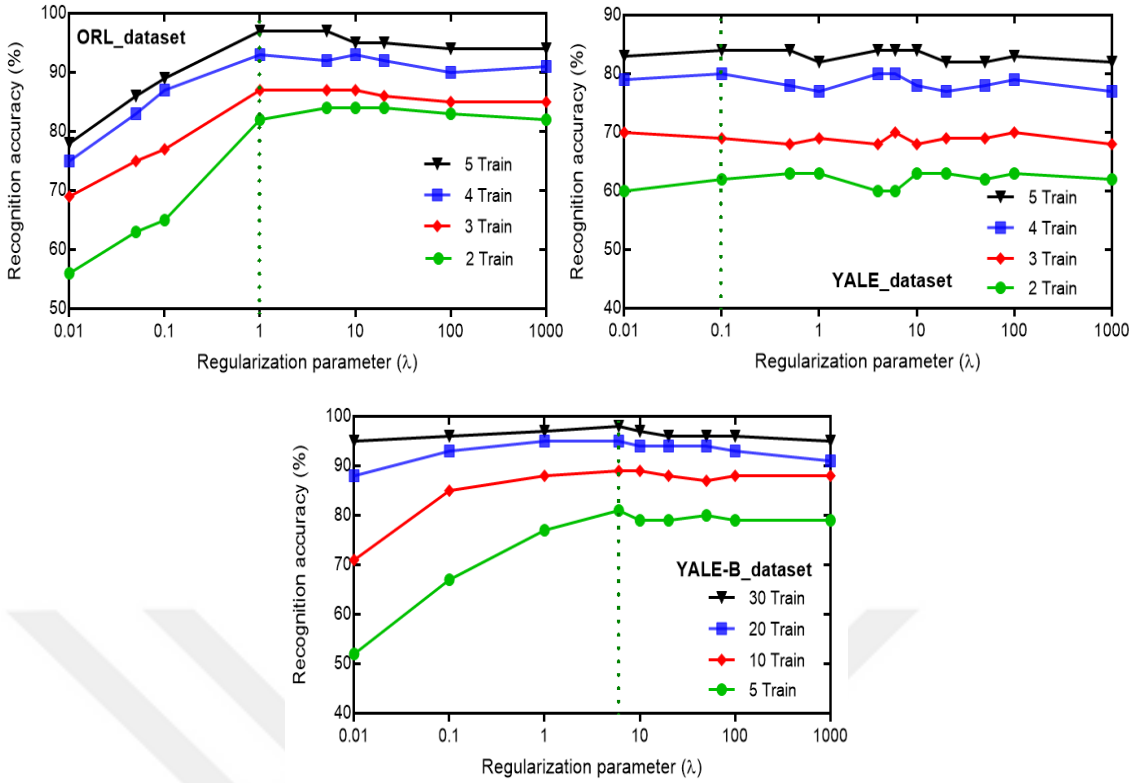
classification. However, the WV-RELM has a distinct accuracy in YALE than other ELM models.

Figures 5.10-5.11 show the classification accuracies of WV-RELM at different trains per subject against the incremental of hidden nodes [100 200 300 500 700 1000 1500 2000] for ORL and YALE datasets respectively. It is clear from these figures that the WV-RELM is stable with respect to different number of hidden neurons and only slight fluctuation when the size of training set is pretty small than 500. The classification performance become less sensitive when the numbers of hidden nodes increase.



Figures 5.10-5.11 WV-RELM performance for different trains per subject verses the incremental of (L)on ORL and YALE

To validate the effectiveness of using the PRESS in RELM, we compare the recognition performance with different choosing λ on the face landmark database. The regularization parameter λ is searched within $\{e^{-10}, e^{-9},\ldots, e^{10}\}$. Among this range, several regularization parameters with $\lambda = [10^{-2}\ 10^{-1}\ \ldots\ 10^{3}\ ]$ range are manually chosen and tested to evaluate the optimum value (λ_{opt}). We firstly fix the number of hidden nodes to a sufficient value as $L = 1000$ then analyze the sensitivity of RELM with λ. Figures 5.12-5.14 represent the experimental results of RELM to parameter λ for ORL, Yale and Extended Yale-B database with different number of training samples per subject. The vertical line shown in each figure allocates the best λ that produces minimum $LOO\ error$ value of RELM. Results exhibit that RELM performs well in ELM training and there is approximately smooth response near the λ_{opt}.



Figures 5.12-5.14 λ-parameters sensitivity analysis of RELM on different trains and various database

For clear depiction of the computational complexity, the waveform database in Table 5.3 is studied as an example, where $N = 5000$ and $L = 200$, according to Table 5.1, the speed up ratio accomplished of RELM base classifier by using Fast PRESS over LOO-CV is (2.88). As shown in Figure 5.15 and for general insight, it can generalize the range L = [100-1000], the range of the speed up ratio = [2.85±0.07].

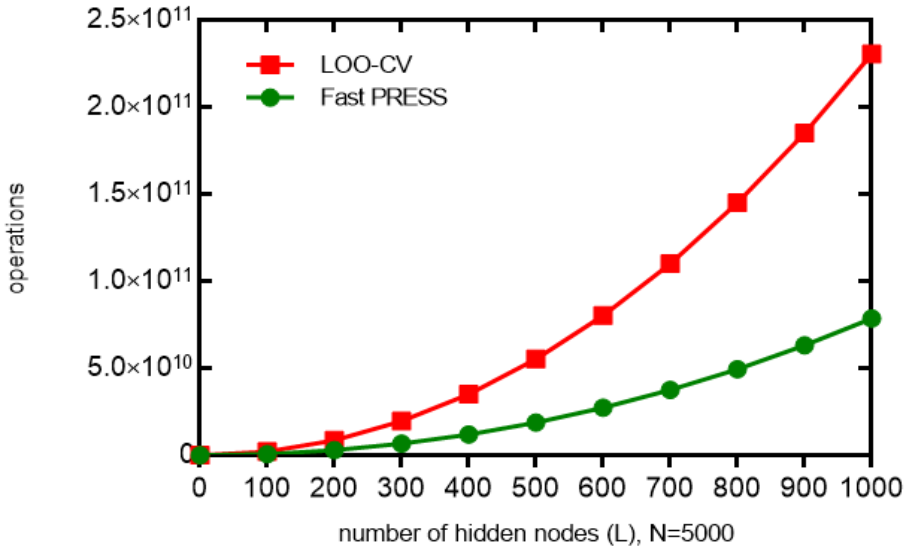


Figure 5.15 The computational complexity for the waveform database against the number of the hidden nodes

It is worth noting that utilizing Dimensionality Reduction (DR) methods can reduce the input dimensionality n to a lower size d, i.e, $(d \ll n)$ and hence improves the performance (accuracy and speed). Also, it can develop a parallel K-ensemble approach with proper parameter initialization that can speed up the overall ensemble with approximately K ratio.

Finally, as in the Tables 5.10-5.11, the performance on YALE-B and AR dataset of WV-RELM approach is compared with the state of the art classification methods: nearest neighbor classifier (k-NN), linear regression classifier (LRC) [118], support vector machine (SVM) [117], sparse representation-based classification (SRC) [116], collaborative representation-based regularized least square (CRC_ RLS) [119]. As it can be seen, the WV-RELM achieves satisfactory performance in all dimension cases, demonstrating the superiority and effectiveness of the proposed paradigm over other methods. Practically the accuracies of the sparse representation methods (SRC and CRC_RLS) and our method are more robust and competitive. However, WV-RELM has the speed advantage while the sparse methods have the immunity advantage against outliers.

