ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

PhD THESIS

Hasan YILDIRIM

EXTREME LEARNING MACHINE BASED ON L_1 AND L_2 NORMS

DEPARTMENT OF STATISTICS

ADANA, 2020

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

EXTREME LEARNING MACHINE BASED ON L_1 AND L_2 NORMS

Hasan YILDIRIM

PhD THESIS

DEPARTMENT OF STATISTICS

We certify that the thesis titled above was reviewed and approved for the award of the degree of the Philosophy of Doctorate by the board of jury on 12/06/2020.

.....	.....	.....
Prof. Dr. Mahmude Revan ÖZKALE	Assoc. Prof. Dr. İlker ÜNAL	Assoc. Prof. Dr. Duygu İÇEN
SUPERVISOR	MEMBER	MEMBER
.....	.....	
Assoc. Prof. Dr. Özlem TÜRKŞEN	Assoc. Prof. Dr. Gülesen ÜSTÜNDAĞ ŞİRAY	
MEMBER	MEMBER	

This PhD Thesis is written at the Department of Statistics of Institute of Natural and Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

This dissertation is dedicated to my devoted mother and deceased father.

ÖZ

DOKTORA TEZİ

L_1 AND L_2 NORMA DAYALI AŞIRI ÖĞRENME ALGORİTMASI

Hasan YILDIRIM

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
İSTATİSTİK ANABİLİM DALI

Danışman: Prof. Dr. Mahmude Revan ÖZKALE
Yıl: 2020, Sayfa: 135
Jüri : Prof. Dr. Mahmude Revan ÖZKALE
Doç. Dr. İlker ÜNAL
Doç. Dr. Duygu İÇEN
Doç. Dr. Özlem TÜRKŞEN
Doç. Dr. Gülesen ÜSTÜNDAĞ ŞİRAY

Bir ileri beslemeli sinir ağı olarak aşırı öğrenme algoritması hızlı öğrenme, basitlik ve farklı gerçek yaşam uygulamalarına yönelik yüksek derecede adapte edilebilirlik gibi üstün özellikleri nedeniyle çeşitli disiplinlerde yoğun olarak kullanılmıştır. Bu ilginin temel nedeni, aşırı öğrenme algoritmasının klasik sinir ağlarındaki iteratif parametre tahminini ortadan kaldırmış olmasıdır. Fakat aşırı öğrenme makinesi, yapısal risk minimizasyonunun bir sonucu olarak en iyi model büyüklüğünün belirlenmesi ile ilgili konularda olduğu gibi verinin tahmin edilmesi noktasında da istikrarsızlık ve zayıf genelleştirilebilirlik gibi dezavantajlara sahiptir. Bu çalışmada, ifade edilen eksikleri gidermek ve aşırı öğrenme algoritmasının performansını iyileştirmek amacıyla bazı istatistiksel tahmin ediciler yapay sinir ağları alanında ele alınmış ve çeşitli algoritmalar önerilmiştir. Çalışmanın ilk bölümünde, ridge tahmin ediciye dayanan aşırı öğrenme algoritmasına yönelik uygun yanlılık parametresinin belirlenmesi için farklı seçim kriterleri önerilmiştir. Daha sonra sırasıyla var olan ridge temelli aşırı öğrenme algoritmalarına alternatif olarak Liu, r-k sınıf tahmin edici, Liu ile Lasso regresyon yöntemlerinin birleşimine dayanan farklı öğrenme algoritmaları geliştirilmiştir. Tez kapsamında, R platformunda sıfırdan yazılan algoritmaların uygulamaları çok sayıda gerçek yaşam verisiyle test edilmiş ve performans karşılaştırması detaylı olarak sunulmuştur. Bu önerilen algoritmalar, aşırı öğrenme algoritması ve aşırı öğrenme algoritması tabanlı bazı gelişmiş algoritmalardan daha istikrarlı, genelleştirilebilir ve kompakt olan etkin büzülme ve/veya değişken seçimi performansı ortaya koymaktadır.

Anahtar Kelimeler: Aşırı öğrenme algoritması, cezalı regresyon, ridge, lasso, Liu, temel bileşenler regresyon, elastic net.

ABSTRACT

PhD THESIS

EXTREME LEARNING MACHINE BASED ON L_1 AND L_2 NORMS

Hasan YILDIRIM

DEPARTMENT OF STATISTICS
INSTITUTE OF NATURAL AND APPLIED SCIENCES
UNIVERSITY OF ÇUKUROVA

Supervisor: Prof. Dr. Mahmude Revan ÖZKALE
Year: 2020, Pages: 135
Jury : Prof. Dr. Mahmude Revan ÖZKALE
Assoc. Prof. Dr. İlker ÜNAL
Assoc. Prof. Dr. Duygu İÇEN
Assoc. Prof. Dr. Özlem TÜRKŞEN
Assoc. Prof. Dr. Gülesen ÜSTÜNDAĞ ŞİRAY

Extreme learning machine (ELM) as a type of single-layer feedforward neural network (SLFN) has been widely used in various disciplines due to its superior properties like fast learning, simplicity, high adaptability for different real-life applications. The main reason of this interest is that ELM eliminates the iterative parameter tuning in the conventional neural networks. However, ELM has some drawbacks such as instability and poor generalizability at the point of estimating the data as well as issues related with the determination of the optimal model size as a consequence of structural risk minimization. In this study, in order to deal with these aforementioned drawbacks and improve the performance of ELM, some statistical estimators are discussed in the context of neural networks and various algorithms are proposed. In the first part of the study, we proposed different selection criteria for determinining appropriate tuning parameter for ELM algorithm based on ridge estimator. Later on, Liu estimator, r-k class estimator, a cascade of Liu and Lasso regression estimators are developed as alternatives to the existing ELM algorithms based on ridge estimator. In the scope of this study, the applications of algorithms written from scratch in the R platform are tested via many real-world data sets and the performance comparison are presented in details. These proposed algorithms present effective shrinkage and/or variables selection performance which is more stable, generalizable and compact than ELM and some advanced algorithms based on ELM.

Key Words: Extreme learning machine, regularized regression, ridge, lasso, Liu, principal components regression, elastic net.

GENİŞLETİLMİŞ ÖZET

İçinde bulunduğumuz bilgi çağında, artan veri boyutunun da etkisiyle makine öğrenmesi hiç olmadığı kadar önemli ve kritik bir yere sahip olmuştur. Makine öğrenmesinin ana bileşenlerinden biri olarak yapay sinir ağları ise farklı disiplinlerde ve çeşitli yapılarda gerçek yaşam problemlerinde yoğun olarak kullanılmaktadır. Yüksek derecede adapte edilebilirlik ve esnekliğin yanısıra sunmuş olduğu hızlı öğrenme becerisi nedeniyle bu ilgi artarak devam etmektedir. Denetimli sinir ağlarıarasında bulunan ileri beslemeli ağlar, tahmin, örüntü tanıma ve doğrusal olmayan fonksiyon uydurma gibi alanlarda oldukça başarılı sonuçlar ortaya koymakta ve en yoğun kullanılan ağlar olarak karşımıza çıkmaktadır. Aşırı öğrenme algoritması, tek katmanlı ileri beslemeli ağlar grubunun bir üyesi olup, söz konusu ağların eğitiminde kullanılan geri yayılım algoritmasında olduğu gibi öğrenme parametrelerinin adımsal olarak ayarlanması yerine sabit ve rastgele belirlenmesi durumunda da etkin ve başarılı bir performans gösterebileceği fikrine dayanılarak geliştirilmiştir.