Table 5.10 Different state of the art classifier accuracies (%) on YALE-B face recognition dataset for different dimension

YALE_B	**Testing accuracy (%)**		
Algorithm	**Dim d=84**	**Dim d=150**	**Dim d=n=300**
1NN	85.8	90.0	91.6
LRC	94.5	95.1	95.9
SVM	94.9	96.4	97.0
SRC	**95.5**	**96.8**	**97.9**
CRC_RLS	**95.0**	96.3	**97.9**
WV-RELM	**95.7**	**97.8**	**98.9**

Table 5.11 Different state of the art classifier accuracies (%) on AR face recognition dataset for different dimension

AR	**Testing accuracy (%)**		
Algorithm	**Dim d=50**	**Dim d=150**	**Dim d=n=300**
1NN	67.8	70.2	71.3
LRC	71.1	74.6	76.0
SVM	68.3	74.7	75.4
SRC	81.9	90.1	**93.3**
CRC_RLS	80.7	**91.0**	**93.7**
WV-RELM	**84.2**	**91.3**	**93.84**

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusions

Although the extreme learning machine (ELM) has become one of the most important and popular algorithms of machine learning, because of its extremely fast training speed, good generalization, and universal approximation/classification capability, it stills has some issues that need to be solved or enhanced. This thesis focused on studying and tackling some ELMs limitations using different techniques. The research results are summarized and listed as:

- **Optimum, Projected, and L2-Tikhonov Regularized ELM (OELM, PELM, TRELM)**

In standard ELMs with a fixed network architecture, the suitable number of hidden nodes (L) is the key to ELM performance and it is the only factor that needs to be tuned by the user. In most cases, L is arbitrary initiated then gradually increased or decreased by a fixed interval until a nearly optimal number is then selected based on the error metric with the cross-validation method. It is however quite time-consuming. In this work, a simple and fast approach based on the least square method (OELM) was developed to calculate the minimum-optimum hidden node number (L_{opt}). As be assessed with the cross validation tool, our investigated approach can acheive the desired (L_{opt}) or close to it at different data types with less human supervision.

In ELM, both the randomness of the SLFN input weights and the correlated features can lead to poor generalization performance or ill-posed problems. To overcome these restrictions, SVD was exploited significantly as in (PELM) to produce effective input features that can enhance the selections of parameters. For more improvement in the

recognition precision and speed, it can extend the using of SVD to reduce the dimensionality of the input data. Finally, L_2-regularized (Tikhonov Regularization) was added to ELM (TRELM) to prevent the model from overfitting and improve the overall robustness. L-curve and SVD approaches were utilized implicitly within the TRELM to find the regularization parameter, to retain the fast ELM learning advantage and to reduce dramatically the complexity of the matrix calculations. As long as L is large enough, TRELM is more accurate, less sensitive to the changing of L, but undoubtedly needs to longer time than OELM and PELM. Overall, building a suitable ELM with optimum hidden nodes is now easy, fast and has automatic learning ability where the users have limited impact, consequently construction of a suitable hardware capable of working with optimum ELM for biometric classification applications is hereby recommended.

- **Graph Based ELM with L21-Norm Regularization (L21GELM)**

Here, we proposed a stable and efficient approach with a proved convergence called L21GELM. It can tackle some ELM restrictions and yield a more compact and distinct hidden layer through handling the graph Laplacian and L_{21}-norm in local and manifold subspace learning. Graph Laplacian mapping is confirmed to exploit the class similarity advantages of local consistency property of data construction that can increase the classification accuracy, moreover, L_{21}-norm can enhance the rejection of outliers through minimizing the contribution of the mapped noisy data. In addition wide subspace learning approaches can benefit from the geometrical structure of the data to increase the discriminative power of feature representation.

L21GELM was tested and evaluated in the widely adopted face and object recognition datasets under various cases and sizes, also it was compared with relevant ELM works that were: ordinary, regularized and graph ELMs. The proposed L21GELM can achieve much performance gain over various ELM models; furthermore, it is ten times faster and more accurate than other state-of-the-art classification methods. This study demonstrated that the proposed method can be considered as a unified classification method which is suitable for all data sizes including big data, noisy environment, binary and multiclass classification. Moreover, due to the simplicity, robustness, and proper time consuming of L21GELM, it is recommended to be used for multi-class classification tasks of supervised and semi-supervised learning applications even in noisy circumstances.

- **Weighted Voting ensemble model based on Regularized ELM (WV-RELM)**

Employing a single ELM network with random hidden parameters can result in ill-posed problems, which lead to low performance or even no solution. To alleviate this deficiency, ensemble pruning based on eigenvalues was investigated to produce the discriminative parameters that can initially minimize the number of base classifiers. Moreover, a weighted voting method based on both the minimal LOO error with min-max normalization and the stability of the variation of the hidden layer output vectors was developed. This method can robustly specify the ensemble weights distributions with confident classification decision.

RELM as compared to the baseline ELM is more accurate, robust, less sensitive to the changing of L, less sensitive to the overfitting issue and has universal approximation capability; therefore, it was utilized as the base classifier for our homogenous ensemble technique. Both the regularized parameter (λ) of the RELM and the weight coefficients (α^k) of the based ensemble method were optimized using the same PRESS error metric which was LOO error.

To address the slow execution issue of LOO-CV and make the algorithm more practical in real world applications, efficient Leave-One-Out Cross Validation called (Fast PRESS) was proposed. It is of fast learning speed and robust against user intervention.

Exploiting the productive properties of SVD factorization can significantly reduce the complexity of operations and shorten time required for the Fast PRESS approach.

The proposed ensemble approach is not limited to a specific type of ELM like RELM, it can utilize the large variety of ELM algorithms such as kernel types or implement heterogeneous ELM ensemble through building difference base classifiers. Future work includes finding an approach to determine the optimal number of base classifiers to be selected in the ensemble like depending on a threshold metric coincident with updating the whole classification system or employing evolutionary approaches.

6.2 Future Works

Although the current work was attempted to entirely cover and implement different improvements to the ELM method including less human intervention, high computational scalability, low error in learning rate, and a good generalization ability at an extremely fast learning speed, further investigations and improvements on several algorithms, frameworks and directions are recommended as follows:

- For multi-class classification applications, ELM can be applied directly with simpler network structure than other popular learning algorithms, such as BP and SVM algorithms. However, scenarios of multi-class classification methods have borders related to class-specific classification and model expansion such as expanding a pre-trained multi-class model in order to include the entire model. As the model potentially includes more classes, the computational complexities of re-training, storing, and testing using the model increase. Finally, the multi-class classification model does not take into consideration the importance of each class. Since the binary classification types are common and useful in many vital applications such as biomedical classification tasks, failure detection in industrial systems, video surveillance and hyperspectral image classification, it is highly recommended to develop an ELM algorithm that solves this type of classification only. Here, it can exploit the best properties and optimization functions handled by SVM where SVM is considered as a good machine learning tool in binary classification and aims to minimize both the training error and the data dispersion whilst maintaining better solutions for this case.
- It can utilize the QP (Quadratic Programming) optimization used in SVM or use the AL (Augmented Lagrange) algorithm instead of the simple LS (Least Square) that used in ELM to enhance the performance especially on the output weight β. The complexity here may be increased significantly, but probably yielding good improvements.
- Constructing different graph embedding functions, for example building within and between Laplacian graph that can minimize the margins between same or near classes while maximize the margins between different classes, also it can use the kernel aspect with these functions.
- In some fields; for example, biological, clinical study, and some biometric applications, high-dimensional data may be presented by a few number of significant features and factors, thus it is more reasonable to use a proper Dimensionality Reduction methods (DR) model than a dense model. Therefore we would recommend implementing different DRs that are based on statistical or linear and nonlinear geometric construction and compatible with ELM to enhance the feature transformations through producing discriminative features. Some of these DR functions were used with L21GELM as described in chapter four.

- Building multiple nonlinear subspace learning approaches (hierarchical) with or without kernel that can produce and enhance the reduced and discriminative features.
- Exploiting different loss functions such as hinge loss L_1, elastic, Huber and Bisquare types to solve significant machine learning problems like noisy, big, unbalance, and sparse data.
- Implementing weighted ELM that can exploit well-known techniques such as Markov model, Gaussian mixture model, Monte Carlo or the fast Evolutionary Algorithms. For example, utilizing the AL (Augmented Lagrange) optimization technique for adjusting the weights that are necessary for the Weighted ELM or the Ensemble ELM.
- Exploiting fast optimization theories to specify the best set and predetermine the exact number of the base classifiers (K) for an ensemble model that is useful for certain data.
- Developing a nonhomogeneous ensemble method independent on the same base classifiers; in other words, building a heterogeneous system that depends on different sub methods like collecting RELM, KELM, and GELM with advanced methods for combining them to reach the final decision.
- Most available ELM models focus on supervised learning scenarios while little effort can be made to extend ELM into semi-supervised and unsupervised learning field.
- Employing the ECOC (Error Correcting Output Code) that commonly used for correcting bits in communication or for converting binary classifiers into multi-class classifiers in machine learning. ECOC can be used within the ensemble setting by encoding different classes for each member of the ensemble, i.e. individual classifiers are trained on several problems to generate a code vector. The ensemble decision is then chosen depending on the shortest Hamming distance between this code vector and each code word in the code matrix. Another assumption, if the output code vector does not exactly match any of the code words, the largest minimum Hamming distance between code words can be considered as misclassifications individual classifiers.
- Exploiting the ELM learning strategies to find a convenient or minimized feature subset corresponding to the output labels. As a result, the latent feature group structures can be selected or extracted through using ELM. The motivation may be helpful for feature selection in big dimensional datasets.

- Practically, the accuracies of the sparse representation methods (SRC and CRC_RLS) and ELM methods are more robust and competitive. However, ELMs have the speed advantage while the sparse methods have the immunity advantage against outliers. For these reasons and in order to incorporate their respective advantages and overcome their respective drawbacks, building a hybrid algorithm that includes both concepts is recommended, it can greatly enhance the classification performance.
- Implementing hierarchical ELM that extends ELM from single hidden layer feedforward networks to a multilayer perceptron to greatly strengthen the applicability of ELM especially for big data cases.
- Examining a parallel ELM network structure where each individual network can handle a certain part of the data, then applying proper information fusion that can optimally combine the decisions to improve the performance.
- Introducing a hybrid scheme that includes Convolutional Neural Network (CNN) to extract the features from the input images by utilizing the multiple layers of nonlinear class-specific subspace learning and Extreme Learning Machine (ELM) to classify the intermediate results.
- Several applications like large scale media analysis problems, natural language processing, image retrieval have been analyzed in multiple levels and have complex classification problems. Here they need to combine the pros of many approaches and topologies, such as extreme learning machines, multi-layer perceptrons, independent subspace analysis, and deep learning architectures.
- Engaging pre-techniques like clustering approaches to reduce the number of samples or utilizing the non-repeating selecting methods like SVM to choose discriminative samples for reducing training time while maintaining high testing accuracy.

REFERENCES

[1] Specht, D. F., (1991). “A General Regression Neural Network.” IEEE Transactions on Neural Networks 2(6):568–576.

[2] Qu, Y.J. and Hu, B.G., (2011). “Generalized Constraint Neural Network Regression Model Subject to Linear Priors.” IEEE Transactions on Neural Networks 22(12):2447–2459.

[3] Zhang, G. P., (2000). “Neural Networks for Classification: A Survey.” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 30(4):451–462.

[4] Ou, G. and Murphey, Y. L., (2007). “Multi-Class Pattern Classification Using Neural Networks.” Pattern Recognition 40(1):4–18.

[5] Huang, G.B., Zhou, H., Ding, X. and Zhang, R., (2012). “Extreme Learning Machine for Regression and Multiclass Classification.” IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics 42(2):513–529.

[6] Bishop, C. M., (1995). Neural Networks for Pattern Recognition. Oxford university press.

[7] Knerr, S., Personnaz, L. and Dreyfus, G., (1992). “Handwritten Digit Recognition by Neural Networks with Single-Layer Training.” IEEE Transactions on Neural Networks 3(6):962–968.

[8] Lippmann, R. P., (1989). “Review of Neural Networks for Speech Recognition.” Neural Computation 1(1):1–38.

[9] Hunt, K. J., Sbarbaro, D., Żbikowski, R. and Gawthrop, P. J., (1992). “Neural Networks for Control Systems—a Survey.” Automatica 28(6):1083–1112.

[10] Levin, E., Gewirtzman, R. and Inbar, G. F., (1991). “Neural Network Architecture for Adaptive System Modeling and Control.” Neural Networks 4(2):185–191.

[11] Fierro, R. and Lewis, F. L., (1999). “Multilayer Feedforward Networks Are Universal Approximators.” IEEE Trans. Syst., Man, Cybern 29(6):649–654.

[12] Hornik, K., (1991). “Approximation Capabilities of Multilayer Feedforward Networks.” Neural Networks 4(2):251–257.

[13] Hornik, K., (1993). “Some New Results on Neural Network Approximation.” Neural Networks 6(8):1069–1072.

[14] Haykin, S. and Network, N., (2004). “A Comprehensive Foundation.” Neural Networks 2(2004):41.

[15] Rumelhart, D. E., Hinton, G. E. and Williams, R. J., (1985). Learning Internal Representations by Error Propagation. California Univ San Diego La Jolla Inst for Cognitive Science.

[16] Huang, G.B., Zhu, Q.Y. and Siew, C.K., (2004). “Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks.” Proceedings of IEEE International Joint Conference on Neural Networks 2:985–990.

[17] Levine, D. S., (2013). Neural Networks for Knowledge Representation and Inference. Psychology Press.

[18] Nelles, O., (2013). Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models. Springer Science & Business Media.

[19] Huang, G.B., Zhu, Q.Y. and Siew, C.K., (2006). “Extreme Learning Machine: Theory and Applications.” Neurocomputing 70(1–3):489–501.

[20] Huang, G. Bin, Chen, L. and Siew, C. K., (2006). “Universal Approximation using Incremental Constructive Feedforward Networks with Random Hidden Nodes.” IEEE Transactions on Neural Networks 17(4):879–892.