Yapay sinir ağı parametrelerinin sabit olarak belirlenebileceği fikri, aşırı öğrenme algoritmasına inanılmaz derecede hız artışının yanı sıra başarılı eğitim ve test performansı sunmuştur. Bu durumun doğal bir sonucu olarak, makine öğrenmesinin regresyon, sınıflama ve kümeleme gibi hem denetimli hemde denetimsiz öğrenme gibi alt araştırma alanlarında araştırmacılar tarafından yoğun olarak kullanılmıştır. Bununla birlikte bu algoritmanın yapısal risk minimizasyonuna (Vapnik, 1995) dayanması nedeniyle bir takım zayıf yanları bulunmaktadır. Bu zayıflıklara (1) stabil olmama, (2) zayıf genelleştirme performansı, (3) optimal model boyutu sunmaması ve (4) iç ilişki sorununa karşı hassas olması gibi sorunlar

örnek gösterilebilir. Bu tez çalışmasında aşırı öğrenme algoritmasının söz konusu eksikliklerine karşı çözüm üretmek ve beraberinde performansını iyileştirmek amacıyla çeşitli algoritmalar önerilmiştir.

İlk olarak, literatürde önerilen ridge regresyona dayanan aşırı öğrenme algoritmasına yönelik olarak yanlılık parametresinin alternatif seçim yöntemleri üzerinde çalışılmıştır. Yanlılık parametresi ridge regresyon ve ilişkili algoritmaların performansını önemli ölçüde etkileyen bir parametredir ve bu nedenle optimal parametre seçimi önem arz etmektedir. Bu amaçla aşırı öğrenme algoritması bağlamında klasik analitik parametre seçim yöntemlerine alternatif olarak Bayesyen bilgi kriteri, Akaike bilgi kriteri ve çapraz geçerlilik kriteri gibi yöntemler önerilmiş ve bu yöntemlerin parametre değeri ve bunun doğal bir sonucu olarak modellerin performansları üzerinde etkili olduğu gösterilmiştir. Sonuç olarak uygun yöntem seçimi ile birlikte klasik aşırı öğrenme algoritması ve ridge temelli versiyonlarından daha stabil ve genelleştirilebilir sonuçlar elde edilmiştir.

Aşırı öğrenme algoritmasının zayıflıklarının temelinde algoritmanın ana bileşenlerinden olan gizli katman çıktı matrisinin yapısal olarak kötü koşullu olması yatar. Bunun en temel nedenlerinden biri, ilgili matrisinin kolonlarının yüksek derecede ilişkili olması diğer bir deyişle bu matriste çoklu bağlantı sorununun var olmasıdır. Bu amaçla bu soruna çözüm olarak çoklu bağlantıyla baş edebilme noktasında etkin bir yaklaşım olan ridge regresyon önerilmiş ve yoğun olarak kullanılmıştır. Oysa ki ridge regresyona alternatif olarak çoklu bağlantıya çözüm olabilecek tahmin edicilerden biri de Liu tahmin edicidir. Tez kapsamında ikinci olarak yanlılık parametresinin doğrusal yapıda olması gibi avantajı da bulunan Liu tahmin ediciye dayanan bir aşırı öğrenme algoritması (L-ELM) önerilmiştir. Uy-

gun yanlılık parametresinin seçimine yönelik olarak alternatif tahmin yöntemleri tanımlanmış ve her bir parametrenin performans üzerindeki etkileri gözlemlenmiştir. Önerilen algoritmanın en az bir yanlılık parametresinde klasik aşırı öğrenme algoritmasının yanısıra ridge regresyon temelli aşırı öğrenme algoritmasından da daha başarı sonuçlar ortaya koyduğu görülmüştür.

Temel bileşenler regresyon, ölçüm verilerinde değişkenliği yüksek oranda temsil edecek daha küçük boyutlu temel bileşen adı verilen sonradan oluşturulmuş niteliklerle tahmin performansını iyileştiren bir yaklaşımdır. Tez kapsamında üçüncü olarak, bu yaklaşımın aşırı öğrenme algoritmasındaki zayıflıklara çözüm olabileceği fikrinden hareketle literatürde var olan temel bileşen analizine dayanan algoritmalara alternatif olarak temel bileşenler regresyona dayanan bir aşırı öğrenme algoritması (PCR-ELM) adında bir algoritma önerilmiştir. Bunun yanısıra daha önce önerilmiş ridge tahmin ediciye dayanan aşırı öğrenme algoritmasının ve burada bahsedilen PCR-ELM'nin genelleştirilmiş bir formu olan r-k sınıf tahmin edici de aşırı öğrenme algoritması bağlamında yeniden tanımlanmış ve söz konusu yöntemler gerçek yaşam verileri kullanılarak kapsamlı olarak karşılaştırılmıştır. Önerilen algoritmaların aşırı öğrenme algoritmasının stabil olmama, yeni verileri tahmin etme noktasındaki zayıf performansı ve çoklu iç ilişki sorununu iyileştirme açısından tatmin edici sonuçlar ortaya koyduğu görülmüştür.

Aşırı öğrenme algoritmasının önemli zayıflıklarından biri olan optimal model boyutunu belirleme sorununa çözüm amacıyla önemli değişken seçimi yaklaşımlarından olan Lasso tahmin edici incelenmiştir. Lasso tahmin edici ilişkili ve/veya önemsiz değişkenlerin varlığı durumunda etkin bir büzme özelliğinin yanı sıra değişken seçim imkanı da sunan bir tahmin edicidir. Bu tahmin edici tek

başına oldukça başarılı sonuçlar üretmesine karşın, sahip olduğu olduğu bir takım eksikliklere çözüm amacıyla literatürde ridge ve Lasso tahmin edicilerini belirli oranlarda birleştirmeye imkan veren elastic net adından alternatif bir tahmin edici önerilmiştir. Lasso ve elastic net tahmin edicileri aşırı öğrenme algoritmasını belirli bir oranda iyileştirmesine rağmen, zaman zaman makul olmayan zayıf çözümler ortaya koymaktadır. Bu nedenle tez kapsamında dördüncü olarak, iç ilişkide tatmin edici performanslar sergileyen Liu tahmin edici elastic netteki yaklaşıma benzer bir yaklaşımla Lasso tahmin edicinin eğitim aşamasında dikkate alınarak bu iki tahmin ediciye dayanan (LL-ELM) adında yeni bir algoritma önerilmiştir. Önerilen algoritmanın gizli katman çıktı matrisinde uygun değişken (diğer bir deyişle düğüm) seçimiyle birlikte belirli bir oranda büzme işlemi yaparak performansı artırması amaçlanmıştır. Gerçek yaşam verisi ile yapılan karşılaştırmalar sonucu önerilen algoritmanın literatürde tanımlanan ve burada da daha önce değindiğimiz algoritmalardan daha stabil ve genelleştirilebilir sonuçları daha uygun model boyutlarında ve daha düşük norm değerlerine sahip katsayılarla birlikte sunduğu görülmüştür.

Çalışma kapsamında önerilen algoritmaların üstünlüğü regresyon temelli problemlerde gösterilmiş olsa da söz konusu algoritmalar kolaylıkla sınıflama problemlerine adapte edilebilir. Her iki alanın makine öğrenmesinde ve veri-odaklı çalışmalardaki önemi düşünüldüğünde makine öğrenmesi alanında tanımlanmış yaklaşımlara güçlü ve tatmin edici birer alternatif olabileceği düşünülmektedir.

ACKNOWLEDGMENTS

I am grateful to my supervisor Prof. Dr. Mahmude Revan ÖZKALE for her support, constructive criticism, devoted guidance and encouragements during my doctorate education.

I would like to thank the jury members: Assoc. Prof. Dr. Gülesen Üstündağ ŞİRAY, Assoc. Prof. Dr. İlker ÜNAL, Assoc. Prof. Dr. Özlem TÜRKŞEN and Assoc. Prof. Dr. Duygu İÇEN for their valuable suggestions and corrections for improving the thesis.

Special thanks to my family and friends for their supports and tolerance during the entire PhD education. I'm also grateful to my dear parents-in-law. They relieved our's burden significantly.

Finally, I would like to thank my wife for her patience and motivation whenever needed. My dear little son, you always supported me without being aware and I am very proud to be your father.

CONTENTS **PAGE**

ÖZ I

ABSTRACT II

GENİŞLETİLMİŞ ÖZET III

ACKNOWLEDGMENTS VII

CONTENTS VIII

LIST OF TABLES X

LIST OF FIGURES XII

1 INTRODUCTION 1
1.1 Overview of Artificial Neural Networks 1
1.2 Related Studies in Literature 4
1.3 Problem Definition 8
1.4 Aims and Objectives 9
1.5 Contributions to Existing Knowledge 9
1.6 Outline of the Thesis 11

2 OVERVIEW OF MACHINE LEARNING 13
2.1 A Brief of Machine Learning 13
2.2 Extreme Learning Machine 22
2.2.1 The model of ELM 23
2.3 Theoretical Properties of ELM 26
2.3.1 Interpolation Theory 26
2.3.2 Universal Approximation Capability 27
2.3.3 Generalizability Capability 28

3 ESTIMATION METHODS IN LINEAR REGRESSION 31
3.1 Ridge Estimator 33
3.1.1 The Properties of Ridge Estimator 34
3.1.2 The Selection Methods for Ridge Tuning Parameter 36
3.2 Generalized Ridge Estimator 38

3.3 Almost Unbiased Ridge Estimator . . . 38
3.4 Generalized Almost Unbiased Ridge Estimator . . . 39
3.5 Principal Components Regression Estimator . . . 39
3.6 r-k Class Estimator . . . 40
3.7 Liu Estimator . . . 41
3.8 Lasso Estimator . . . 43
3.8.1 The Solution of Lasso Estimator . . . 45
3.9 Elastic Net Estimator . . . 49

4 THE PROPOSED EXTREME LEARNING MACHINE ALGORITHMS . . . 53
4.1 ELM Based on Ridge Estimator . . . 54
4.1.1 Ridge Type Estimators . . . 55
4.1.2 Determination of Appropriate Ridge Tuning Parameter . . . 56
4.2 ELM Based on Liu Estimator . . . 57
4.2.1 Selection Criteria for Liu Tuning Parameter . . . 60
4.3 ELM Based on r-k Class Estimator . . . 62
4.3.1 The ELM algorithm based on principal components regression 62
4.3.2 Proposed Algorithm: ELM Based on the R-K Class Estimator 63
4.3.3 Parameter Selection for the Proposed Algorithm . . . 65
4.4 ELM Based on Liu and Lasso Estimators . . . 68
4.4.1 The Proposed Algorithm: LL-ELM . . . 68
4.4.2 Parameter and Model Selection . . . 70

5 NUMERICAL ANALYSIS . . . 73
5.1 Performance Evaluation of RR-ELM and AUR-ELM . . . 73
5.2 Performance Evaluation of L-ELM . . . 84
5.3 Performance Evaluation of RK-ELM . . . 99
5.4 Performance Evaluation of LL-ELM . . . 109

6 CONCLUSIONS AND DISCUSSIONS . . . 119
6.1 Conclusions . . . 119
6.2 Recommendations and Future Works . . . 122

REFERENCES . . . 123

CURRICULUM VITAE . . . 135

LIST OF TABLES **PAGE**

Table 2.1. Types of activation functions in a neural network 19
Table 2.2. Comparison of conventional learning and ELMs 29
Table 5.1. The benchmark data sets for comparison of ridge-type ELM algorithms 74
Table 5.2. Performance comparison of ridge-type ELM algorithms 76
Table 5.3. The benchmark data sets for Liu-based ELM algorithm 84
Table 5.4. The performance results of ELM and L-ELM algorithms 87
Table 5.5. The post hoc results on the testing RMSE to show the statistical significance for each data set 92
Table 5.6. The properties of benchmark data sets 99
Table 5.7. Performance comparison between RK-ELM and other algorithms 102
Table 5.8. The benchmark data sets for Liu-Lasso based algorithm 109
Table 5.9. The values of tuning parameters 111
Table 5.10. Comparison of all algorithms in terms of training and testing RMSE 113
Table 5.11. Norm comparison of Liu-ELM and LL-ELM algorithms 116

LIST OF FIGURES **PAGE**

Figure 2.1. A general structure of a typical neural network (Hagan et al., 2014) 17