[21] Geman, S., Bienenstock, E. and Doursat, R., (1992). “Neural Networks and the Bias/variance Dilemma.” Neural Computation 4(1):1–58.

[22] Dreiseitl, S. and Ohno-Machado, L., (2002). “Logistic Regression and Artificial Neural Network Classification Models: A Methodology Review.” Journal of Biomedical Informatics 35(5–6):352–359.

[23] Pisani, D., (2004). “Matrix Decomposition Algorithms for Feature Extraction.” BioScience (1).

[24] Moravec, P., (2009). “Dimension Reduction Methods for Iris Recognition.” CEUR Workshop Proceedings 80–89.

[25] Xu, X., Wang, Z., Zhang, X., Yan, W., Deng, W. and Lu, L., (2013). “Human Face Recognition Using Multi-Class Projection Extreme Learning Machine.” 2(6).

[26] Allen, D. M., (1974). “The Relationship between Variable Selection and Data Agumentation and a Method for Prediction.” Technometrics 16(1):125–127.

[27] Myers, R. H., (1990). Classical and Modern Regression with Applications.

[28] Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C. and Lendasse, A., (2010). “OP-ELM: Optimally Pruned Extreme Learning Machine.” IEEE Transactions on Neural Networks 21(1):158–162.

[29] Han, M. and Wang, X., (2012). “A Modified Fast Recursive Hidden Nodes Selection Algorithm for ELM.” Pp. 1–7 in Neural Networks (IJCNN), The 2012 International Joint Conference on IEEE.

[30] Lan, Y., Soh, Y. C. and Huang, G. B., (2010). “Two-Stage Extreme Learning Machine for Regression.” Neurocomputing 73(16–18):3028–3038.

[31] Pouzols, F. M. and Lendasse, A., (2010). “Evolving Fuzzy Optimally Pruned Extreme Learning Machine for Regression Problems.” Evolving Systems 1(1):43–58.

[32] Sovilj, D., Lendasse, A. and Simula, O., (2013). “Extending Extreme Learning Machine with Combination Layer.” Pp. 417–426 in International Work-Conference on Artificial Neural Networks. Springer.

[33] Van Heeswijk, M., Miche, Y., Oja, E. and Lendasse, A., (2011). “GPU-Accelerated and Parallelized ELM Ensembles for Large-Scale Regression.” Neurocomputing 74(16):2430–2437.

[34] SHEHAB, M. A. and KAHRAMAN, N. (2018) “Optimum, Projected and Regularized Extreme Learning Machine Methods with Singular Value Decomposition and L2-Tikhonov Regularization.” Turk J Elec Eng & Comp Sci (2018) 26: 1685 – 1697, © TÜBİTAK, doi:10.3906/elk-1706-60.

[35] Pang, J., Cheung, G., Ortega, A. and Au, O. C., (2015). “Optimal Graph Laplacian Regularization for Natural Image Denoising.” ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings 2015–Augus(1):2294–2298.

[36] Widmer, E. and Bärtschi, A., (2010). “Regularization Methods Inverse Problems in Image Processing 2010 Outline Introduction.”

[37] Shehab, M. A., Kahraman, N. and Bilgin, G., (2017). “Face Recognition Using Graph Extreme Learning Machine with L21-Norm Regularization.” in Electrical and Electronics Engineering (ELECO), 2017 10th International Conference on IEEE. 881–884.

[38] Dietterich, T. G., (2000). “Ensemble Methods in Machine Learning.” Multiple Classifier Systems 1857:1–15.

[39] Hansen, L. K. and Salamon, P., (1990). “Neural Network Ensembles.” IEEE Transactions on Pattern Analysis and Machine Intelligence 12(10):993–1001.

[40] Baum, E. B., (1988). “On the Capabilities of Multilayer Perceptrons.” Journal of Complexity 4(3):193–215.

[41] Bazi, Y., Alajlan, N., Melgani, F., AlHichri, H., Malek, S. and Yager, R. R., (2014). “Differential Evolution Extreme Learning Machine for the Classification of Hyperspectral Images.” IEEE Geoscience and Remote Sensing Letters 11(6):1066–1070.

[42] Cao, F., Liu, B. and Park, D. S., (2013). “Image Classification Based on Effective Extreme Learning Machine.” Neurocomputing 102:90–97.

[43] Chang, N. B., Han, M., Yao, W., Chen, L. C. and Xu, S., (2010). “Change Detection of Land Use and Land Cover in an Urban Region with SPOT-5 Images and Partial Lanczos Extreme Learning Machine.” Journal of Applied Remote Sensing 4(1):43551.

[44] Sovilj, D., Sorjamaa, A., Yu, Q., Miche, Y. and Séverin, E., (2010). “OPELM and OPKNN in Long-Term Prediction of Time Series using Projected Input Data.” Neurocomputing 73(10–12):1976–1986.

[45] Butcher, J. B., Verstraeten, D., Schrauwen, B., Day, C. R. and Haycock, P. W., (2013). “Reservoir Computing and Extreme Learning Machines for Non-Linear Time-Series Data Analysis.” Neural Networks 38:76–89.

[46] Wang, X. and Han, M., (2014). “Multivariate Time Series Prediction Based on Multiple Kernel Extreme Learning Machine.” Pp. 198–201 in Neural Networks (IJCNN), 2014 International Joint Conference on IEEE.

[47] Hong-Li, Z. X. W., (2011). “Incremental Regularized Extreme Learning Machine Based on Cholesky Factorization and Its Application to Time Series Prediction [J].” Acta Physica Sinica 11:1.

[48] Yang, H., Xu, W., Zhao, J., Wang, D. and Dong, Z., (2011). “Predicting the Probability of Ice Storm Damages to Electricity Transmission Facilities Based on ELM and Copula Function.” Neurocomputing 74(16):2573–2581.

[49] Martínez-Rego, D., Fontenla-Romero, O., Pérez-Sánchez, B. and Alonso-Betanzos, A., (2010). “Fault Prognosis of Mechanical Components Using on-Line Learning Neural Networks.” Pp. 60–66 in International Conference on Artificial Neural Networks. Springer.

[50] Pal, M., Maxwell, A. E. and Warner, T. A., (2013). “Kernel-Based Extreme Learning Machine for Remote-Sensing Image Classification.” Remote Sensing Letters 4(9):853–862.

[51] Wang, T., Wang, S. and Zhang, H., (2009). “Dynamic Extreme Learning Machine: A Learning Algorithm for Neural Network with Elastic Output Structure.” Pp. 271–275 in Proceedings of the International Symposium on Intelligent Information Systems and Applications.

[52] Suresh, S., Babu, R. V. and Kim, H. J., (2009). “No-Reference Image Quality Assessment Using Modified Extreme Learning Machine Classifier.” Applied Soft Computing 9(2):541–552.

[53] Duan, G., Huang, Z. and Wang, J., (2009). “Extreme Learning Machine for Bank Clients Classification.” Pp. 496–499 in Information Management, Innovation Management and Industrial Engineering, International Conference on vol.2. IEEE.

[54] Sun, Z.L., Choi, T.M., Au, K.F. and Yu, Y., (2008). “Sales Forecasting Using Extreme Learning Machine with Applications in Fashion Retailing.” Decision Support Systems 46(1):411–419.

[55] Kim, J., Shin, H. S., Shin, K. and Lee, M., (2009). “Robust Algorithm for Arrhythmia Classification in ECG Using Extreme Learning Machine.” Biomedical Engineering Online 8(1):31.

[56] Lan, Y., Soh, Y. C. and Huang, G.B., (2008). “Extreme Learning Machine Based Bacterial Protein Subcellular Localization Prediction.” Pp. 1859–1863 in Neural Networks, International Joint Conference on IEEE.

[57] Zhang, R., Huang, G.B., Sundararajan, N. and Saratchandran, P., (2007). “Multicategory Classification Using an Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis.” IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 4(3):485–495.

[58] Bhat, A. U., Merchant, S. S. and Bhagwat, S. S., (2008). “Prediction of Melting Points of Organic Compounds Using Extreme Learning Machines.” Industrial & Engineering Chemistry Research 47(3):920–925.

[59] Hu, X., Zhao, Z., Wang, S., Wang, F., He, D. and Wu, S., (2008). "Multi-Stage Extreme Learning Machine for Fault Diagnosis on Hydraulic Tube Tester." Neural Computing and Applications 17(4):399–403.

[60] Nizar, A. H., Dong, Z. Y. and Wang, Y., (2008). "Power Utility Nontechnical Loss Analysis with Extreme Learning Machine Method." IEEE Transactions on Power Systems 23(3):946–955.

[61] Pal, M., (2009). "Extreme-learning-machine-based Land Cover Classification." International Journal of Remote Sensing 30(14):3835–3841.

[62] Yeu, C.-W. T., Lim, M.H. and Huang, G.B., (2006). "Terrain Modeling Using Machine Learning Methods." Pp. 1–4 in Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on IEEE.

[63] Zhai, C.M. and Du, J.X., (2008). "Applying Extreme Learning Machine to Plant Species Identification." Pp. 879–884 in Information and Automation, 2008. International Conference on IEEE.

[64] Loh, Q.J. B. and Emmanuel, S., (2006). "ELM for the Classification of Music Genres." Pp. 1–6 in Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on IEEE.

[65] Huang, G.B., (2014). "An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels." Cognitive Computation 6(3):376–390.

[66] Huang, G. Bin, Bai, Z., Kasun, L. L. C. and Vong, C. M., (2015). "Local Receptive Fields Based Extreme Learning Machine." IEEE Computational Intelligence Magazine 10(2):18–29.

[67] Huang, G.B. and Chen, L., (2008). "Enhanced Random Search Based Incremental Extreme Learning Machine." Neurocomputing 71(16–18):3460–3468.

[68] Petersen, K. B. and Pedersen, M. S., (2007). "The Matrix Cookbook." Citeseer 16(4):1–66.

[69] Bartlett, P. L., (1998). "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.

[70] Feng, G., Huang, G.B., Lin, Q. and Gay, R., (2009). "Error Minimized Extreme Learning Machine with Growth of Hidden Nodes and Incremental Learning." IEEE Transactions on Neural Networks 20(8):1352–1357.

[71] Rong, H.J., Ong, Y.S., Tan, A.H. and Zhu, Z., (2008). "A Fast Pruned-Extreme Learning Machine for Classification Problem." Neurocomputing 72(1–3):359–366.

[72] Miche, Y., Sorjamaa, A. and Lendasse, A., (2008). "OP-ELM: Theory, Experiments and a Toolbox." Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5163 LNCS(PART 1):145–154.

[73] Huang, G.B. and Chen, L., (2007). "Convex Incremental Extreme Learning Machine." Neurocomputing 70(16–18):3056–3062.

[74] Barron, A. R., (1993). “Universal Approximation Bounds for Superpositions of a Sigmoidal Function.” IEEE Transactions on Information Theory 39(3):930–945.

[75] Lan, Y., Soh, Y. C. and Huang, G.B., (2010). “Random Search Enhancement of Error Minimized Extreme Learning Machine.” in ESANN.

[76] Zhang, R., Lan, Y., Huang, G.B., Xu, Z.B. and Soh, Y. C., (2013). “Dynamic Extreme Learning Machine and Its Approximation Capability.” IEEE Transactions on Cybernetics 43(6):2054–2065.

[77] Yang, Y., Wang, Y. and Yuan, X., (2012). “Bidirectional Extreme Learning Machine for Regression Problem and Its Learning Effectiveness.” IEEE Transactions on Neural Networks and Learning Systems 23(9):1498–1505.

[78] Deng, J., Li, K. and Irwin, G. W., (2011). “Fast Automatic Two-Stage Nonlinear Model Identification Based on the Extreme Learning Machine.” Neurocomputing 74(16):2422–2429.

[79] Zhang, R., Lan, Y., Huang, G. and Xu, Z.B., (2012). “Universal Approximation of Extreme Learning Machine with Adaptive Growth of Hidden Nodes.” IEEE Transactions on Neural Networks and Learning Systems 23(2):365–371.

[80] Du, D., Li, K., Irwin, G. W. and Deng, J., (2013). “A Novel Automatic Two-Stage Locally Regularized Classifier Construction Method Using the Extreme Learning Machine.” Neurocomputing 102:10–22.

[81] Wang, N., Er, M. J. and Han, M., (2014). “Parsimonious Extreme Learning Machine Using Recursive Orthogonal Least Squares.” IEEE Transactions on Neural Networks and Learning Systems 25(10):1828–1841.

[82] Luo, J., Vong, C.M. and Wong, P.K., (2014). “Sparse Bayesian Extreme Learning Machine for Multi-Classification.” IEEE Transactions on Neural Networks and Learning Systems 25(4):836–843.

[83] Liang, N.Y., Huang, G. B., Saratchandran, P. and Sundararajan, N., (2006). “A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks.” IEEE Transactions on Neural Networks 17(6):1411–1423.

[84] Zong, W., Huang, G. Bin and Chen, Y., (2013). “Weighted Extreme Learning Machine for Imbalance Learning.” Neurocomputing 101:229–242.

[85] Bai, Z., Huang, G.B., Wang, D., Wang, H. and Westover, M. B., (2014). “Sparse Extreme Learning Machine for Classification.” IEEE Transactions on Cybernetics 44(10):1858–1870.

[86] Zhang, K. and Luo, M., (2015). “Outlier-Robust Extreme Learning Machine for Regression Problems.” Neurocomputing 151(P3):1519–1527.

[87] Akusok, A., Bjork, K. M., Miche, Y. and Lendasse, A., (2015). “High-Performance Extreme Learning Machines: A Complete Toolbox for Big Data Applications.” IEEE Access 3:1011–1025.

[88] Chen, K., Lv, Q., Lu, Y. and Dou, Y., (2017). “Robust Regularized Extreme Learning Machine for Regression Using Iteratively Reweighted Least Squares.” Neurocomputing 230(December 2016):345–358.

[89] Kwok, T.Y. and Yeung, D.Y., (1997). “Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems.” IEEE Transactions on Neural Networks 8(3):630–645.

[90] Akaike, H., (1974). “A New Look at the Statistical Model Identification.” IEEE Transactions on Automatic Control 19(6):716–723.

[91] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R., (2004). “Least Angle Regression.” The Annals of Statistics 32(2):407–499.