Figure 5.1. The testing errors for Auto mpg data set 80
Figure 5.2. The testing errors for Boston housing data set 81
Figure 5.3. The testing errors for Abalone data set 82
Figure 5.4. Comparison of models in terms of testing RMSE 83
Figure 5.5. Comparison of models in terms of standard deviation of testing RMSE 83
Figure 5.6. The flow chart of experimental process on training and testing phases for L-ELM 85
Figure 5.7. The testing errors of ELM, RR-ELM, AUR-ELM, OP-ELM and L-ELM for $\hat{d}_2$ algorithms for Boston Housing data set 94
Figure 5.8. The testing errors of ELM, RR-ELM, AUR-ELM, OP-ELM and L-ELM for $\hat{d}_2$ algorithms for Auto Mpg data set 95
Figure 5.9. The testing errors of ELM, RR-ELM, AUR-ELM, OP-ELM and L-ELM for $\hat{d}_1$ algorithms for House Census data set 96
Figure 5.10. The testing errors of ELM, RR-ELM, AUR-ELM, OP-ELM and L-ELM for $\hat{d}_1$ algorithms for Bank Domains data set 97
Figure 5.11. The L-ELM performance change for each data set based on sequential tuning parameter in the range of $[0,1]$ 98
Figure 5.12. The general architecture of the PCR-ELM and RK-ELM algorithms 100
Figure 5.13. The comparison based on testing RMSE 103
Figure 5.14. The comparison based on standard deviations of testing RMSE . 103

Figure 5.15.The residuals based on testing RMSE for Auto Mpg ($m = 258$) . 105
Figure 5.16.The residuals based on testing RMSE of Boston ($m = 114$) . . . 106
Figure 5.17.The residuals based on testing RMSE of Body Fat ($m = 76$) . . . 107
Figure 5.18.The comparison of RK-ELM and RR-ELM for each data set for arbitrary k values from 0 to 1 108
Figure 5.19.The performance comparison of Liu-ELM and LL-ELM based on random tuning parameter 117

1. INTRODUCTION

1.1 Overview of Artificial Neural Networks

Due to the abundance of data size produced in this information era, machine learning has gained special importance to process it effectively and usefully. The main goals of machine learning are to develop new algorithms and improve the existing algorithms to transform real-world data into beneficial and intelligent actions. Various types of algorithms have been employed to perform different tasks like regression, classification, clustering, and association. One of the most important components of the machine learning field is artificial neural networks (ANNs). In the early stages of ANNs, McCulloch and Pitts (1943) put forward a mathematical model that highly simplifies the model of a neuron. This contribution to the literature opened a door of ANNs researchers.

ANNs have been extensively used in many fields due to their advantages: (i) to approximate any complex nonlinear mapping by using inputs directly, (ii) to be able to give solutions for some difficult problems that can not be dealt with the conventional analysis techniques. There exist many types of neural networks. According to the direction of the neural network internal information transfer, the neural networks can be divided into two categories (Ding et al., 2015): feedforward neural networks and feedback type neural networks. Feedforward neural networks are the most popular and are the first and simplest types of ANNs (Schmidhuber, 2015).

The feedforward neural networks have been extensively studied and used in various areas by virtue of the ability to approximate complex nonlinear mappings directly from the input samples. Hornik (1991) proved that if the activation func-

tion, which is one of the main components of any neural networks, is continuous, bounded and nonconstant, then continuous mappings can be approximated with respect to measuring by using feedforward neural networks over compact input data sets. Leshno et al. (1993) showed that a feedforward neural network can approximate any continuous function even if it has a nonlinear activation function. Due to the obligation of using finite data set in real applications, the ability of approximation in these situations is also important. Huang and Babri (1998) precisely proved that a single layer feedforward neural network (SLFN) with any arbitrary and with at most N hidden nodes can exactly approximate any arbitrary N samples with zero error.

Although the feedforward neural networks have some abilities: (i) to approximate any complex nonlinear mappings; and (ii) to be able to overcome some problems which can not be solved via the conventional parametric techniques, it should be noted that some parameters such as input weights and hidden layer biases have to be adjusted by using learning algorithms to get desirable results (Huang et al., 2006). This obligation brings some lacks such as slower learning ability in most practical applications. In addition, sub-optimal solutions may be obtained in some gradient descent-based learning algorithms like back-propagation. It may easily converge local minima in most problems. To summarize, feedforward neural networks face some challenging issues because of learning algorithm deficiencies such as (i) slow learning process, (ii) providing sub-optimal solutions or easily converge local minima and (iii) the risk of over-fitting (Huang et al., 2006). In order to deal with these issues, many studies have been done in the literature. Hagan and Menhaj (1994) presented the Marquardt algorithm and incorporated it with backpropagation

algorithm to obtain a more compact and faster network. Li et al. (2005) proposed the fast recursive algorithm as an alternative to the classical least squares and estimated the model parameters without matrix decomposition. The performance of the fast recursive algorithm is found as much more efficient in terms of computation cost and stability of a neural network based on the classical least-squares method. Wilamowski and Hao (2010) proposed a method based on the training of neural networks by using forward-only computation, unlike the backpropagation algorithm. However, the studies on the improvement of the backpropagation algorithms could not give a definite solution to these problems. Huang et al. (2015) and Ding et al. (2015) have been also pointed out the problems of the feedforward neural networks, especially neural networks based on backpropagation learning algorithm.

As a solution to these mentioned above issues, a novel and effective algorithm called Extreme Learning Machine (ELM) has been proposed by Huang et al. (2006). The idea behind this algorithm is to select the input weights and hidden layer biases fixed. As long as the activation functions in the hidden layer are infinitely differentiable, these parameters can be randomly assigned without tuning. In this way, the problem reduces to a linear system. Then, the output weights can be calculated by solving this system analytically. This property brings some superiorities to ELM including (i) fast learning speed, (ii) good generalization performance, (iii) universal approximation capability (iv) easy implementation to any system, (v) functionality on common machine learning fields such as regression, classification, clustering, (vi) minimal requirement to human intervention, etc.

1.2 Related Studies in Literature

In this section, some major studies related to the objectives of this thesis study which includes basic ELM algorithm and its extensions are reviewed. These studies may be divided into three categories according to their objectives as: (i) extensions for improving the stability and compactness of ELM, (ii) extensions based on reducing the dimensions of ELM and (iii) applications and reviews.

The majority of studies investigates the structural properties of the hidden layer output matrix of ELM. ELM with superior properties has drawn attraction to many researchers and studied to be improved in terms of the aforementioned problems. Deng et al. (2009) proposed a novel algorithm called regularized ELM based on structural risk minimization and weighted least squares approach to handle the drawbacks of ELM such as heteroscedasticity, outliers, and over-fitting. Feng et al. (2009) proposed error minimized ELM (EM-ELM) based on the calculation of the output weights in the hidden layer incrementally and determining the appropriate number of hidden layer neurons to obtain a faster network than ELM. Wang et al. (2011) proposed EELM guarantees full column rank of hidden layer output matrix in theory and tested it with ELM in terms of learning capability and robustness of the network. Yuan et al. (2011) studied ELM from the theoretical aspect and suggested some useful works including reformulation the basic estimation model, ensure the full column rank of hidden layer output matrix, etc., to make sure for obtaining optimal approximation solution of classical ELM. Lu et al. (2015) studied the performance of the calculation of Moore-Penrose inverse matrices in ELM and proposed different algorithms based on Cholesky factorization of a singular matrix, QR factorization with generalized inverse and conjugate Gram-Schmidt processes.