[92] Similä, T. and Tikka, J., (2005). “Multiresponse Sparse Regression with Application to Multidimensional Scaling.” Pp. 97–102 in International Conference on Artificial Neural Networks. Springer.

[93] Wang, N., Han, M., Dong, N. and Er, M. J., (2014). “Constructive Multi-Output Extreme Learning Machine with Application to Large Tanker Motion Dynamics Identification.” Neurocomputing 128:59–72.

[94] Deng, W., Zheng, Q. and Chen, L., (2009). “Regularized Extreme Learning Machine.” Pp. 389–395 in Computational Intelligence and Data Mining, 2009. CIDM’09. IEEE Symposium.

[95] Miche, Y., van Heeswijk, M., Bas, P., Simula, O. and Lendasse, A., (2011). “TROP-ELM: A Double-Regularized ELM Using LARS and Tikhonov Regularization.” Neurocomputing 74(16):2413–2421.

[96] Fakhr, M. W., Youssef, E. N. S. and El-Mahallawy, M. S., (2015). “L1-Regularized Least Squares Sparse Extreme Learning Machine for Classification.” 2015 International Conference on Information and Communication Technology Research, ICTRC 222–225.

[97] Wang, Y., Cao, F. and Yuan, Y., (2011). “A Study on Effectiveness of Extreme Learning Machine.” Neurocomputing 74(16):2483–2490.

[98] Luo, M. and Zhang, K., (2014). “A Hybrid Approach Combining Extreme Learning Machine and Sparse Representation for Image Classification.” Engineering Applications of Artificial Intelligence 27:228–235.

[99] Huang, G., Huang, G. Bin, Song, S. and You, K., (2015). “Trends in Extreme Learning Machines: A Review.” Neural Networks 61:32–48.

[100] Liu, X., Wang, L., Huang, G.B., Zhang, J. and Yin, J., (2015). “Multiple Kernel Extreme Learning Machine.” Neurocomputing 149, Part(September):253–264.

[101] Huang, G., Song, S., Gupta, J. N. D. and Wu, C., (2014). “Semi-Supervised and Unsupervised Extreme Learning Machines.” IEEE Transactions on Cybernetics 44(12):2405–2417.

[102] Cao, J., Lin, Z. and Huang, G. Bin, (2012). “Self-Adaptive Evolutionary Extreme Learning Machine.” Neural Processing Letters 36(3):285–305.

[103] Akusok, A., (2016). Extreme Learning Machines: Novel Extensions and Application to Big Data. The University of Iowa.

[104] Cao, W., Ming, Z., Wang, X. and Cai, S., (2017). “Improved Bidirectional Extreme Learning Machine Based on Enhanced Random Search.” Memetic Computing 1–8.

[105] Cambria, E., Huang, G.B., Kasun, L. L. C., Zhou, H., Vong, C. M., Lin, J., Yin, J., Cai, Z., Liu, Q. and Li, K., (2013). “Extreme Learning Machines [Trends & Controversies].” IEEE Intelligent Systems 28(6):30–59.

[106] Huang, G. Bin, (2015). “What Are Extreme Learning Machines? Filling the Gap Between Frank Rosenblatt's Dream and John von Neumann's Puzzle.” Cognitive Computation 7(3):263–278.

[107] Zhu, W., Miao, J. and Qing, L., (2015). “Constrained Extreme Learning Machines: A Study on Classification Cases.” 14.

[108] Deng, W. Y., Zheng, Q. H. and Wang, Z. M., (2013). “Projection Vector Machine.” Neurocomputing 120:490–498.

[109] Hoerl, A. E. and Kennard, R. W., (2000). “Ridge Regression: Biased Estimation for Nonorthogonal Problems.” Technometrics 42(1):80–86.

[110] Chung, J., Español, M. I. and Nguyen, T., (2014). “Optimal Regularization Parameters for General-Form Tikhonov Regularization.” 1–21.

[111] Henderson, H. V and Searle, S. R., (1981). “On Deriving the Inverse of a Sum of Matrices.” Siam Review 23(1):53–60

[112] “29 UCI Machine Learning Repository.pdf.” .

[113] Van Heeswijk, M. and Miche, Y., (2015). “Binary/ternary Extreme Learning Machines.” Neurocomputing 149(Part A):187–197.

[114] Xiang, H. and Zou, J., (2013). “Regularization with Randomized SVD for Large-Scale Discrete Inverse Problems.” Inverse Problems 29(8):85008.

[115] Hansen, P. C., (1994). “REGULARIZATION TOOLS: A Matlab Package for Analysis and Solution of Discrete Ill-Posed Problems.” Numerical Algorithms 6(1):1–35.

[116] Cao, J., Zhang, K., Luo, M., Yin, C. and Lai, X., (2016). “Extreme Learning Machine and Adaptive Sparse Representation for Image Classification.” Neural Networks 81:91–102.

[117] Suykens, J. A. K. and Vandewalle, J., (1999). “Least Squares Support Vector Machine Classifiers.” Neural Processing Letters 9(3):293–300.

[118] Friedman, J., Hastie, T. and Tibshirani, R., (2000). “Additive Logistic Regression: A Statistical View of Boosting .” The Annals of Statistics 28(2):337–407.

[119] Zhang, L., Yang, M. and Feng, X., (2011). “Sparse Representation or Collaborative Representation: Which Helps Face Recognition?” Pp. 471–478 in Computer vision (ICCV), 2011 IEEE international conference on IEEE.

[120] Xueyi, L., Ping, L. and Chuanhou, G., (2011). “Fast Leave-One-out Cross-Validation Algorithm for Extreme Learning Machine.” Journal of Shanghai Jiaotong University 45(8):1140–1145.

[121] Iosifidis, A., Tefas, A. and Pitas, I., (2016). “Graph Embedded Extreme Learning Machine.” IEEE Transactions on Cybernetics 46(1):311–324.

[122] Peng, Y., Wang, S., Long, X. and Lu, B. L., (2015). “Discriminative Graph Regularized Extreme Learning Machine and Its Application to Face Recognition.” Neurocomputing 149(Part A):340–353.

[123] Abdi, H. and Williams, L. J., (2010). “Principal Component Analysis.” Wiley Interdisciplinary Reviews: Computational Statistics 2(4):433–459.

[124] KPFRS, L., (1901). “On Lines and Planes of Closest Fit to Systems of Points in Space.” in Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (SIGMOD).

[125] Ding, C., He, X., Zha, H. and Simon, H. D. (2002) “Adaptive Dimension Reduction for Clustering High Dimensional Data.” Pp. 147–154 in Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on IEEE.

[126] Widodo, A. and Yang, B.-S., (2007). “Application of Nonlinear Feature Extraction and Support Vector Machines for Fault Diagnosis of Induction Motors.” Expert Systems with Applications 33(1):241–250.

[127] Dong, W., Chen, T., Tino, P. and Yao, X., (2013). “Scaling up Estimation of Distribution Algorithms for Continuous Optimization.” IEEE Transactions on Evolutionary Computation 17(6):797–822.

[128] Cai, D., He, X. and Han, J., (2008). “SRDA: An Efficient Algorithm for Large-Scale Discriminant Analysis.” IEEE Transactions on Knowledge and Data Engineering 20(1):1–12.

[129] Baudat, G. and Anouar, F., (2000). “Generalized Discriminant Analysis Using a Kernel Approach.” Neural Computation 12(10):2385–2404.