The efforts for improving ELM were effective but not enough to be sure about obtaining the optimal solution of the estimation problem.

One of the main reasons for this situation is the problem called the ill-conditioning in the hidden layer output matrix. The main reason of ill-conditioning in a matrix is to have some highly correlated variables in the data sets. This situation is also known as multicollinearity in the regression context and a common problem in practical applications. Researchers have been widely studied on this problem in linear regression field and developed alternative approaches to reduce the effect of multicollinearity. Ridge estimator (Hoerl and Kennard, 1970) is the most popular and well-known method among these approaches. When multicollinearity exists in the data set, the least-squares estimates become unstable and provide poor results in terms of generalization performance. Even if ridge estimator improves least squares estimators by reducing the mean squared error (MSE) value, it may provide too biased results which is not reasonable. That is, the model based on a biased estimator may overfit or underfit the data. Therefore, Singh et al. (1986) proposed the almost unbiased ridge (AUR) estimator based on the jackknife procedure which was effective in reducing the bias term. Recently, many algorithms based on these two estimators have been proposed to enhance ELM in terms of stability and generalization performance. It should be noted that the ridge estimator may be also called in Tikhonov or L_2 norm regularization in the context of neural networks and it has been studied widely. Firstly, the optimally pruned extreme learning machine (OP-ELM) was proposed by Miche et al. (2010) to solve the problem of having some irrelevant or highly correlated variables. In OP-ELM, the computation of the pruning criterion uses a leave-one-out criterion based on Allen's (1974) PRESS statistic. However,

if the output matrix of the hidden layer is not full rank, caused by multicollinearity, then drawbacks will exist in the computation of the PRESS statistic. Therefore, Miche et al. (2011) proposed a new algorithm called double-regularized ELM using LARS (Efron et al., 2004) and Tikhonov regularization (TROP-ELM) which is a modification of OP-ELM. It is based on a cascade of two regularization penalties: L_1 and L_2 norm-based penalties. Martínez et al. (2011) proposed a pruning ELM algorithm based on ridge, lasso (Tibshirani, 1996) and elastic net (Zou and Hastie, 2005) estimators to select a suitable number of hidden layer nodes. Huang et al. (2012) proposed a general framework on the usage of ridge estimator based on regularization in ELM and presented a comparative study in terms of efficiency, generalization performance and sensitivity to the dimension of feature space. Li and Niu (2013) proposed an ELM algorithm based on ridge estimator (RR-ELM) and almost unbiased ridge estimator (AUR-ELM) in a regular form based on calculating the output weights by ridge and AUR estimators instead of least squares, respectively. Zhang et al. (2013) proposed an ELM algorithm based on error minimization (EM-ELM) and presented a comparative study by using the basic ELM and RR-ELM on three data sets where two of them are regression task and one for classification task. The results of this study show that RR-ELM has better stability and generalization performance than ELM on both regression and classification tasks. Fakhr et al. (2015) have carried a comparative study based on L_1 norm regularization to achieve the sparsity in the hidden layer output weights and presented the performance results between ELM based on L_1 norm, ELM based on L_2 norm and basic ELM algorithms. He et al. (2017) developed an $L_{1/2}$ norm regularized pruning ensemble model of ELM to provide the optimal solution of parameter estimation and to determine the appropri-

ate number of hidden layer nodes. Chen et al. (2017) proposed a unified model for robust regularized ELM regression using iteratively reweighted least-squares (IRLS) and used four different loss functions and two regularization methods as a solution for robustness and overfitting, respectively.

On the other hand, principal components analysis (PCA) is a statistical technique which may be functional in the context of ELM and based on orthogonal transformation of a large number of correlated variables into a much smaller set of uncorrelated variables. PCA reduces the dimension of the data set by some creating fewer principal components (PCs) referring to weighted linear forms of variables. Owing to its orthogonality property, it is quite functional and successful to handle the multicollinearity problem in linear models. Therefore, Kim et al. (2007) employed PCA on sample data and applied the classical ELM algorithms to obtain covariates, i.e. PCs to classify arrhythmias. Singh et al. (2012) proposed novel ELM algorithms with various kernels using the covariates obtaining by PCA and linear discriminant analysis for multi-class protein fold recognition problem. Li et al. (2012) applied PCA on the Thyroid disease data set, determined the optimal number of hidden layer neurons by carrying on the cross-validation process and got the results by using the best feature space and optimal parameters. Zhao et al. (2013) applied PCA to get the most distinct feature space of wine data set and applied ELM on these features to classify wine types effectively. Castano et al. (2013) used PCA to determine the number of hidden neurons via the explained variance ratio threshold criterion and applied the ELM algorithm by considering the identified number. Zhang et al. (2016) has used P-ELM (a.k.a PCA-ELM) algorithm based on PCA on the hidden layer output matrix to make it a full column rank and to get faster training results.

Zhang et al. (2016) aim to reduce the dimension of the hidden layer output matrix and get an effective model to estimate the hot metal temperature in the blast furnace.

Finally, the applications of ELM have been widely studied and used in industrial applications and expert systems like medical/biomedical (Barea et al., 2012; You et al., 2013), computer vision (Choi et al., 2012; Nian et al., 2013), image/video processing (Suresh et al., 2009; Cao et al., 2013; Bazi et al., 2014), text classification (Zheng et al., 2013; Poria et al., 2014), system modeling and prediction (Wong & Guo, 2010; Chen et al., 2011; Zhao et al., 2012; Zhao et al., 2013; Yan & Wang, 2014), control and robotics (Rong et al., 2011), chemical process (Wang et al., 2013), fault detection and diagnosis (Creech & Jiang, 2012), time series analysis (Butcher et al., 2013), remote sensing (Tang et al., 2015). For some comprehensive review studies on ELM theory and applications, the readers are referred to Huang et al. (2011), Huang et al. (2015) and Deng et al. (2015).

1.3 Problem Definition

Although ELM produces successful results in many real-world applications, it has several drawbacks caused by its learning nature. As a result of the structural risk minimization approach (Vapnik, 1995), ELM may provide poor results in terms of stability, generalization performance and sparsity perspectives. In some situations, even ELM presents efficient convergence in theory, its results may be overfitted or under fitted compared to what it should be in reality. Besides, the performance of ELM significantly depends on the number of hidden layer nodes. The selection of optimal node number is a complicated issue because of the possibility of over-fitting or under-fitting (Martinez et al., 2011).

In order to deal with the drawbacks of ELM, alternative approaches are nec-

essary. Instead of the ordinary least squares method which suffers because of the structural risk minimization approach, some statistical estimators can be considered as powerful tools to improve the performance of the ELM algorithm.

1.4 Aims and Objectives

The focus of this thesis study is to develop efficient and novel algorithms based on ELM which provide reasonable performances in terms of stability, generalizability, and sparsity and can deal with the drawbacks of ELM. The following objectives are investigated:

- to examine the effect of different selection criteria for determining the biasing parameters in ELM algorithms based on ridge-type estimators,
- to propose an alternative biased estimator to ridge in the context of ELM,
- to present a unified framework for ELM algorithms based on both principal components regression and ridge regression,
- to define a new and novel algorithm that is efficient in terms of regularization and shrinkage capability,
- to carry out a comprehensive comparative study to investigate the performance of related ELM algorithms.

1.5 Contributions to Existing Knowledge

Firstly, the effect of the selection method of tuning parameter for ridge-based ELM algorithms including classical ridge regression and AUR has been investigated. Instead of analytical methods, Yıldırım and Özkale (2019) proposed three different

criteria including Bayesian information criterion (BIC) proposed by Schwarz (1978), Akaike information criterion (AIC) proposed by Akaike (1973, 1998) and cross-validation error (CV) suggested by Hastie et al. (2009) in the context of ELM . By considering these criteria in the training process, more accurate and stable model results have been obtained. Besides, it has been clearly seen that the type of selection criterion is significantly effective on the performance of both stability and generalizability in models based on the ELM algorithm.

Secondly, a new algorithm based on Liu estimator has been proposed (Yıldırım and Özkale, 2020) instead of ridge regression to overcome the multicollinearity problem in real-world applications. Liu estimator has some advantages compared to ridge regression like (i) linear form of tuning parameter and (ii) learning speed. Furthermore, appropriate selection methods for Liu tuning parameters have been presented. A comparative study has been carried out. The results show that the proposed algorithms for at least one Liu tuning parameter may overcome ridge-based ELM algorithms and the basic ELM algorithm and produce more stable and generalizable results.

Thirdly, a unified algorithm called r-k class ELM has been proposed to present the virtues of ridge and principal components regression together. As a contribution of ridge-type ELM in the literature, the ELM algorithm based on principal components regression has been redefined and the performance of each algorithm also including r-k class ELM has been compared on several benchmark data sets. As a result of the comparison, the proposed algorithms are better than competitor algorithms and more effective in dealing with instability and generalizability drawbacks of ELM.

Finally, a novel regularization and variable selection algorithm called Liu-Lasso extreme learning machine (LL-ELM) has been proposed in order to deal with the ELM's drawbacks like instability, poor generalizability and under or overfitting due to the selection of inappropriate hidden layer nodes. The proposed algorithm is compared with the conventional ELM and its variants including Liu-ELM, ELM based on L_1 norm, ELM based on L_2 norm, ELM based on the elastic net. The convenient selection method for the determination of tuning parameters for each algorithm has been used in comparisons. The results show that LL-ELM can overperform its competitors in terms of learning and generalization performance. The shrinkage level of LL-ELM is also notable.

1.6 Outline of the Thesis

The remainder of this thesis is organized as follows:

A brief overview of machine learning and the structure details of ELM algorithm are given in Section 2. In Section 3, the statistical estimators in linear regression are explained. The proposed algorithms to improve classical ELM and its variants for dealing with its drawbacks are presented in Section 4. Some mathematical and conceptual definitions related to the proposed algorithms are shared in detail. In Section 5, the details of all experimental settings and the performance results of comparisons based on each proposed algorithms are given. Finally, the last section of this thesis study includes some detailed conclusions, remark, discussions, and suggestions which cover all studies.

2. OVERVIEW OF MACHINE LEARNING

2.1 A Brief of Machine Learning

Learning is ”a process that leads to change, which occurs as a result of experience and increases the potential for improved performance and future learning” (Ambrose et al., 2010). Transferring the learning outcomes like experiences, knowledge or developments from generation to generation is one of the main goals for any human-beings and learning is the key concept of human intelligence. From the scientific perspective, advances in scientific studies are thanks to the curiosity of human learning. The advances are based on some experimental works which aim to find some evidence by using some measurements for a particular objective. Basically, human beings learn from data. Regardless of the type of discipline, data have a particular importance and the main focus is to obtain functional inferences from data and to reduce the uncertainty of any objective.

Due to the increase of data size in the big data era, some automated mechanisms have been needed to deal with the deluge of data. The accumulation of scientific knowledge and technology make possible to infer more easier. As a solution to this need, the machine learning field has arisen. Machine learning is defined as an automated process that extracts patterns and then uses the uncovered patterns to predict the future or obtain some useful insight for any kind of decision making under uncertainty (Murphy, 2012; Theodoridis, 2015).

The learning types of machine learning vary according to the application problem. Machine learning is generally divided into two categories: (i) supervised learning and (ii) unsupervised learning. In the former one, learning is basically defined as mapping from inputs (i.e. feature, attributes or covariates) to outputs (i.e.

response variable), given a labeled set of input-output pairs which generally shown as $\mathbf{D} = \{(x_i, t_i)\}_{i=1}^{N}$ where $\mathbf{D}$ refers to the training set and N is the sample size of training set. In supervised learning, the scale of the response variable (t) is determinative of the type of methods for a particular problem. The general family of methods for a categorical or nominal response variable is called as classification or pattern recognition. Similarly, the type of problem is known as regression or ordinal regression for a numeric-valued and ordinal response variables, respectively. In unsupervised learning, the learning process is carried out without some known outputs. The main goal is to find some useful patterns between inputs $\left(\mathbf{D} = \{(x_i)\}_{i=1}^{N}\right)$ by observing the relationships and similarities. This field of machine learning is also called as descriptive, explanatory learning or knowledge discovery. Actually, reinforcement learning is seen as the third type of learning but used less common in practical applications. In reinforcement learning, learning is based on taking some actions providing a maximum reward for a particular situation. Unlike supervised learning, the learning is conducted via some feedbacks which is effective in updating the route to maximize the reward among various alternative solutions.

Machine learning has wide application areas including medical diagnosis, marketing, business analytics, telecommunication, bioinformatics, biostatistics, insurance, economics, natural language processing, image, speech, text and handwriting recognition, computer vision, banking, agriculture, statistics, anatomy, genetics, cheminformatics. In a particular machine learning problem, some models are created based on the research objectives. The type of model is problem-dependent. Among various models, ANNs occupy an important position due to some properties like speed, functionality, adaptability and providing solutions for complex problems.