[130] Mika, S., Ratsch, G., Weston, J., Scholkopf, B. and Mullers, K.-R., (1999). “Fisher Discriminant Analysis with Kernels.” Pp. 41–48 in Neural networks for signal processing IX, 1999.

[131] He, X. and Niyogi, P., (2004). “Locality Preserving Projections.” Pp. 153–160 in Advances in neural information processing systems.

[132] Cai, D., He, X. and Han, J., (2007). “Spectral Regression for Efficient Regularized Subspace Learning.” Pp. 1–8 in Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on IEEE.

[133] Cai, D., He, X., Zhang, W. V. and Han, J., (2007). “Regularized Locality Preserving Indexing via Spectral Regression.” Pp. 741–50 in Proceedings of the sixteenth ACM conference on Conference on information and knowledge management ACM.

[134] He, X., Cai, D., Yan, S. and Zhang, H.-J., (2005). “Neighborhood Preserving Embedding.” Pp. 1208–13 in Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, vol. 2 IEEE.

[135] Peng, H., Long, F. and Ding, C., (2005). “Feature Selection Based on Mutual Information Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy.” IEEE Transactions on Pattern Analysis and Machine Intelligence 27(8):1226–1238.

[136] Yu, L. and Liu, H., (2003). “Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution.” Pp. 856–863 in Proceedings of the 20th international conference on machine learning (ICML-03).

[137] Zhou, L., Wang, L. and Shen, C., (2010). “Feature Selection with Redundancy-Constrained Class Separability.” IEEE Transactions on Neural Networks 21(5):853–858.

[138] Kohavi, R. and John, G. H., (1997). “Wrappers for Feature Subset Selection.” Artificial Intelligence 97(1–2):273–324.

[139] Zhu, Z., Ong, Y.S. and Dash, M., (2007). “Markov Blanket-Embedded Genetic Algorithm for Gene Selection.” Pattern Recognition 40(11):3236–3248.

[140] Guyon, I. and Elisseeff, A., (2003). “An Introduction to Variable and Feature Selection.” Journal of Machine Learning Research 3(Mar):1157–1182.

[141] Blum, A. L. and Langley, P., (1997). “Selection of Relevant Features and Examples in Machine Learning.” Artificial Intelligence 97(1–2):245–271.

[142] Rakotomamonjy, A., (2003). “Variable Selection Using SVM-Based Criteria.” Journal of Machine Learning Research 3(Mar):1357–1370.

[143] Guyon, I., Weston, J., Barnhill, S. and Vapnik, V., (2002). “Gene Selection for Cancer Classification Using Support Vector Machines.” Machine Learning 46(1–3):389–422.

[144] Bradley, P. S. and Mangasarian, O. L., (1998). “Feature Selection via Concave Minimization and Support Vector Machines.” Pp. 82–90 in ICML, vol. 98.

[145] Zhu, J., Rosset, S., Tibshirani, R. and Hastie, T. J., (2004). “1-Norm Support Vector Machines.” Pp. 49–56 in Advances in neural information processing systems.

[146] Fung, G. M. and Mangasarian, O. L., (2004). “A Feature Selection Newton Method for Support Vector Machine Classification.” Computational Optimization and Applications 28(2):185–202.

[147] Tan, M., Wang, L. and Tsang, I. W., (2010). “Learning Sparse Svm for Feature Selection on Very High Dimensional Datasets.” Pp. 1047–1054 in Proceedings of the 27th international conference on machine learning (ICML-10).

[148] Nie, F., Huang, H., Cai, X. and Ding, C. H., (2010). “Efficient and Robust Feature Selection via Joint L2,1-Norms Minimization.” Pp. 1813–1821 in Advances in neural information processing systems.

[149] Feng, G., Lan, Y., Zhang, X. and Qian, Z., (2015). “Dynamic Adjustment of Hidden Node Parameters for Extreme Learning Machine.” IEEE Transactions on Cybernetics 45(2):279–288.

[150] Zhang, D., Xu, Y. and Zuo, W., (2016). “Discriminative Learning in Biometrics.” Pp. 3–20 in Discriminative Learning in Biometrics Springer.

[151] Ma, Z., Yang, Y., Nie, F., Uijlings, J. and Sebe, N., (2011). “Exploiting the Entire Feature Space with Sparsity for Automatic Image Annotation.” Pp. 283–292 in Proceedings of the 19th ACM international conference on Multimedia ACM.

[152] Yang, Y., Shen, H. T., Ma, Z., Huang, Z. and Zhou, X., (2011). “L2,1-Norm Regularized Discriminative Feature Selection for Unsupervised Learning.” P.1589 in IJCAI proceedings-international joint conference on artificial intelligence, vol. 22.

[153] Ren, C. X., Dai, D. Q. and Yan, H., (2012). “Robust Classification using L2,1-Norm Based Regression Model.” Pattern Recognition. Elsevier, 45(7):2708–2718.

[154] Zhou, S., Liu, X., Liu, Q., Wang, S., Zhu, C. and Yin, J., (2016). "Random Fourier Extreme Learning Machine with L2,1-Norm Regularization." Neurocomputing 174:143–153.

[155] Kondor, R. I. and Lafferty, J., (2002). "Diffusion Kernels on Graphs and Other Discrete Input Spaces." Pp. 315–322 in ICML, vol. 2.

[156] Cai, D., He, X., Han, J. and Huang, T. S., (2011). "Graph Regularized Nonnegative Matrix Factorization for Data Representation." IEEE Transactions on Pattern Analysis and Machine Intelligence 33(8):1548–1560.

[157] Belkin, M. and Niyogi, P., (2003). "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." Neural Computation 15(6):1373–1396.

[158] Belkin, M., Niyogi, P. and Sindhwani, V., (2006). "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples." Journal of Machine Learning Research 7(Nov):2399–2434.

[159] Zong, W. and Huang, G.B., (2014). "Learning to Rank with Extreme Learning Machine." Neural Processing Letters 39(2):155–166.

[160] Boyd, S. and Vandenberghe, L., (2004). Convex Optimization. Cambridge university press.

[161] Georghiades, A. S., Belhumeur, P. N. and Kriegman, D. J., (2001). "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose." IEEE Transactions on Pattern Analysis and Machine Intelligence 23(6):643–660.

[162] Lee, K.C., Ho, J. and Kriegman, D. J., (2005). "Acquiring Linear Subspaces for Face Recognition under Variable Lighting." IEEE Transactions on Pattern Analysis and Machine Intelligence 27(5):684–698.

[163] Martinez, A. M., (1998). "The AR Face Database." CVC Technical Report24.

[164] Nene, S. A., Nayar, S. K. and Murase, H., (1996). "Columbia Object Image Library (Coil-20)."

[165] Li, Z., Yang, Y., Liu, J., Zhou, X. and Lu, H., (2012). "Unsupervised Feature Selection Using Nonnegative Spectral Analysis." Pp. 1026–32 in AAAI, vol. 2.

[166] Peng, Y. and Lu, B., (2015). "Discriminative Manifold Extreme Learning Machine and Applications to Image and EEG Signal Classification."

[167] Zhao, G., Shen, Z., Miao, C. and Man, Z., (2009). "On Improving the Conditioning of Extreme Learning Machine: A Linear Case." Pp. 1–5 in Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on IEEE.