The fields of ANNs have developed significantly until today. The beginning of the neural networks started with the study of McCulloch and Pitts (1943) who presented that a networks with artificial neurons can compute any arithmetic or logical function. This study is based on modelling a simple neural network by using electric circuits to understand how a neuron works and seen as the origin of this field. (Hagan et al., 2014). Hebb (1949) claimed that behaviors can be explained by neurons in his famous book named as "The organization of behavior". The basics of this claim were given by Hebb as: "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." This idea provided the first insights about the mechanism in biological neurons and was seen as the origin of the Hebbian learning which is one of the most well-known learning rules. Rosenblatt (1958) introduced the first application of artificial neural networks called perceptron network which was successful on the pattern recognition. Perceptron network has its own learning rule and has been widely used for a certain type of problems. Widrow and Hoff (1960) proposed Adaline and Madaline learning rules as alternatives to the perceptron rule. Similar to perceptron learning, the proposed learning rule is called as Widrow-Hoff learning and is applicable for various problems. After the proposing the perceptron and Widrow-Hoff learning rules, Minsky and Papert (1969) showed the lack of capabilities about these rules on different learning problems and proposed new algorithms which were not widely used by the researchers of ANNs. This study adversely and dramatically affected the existing interest to the field. Kohonen (1972) and Anderson (1972) proposed separately a new type of neural network which has the ability

of acting like memory. This networks has been known as Kohonen networks in today. Grossberg (1976) studied on the self-organizing networks which included the characterization of the usage of non-linear mathematics to understand some particular function of brain, for example vision or memory. His studies were based on different disciplines like mathematics, psychology and neurophysiology (Hagan et al., 2014). Hopfield (1982) introduced the recurrent networks which affects the interest to the field dramatically. The underlying idea behing the recurrent networks was the possibility of the two-ways connections between neurons unlike the popular one-way. This idea made the developments memory-based associative networks possible. Reilly et al. (1982) proposed a multi-layer hybrid neural network with each layer uses different strategy for a different learning problem. The most important study of ANNs fiels is seen the study of Rumelhart et al. (1988). In this study, the backpropagation learning rule which carries the Widrow-Hoff learning rule to the multi layer networks (Hagan et al., 2014). While hybrid networks uses two layers, networks based on backpropagation can use more layers. More layers improves the capability of network to provide better solutions for complex, non-linear and out of common problems. The essence of the backpropagation learning is to update the parameters, which are also the main components of a network, until the desired convergence is reached.

A typical neural network consists of some components including inputs, weights, biases, activation function, layers and outputs. The inputs correspond to the values of each attributes measures for a particular problem. Each input values is represented with a weight which connects the inputs and the corresponding neuron. The biases are similar to the weights and can be seen as offset term. These

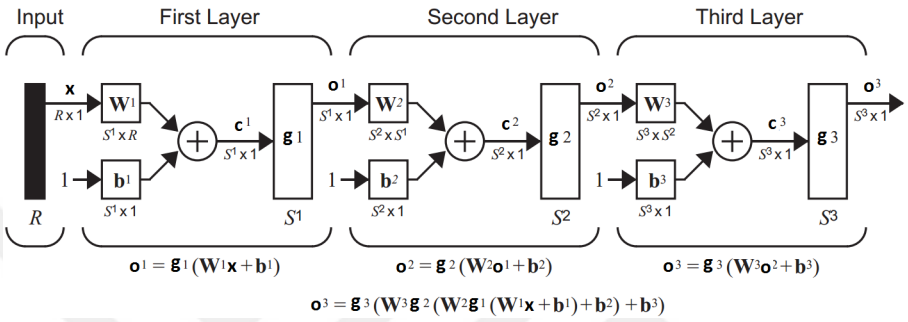

Figure 2.1. A general structure of a typical neural network (Hagan et al., 2014)

terms have effects to lead the network to the better solution but can be omitted by the researcher. The bias values have been weighted with a same and constant which is generally taken as 1. Activation function is a transfer function which provides the net output of a neuron by using net input which represents the multiplication of the weights and the inputs with adding the bias values. There have been proposed a variety of activation functions which can be linear or non-linear. According to the selection of activation function, the corresponding function takes the net input and produces a transfered value to the next neuron. A layer is an element including more than one neurons which operate in parallel. The number of neurons and layers are variable depending on the complexity of problem. More layers including more neurons may be needed for more complex problems. The outputs correspond to the outputs estimated by network. The estimated output values are compared with the actual outputs and the performance of networks can be measured by using the difference between them. A general structure with common notations for a multilayer neural network with three hidden layers (Hagan et al., 2014) is given in Figure 2.1.

In Figure 2.1, $\mathbf{x}$ corresponds to the inputs vector belongs to the each units, $\mathbf{W}$

is an $S \times R$ matrix whose elements are the weights between the inputs and the hidden layers. S is the number of hidden layer neurons and R is the number of the inputs referring to the number of attributes. $\mathbf{b}$ is the $S \times 1$ vector including the bias values. $+$ symbol represents the type of combination formula which is generally selected as the summation to obtain the net weighted inputs. $\mathbf{c}$ is the net input values with the size of S. g corresponds to the activation function (i.e. transfer function) to tranform the net input values to the net output values of the corresponding hidden layer. $\mathbf{o}$ is the aforementioned net output values given by g function. In the representations, such as $\mathbf{W}^1$ corresponds the weights of 1st hidden layer. For other layers and components, the process and representations is similar to the first layer. An output of a hidden layer is presented as the input of the next hidden layer. In the last phase, the final output is calculated by using the net output of the last hidden layer (Hagan et al., 2014). The effects of the former layers calculations are reflected into the last layer calculation. The selection of summation formula, the type of activation function, the number of hidden layers and the number of neuron for each layer change to type and complexity of the problem. One of the most important element of a neural network is the activation function. There have been proposed many activation functions in the literature. A summary of the most used functions (Hagan et al., 2014) is given in Table 2.1.

In addition to the details of the components of a neural networks, it should be noted that the structure of the inputs is important on the performance of ANNs models. In practical applications, some approaches have been prefered to obtain better results. These approaches can be grouped as: (i) Normalization, (ii) Centering and (iii) Standardizing. Normalization is to transform the inputs values such that

each variable has range like $[0,1]$ or $[-1,1]$. Centering is to tranform the inputs values such that each variables has zero mean. Standardization is to tranform the inputs values such that each variables has zero-mean and unit-variance. The effect of the type of this pre-processing is to obtain more robust models. Especially, the outliers, correlated attributes or noises can adversely affect to the performance of networks. That's why, an appropriate selection of scaling methods can present more suitable inputs for a successful learning performance.