[168] Jalab, H. A. and Ibrahim, R. W., (2011). "New Activation Functions for Complex-Valued Neural Network." International Journal of Physical Sciences 6(7):1766–1772.

[169] Javed, K., Gouriveau, R., Zerhouni, N., Zemouri, R. and Li, X., (2012). "Robust, Reliable and Applicable Tool Wear Monitoring and Prognostic: Approach Based on an Improved-Extreme Learning Machine." Pp. 1–9 in Prognostics and Health Management (PHM), Conference on IEEE.

[170] Zhou, Z.H., (2012). Ensemble Methods: Foundations and Algorithms. CRC press.

[171] Breiman, L., (2001). “Random Forests.” Machine Learning 45(1):5–32.

[172] Kirchner, M., Timm, W., Fong, P., Wangemann, P. and Steen, H., (2010). “Non-Linear Classification for on-the-Fly Fractional Mass Filtering and Targeted Precursor Fragmentation in Mass Spectrometry Experiments.” Bioinformatics 26(6):791–797.

[173] Kruglyak, L. and Nickerson, D. A., (2001). “Variation is the Spice of Life.” Nature Genetics 27(3):234.

[174] Barenboim, M., Masso, M., Vaisman, I. I. and Jamison, D. C., (2008). “Statistical Geometry Based Prediction of Nonsynonymous SNP Functional Effects Using Random Forest and Neuro-fuzzy Classifiers.” Proteins: Structure, Function, and Bioinformatics 71(4):1930–1939.

[175] Kuncheva, L. I., Bezdek, J. C. and Duin, R. P. W., (2001). “Decision Templates for Multiple Classifier Fusion: An Experimental Comparison.” Pattern Recognition 34(2):299–314.

[176] Menze, B. H., Kelm, B. M., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W. and Hamprecht, F. A., (2009). “A Comparison of Random Forest and Its Gini Importance with Standard Chemometric Methods for the Feature Selection and Classification of Spectral Data.” BMC Bioinformatics 10(1):213.

[177] Lu, Y., (1996). “Knowledge Integration in a Multiple Classifier System.” Applied Intelligence 6(2):75–86.

[178] Kittler, J., Hatef, M., Duin, R. P. W. and Matas, J., (1998). “On Combining Classifiers.” IEEE Transactions on Pattern Analysis and Machine Intelligence 20(3):226–239.

[179] Kuncheva, L. I., (2004). “Classifier Ensembles for Changing Environments.” Pp. 1–15 in International Workshop on Multiple Classifier Systems Springer.

[180] Han, D. H., Zhang, X. and Wang, G. R., (2015). “Classifying Uncertain and Evolving Data Streams with Distributed Extreme Learning Machine.” Journal of Computer Science and Technology 30(4):874–887.

[181] Wang, X. L., Chen, Y. Y., Zhao, H. and Lu, B. L., (2014). “Parallelized Extreme Learning Machine Ensemble Based on Min-Max Modular Network.” Neurocomputing 128:31–41.

[182] Cao, J., Hao, J., Lai, X., Vong, C. M. and Luo, M., (2016). “Ensemble Extreme Learning Machine and Sparse Representation Classification.” Journal of the Franklin Institute 353(17):4526–4541.

[183] Samat, A., Du, P., Liu, S., Li, J. and Cheng, L., (2014). “E2LMs: Ensemble Extreme Learning Machines for Hyperspectral Image Classification.” IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 7(4):1060–1069.

[184] Cao, J., Lin, Z., Huang, G. Bin and Liu, N. (2012) “Voting Based Extreme Learning Machine.” Information Sciences 185(1):66–77.

[185] Lan, Y., Soh, Y. C. and Huang, G.B., (2009). “Ensemble of Online Sequential Extreme Learning Machine.” Neurocomputing 72(13–15):3391–3395.

[186] Cao, J., Kwong, S. A. M., Wang, R. A. N. and Li, K. E., (2012). “A Weighted Voting Method Using Minimum Square Error Based On Extreme Learning Machine.” International Conference on Machine Learning, 2012, 15–17.

[187] Liu, Y., He, B., Dong, D., Shen, Y., Yan, T., Nian, R. and Lendasse, A., (2015). “ROS-ELM: A Robust Online Sequential Extreme Learning Machine for Big Data Analytics.” 325–344.

[188] Liu, N., Cao, J., Lin, Z., Pek, P. P., Koh, Z. X. and Ong, M. E. H., (2014). “Evolutionary Voting-Based Extreme Learning Machines.” Mathematical Problems in Engineering 2014.

[189] Escandell-montero, P. and Mart, M., (2012). “Regularized Committee of Extreme Learning Machine for Regression Problems.” ESANN 2012 Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. (April):25–27.

[190] Zhang, P. and Yang, Z., (2015). “A Robust AdaBoost.RT Based Ensemble Extreme Learning Machine.” Mathematical Problems in Engineering 2015.

[191] Cao, J., Kwong, S., Wang, R., Li, X., Li, K. and Kong, X., (2015). “Class-Specific Soft Voting Based Multiple Extreme Learning Machines Ensemble.” Neurocomputing 149(Part A):275–284.

[192] Zhou, Z.H., Wu, J. and Tang, W., (2002). “Ensembling Neural Networks: Many Could Be Better than All.” Artificial Intelligence 137(1–2):239–263.

[193] Shao, Z. and Er, M. J., (2016). “An Online Sequential Learning Algorithm for Regularized Extreme Learning Machine.” Neurocomputing 173:778–788.

[194] Shao, Z., Joo, M. and Wang, N., (2015). “An Effective Semi-Cross-Validation Model Selection Method for Extreme Learning Machine with Ridge Regression.” Neurocomputing 151:933–942.

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname : Mohanad Abd Shehab AL_Karawi

Date of birth and place : 01.09.1972, Iraq-Baghdad

Foreign Languages : English and someTurkish

E-mail : engineer.moh72@gmail.com

EDUCATION

Degree	Department	University	Date of Graduation
Msc.	Electrical and Electronic Engineering	University of Technology/ Baghdad	1998
Undergraduate	Electrical and Electronic Engineering	University of Technology/ Baghdad	1996
High School	Applied Sciences	New Baghdad School	1991

WORK EXPERIENCE

Year	Corporation/Institute	Enrollment
2006	Al-Mustansiriyah University, Baghdad, Iraq	Lecturer

PUBLISHMENTS

Conference Papers

1. SHEHAB, M. A., (2017) “Speech Recognition using Wavelet Transform and ELM”, Proceedings of International Conference on Progress in Applied Science 2017 (ICPAS 2017), 04-06 January 2017, Istanbul, Turkey. ISBN: 978-605-9546-02-7.

2. SHEHAB, M. A., KAHRAMAN, N., BİLGİN, G., (2017) “Face recognition using graph extreme learning machine with L21-norm regularization” , In: Electrical and Electronics Engineering (ELECO), 2017 10th International Conference on IEEE; 2017:881-884.

Journal Papers

1. SHEHAB, M. A., KAHRAMAN, N., (2018) “Optimum, Projected and Regularized Extreme Learning Machine Methods with Singular Value Decomposition and L2-Tikhonov Regularization” . Turk J Elec Eng & Comp Sci (2018) 26: 1685 – 1697, © TÜBİTAK, doi:10.3906/elk-1706-60.