Table 2.1. Types of activation functions in a neural network

Activation function	Calculation of the output	Icon
Hard Limit	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \begin{array}{ll} 0, & \mathbf{w}.\mathbf{x}+b < 0 \\ 1, & otherwise \end{array} \right\}$	
Symmetrical Hard Limit	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \begin{array}{ll} -1, & \mathbf{w}.\mathbf{x}+b < 0 \\ +1, & otherwise \end{array} \right\}$	
Competitive	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \begin{array}{ll} -1, & for\ neuron\ with\ \max\ \mathbf{w}.\mathbf{x}+b\ (input) \\ -1, & otherwise \end{array} \right\}$	
Saturating Linear	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \begin{array}{ll} 0, & \mathbf{w}.\mathbf{x}+b < 0 \\ \mathbf{w}.\mathbf{x}+b, & 0 \le \mathbf{w}.\mathbf{x}+b \le 1 \\ 1, & \mathbf{w}.\mathbf{x}+b > 1 \end{array} \right\}$	
Symmetric Saturating Linear	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \begin{array}{ll} -1, & \mathbf{w}.\mathbf{x}+b < -1 \\ \mathbf{w}.\mathbf{x}+b, & -1 \le \mathbf{w}.\mathbf{x}+b \le 1 \\ 1, & \mathbf{w}.\mathbf{x}+b > 1 \end{array} \right\}$	
Positive Linear	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \begin{array}{ll} 0, & \mathbf{w}.\mathbf{x}+b < 0 \\ \mathbf{w}.\mathbf{x}+b, & otherwise \end{array} \right\}$	
Linear	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \mathbf{w}.\mathbf{x}+b \right\}$	
Log-Sigmoid	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \frac{1}{1+\exp(-(\mathbf{w}.\mathbf{x}+b))} \right\}$	
Hyperbolic Tangent	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \frac{1-\exp(-(\mathbf{w}.\mathbf{x}+b))}{1+\exp(-(\mathbf{w}.\mathbf{x}+b))} \right\}$	
Gaussian	$g(\mathbf{w},b,\mathbf{x}) = \left\{ \exp\left(-b\|\mathbf{x}-\mathbf{w}\|\right) \right\}$	

There have been important advances in the field of ANNs until today. The numerious studies have been carried out from both theoretical and application perspectives. With these advancements, new concepts, network structures and learning rules have been proposed. The learning rules are the main focus among these advancements because of the effectiveness on the performance of network. Different learning rules can be applicable for different types of problems to get the fastest and best results. The majority of learning rules are based on the variants of gradient descent methods which actually use the backpropagation approach. This process mainly includes to use the gradient and based on updating the parameters of networks according to these gradients. The algorithms (i.e. methods) based on backpropagation can be generally classified into three categories: (i) Steepest descent algorithm, (ii) Newton's method and (iii) Conjugate gradient algorithm. All of these algorithms have been widely used in real-world applications. While the steepest descent algorithm is based on the the first-order Taylor series expansion, Newton's method uses the second-order Taylor series. In other words, the underlying idea behind Newton's method is to converge the target function by using quadratic approximation unlike the linear approximation in steepest descent algorithm. This means that it minimizes a quadratic function exactly in a finite number of iterations. When the number of parameters are large, the calculation and storage of the second derivatives will be costly. Conjugate gradient algorithm could be beneficial in order to deal with this drawback by requiring only first derivatives but still have quadratic termination. Conjugate gradient algorithm uses a set of the best directions among many possibilities to guarantee quadratic termination (Hagan et al., 2014). For more details in terms of the theoretical and application properties of these algorithms, the

readers are refered to the study of Hagan et al. (2014).

Consequently, the algorithms based on the backpropagation have been widely used for various problems. However, there have some drawbacks to the usage of these learning algorithms. These drawbacks can be listed as follows (Huang et al., 2006; Yuan et al., 2011; Huang et al., 2015):

(1) The convergence may be too slow or unstable depending on the value of learning rate of the network. While a small learning rate can cause too slow convergence, large values of learning rate may provide unstable results.

(2) The possibility of stopping at local minima points instead of providing a global optimum. This property is not reasonable if there is better points in terms of predictive performance.

(3) Due to the high possibility of overfitting problem, the networks should be carefully tuned by using some stopping criteria. Otherwise, the network could provide unsatisfactory results from the perspective of generalization performance.

(4) These algorithm are time-consuming because of the necessity to tune more parameters effectively.

As mentioned before, the classical neural networks have various parameters which should be tuned carefully. The most important parameters are generally the learning rate and the momentum value. The learning rate determines the amount of changes in weights. It is generally in the range of $[0, 1]$. While too small learning rate produces unstable results, too large value of it can cause too slow convergence. The value is proportional with the complexity of the problem. More complex problem needs large learning rate. The momentum value controls the oscillation of the network. Similar to the learning rate, it take values between 0 and 1. An optimum mo-

mentum value provides more balanced learning. The large momentum value mean that the previous weight can be considered much more when compared to a small value. Small learning rate with large momentum can be used for a complex problem. These parameters are effective on the learning speed of the networks not the performance. That's why, the learning algorithms need more workloads because of the determination of the tuning parameters. Besides, ANNs, particularly feedforward neural networks may produce some sub-optimal solutions. As mentioned in the Section 1, the ELM algorithm is proposed to deal with the issues of single layer feedforward neural networks (SLFNs). In the next section, the details of the ELM algorithm are given.

2.2 Extreme Learning Machine

ELM is proposed as a special and fast SLFN by Huang et al. (2006) and provided both fast learning speed and good generalization performance. It basically assumes that the weights and biases between input and hidden layer nodes can be generated randomly. Therefore the estimation problem is reduced to a solution of a linear equations system that can be solved analytically without any tuning and iterations. In finding the output layer weights of such a linear equations system, the least-squares method is generally used. This simple logic moves ELM further in terms of speed. The training speed is improved and extremely fast compared with other gradient descent-based methods. ELM algorithm can be used for regression and classification context. Some comprehensive surveys for ELM and its application are given in Huang et al. (2011, 2012, 2015) and Ding et al. (2015).

2.2.1 The model of ELM

Consider a group of samples (x_j, t_j), where $x_j = (x_{j1}, x_{j2}, ..., x_{jn})^T \in \mathbb{R}^n$ and $t_j = (t_{j1}, t_{j2}, ..., t_{jm})^T \in \mathbb{R}^m$ to be used in a problem. ELM algorithm as an SLFN claims that it can converge with zero error to the actual values of N samples. In other words, for a given activation function (g) and a number of hidden layer neurons $\left(\tilde{N}\right)$, the mathematical model of estimation based on a SLFN can be written as

$$\mathbf{t}_j = \sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i g\left(\mathbf{w}_i.\mathbf{x}_j + b_i\right), \ j = 1, 2, ..., N. \tag{2.1}$$

where $\mathbf{t}_j^T$ is the jth output of neural network, $\mathbf{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ and b_i are the learning parameters between the ith hidden node and the input nodes. $\boldsymbol{\beta}_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ is the weight vector linking the ith hidden node and the output nodes. $\mathbf{w}_i.\mathbf{x}_j$ corresponds to the inner product between $\mathbf{w}_i$ and $\mathbf{x}_j$ (Huang et al., 2004, 2006). Eq. (2.1) refers to the general structure of ELM and the values of n and m correspond to the number of neurons in input layer and the number of neurons in the output layer, respectively. Generally, n is equal to the number of explanatory variables and m is equal to the number of response variables which is commonly taken as one in most practical applications. Eq. (2.1) can be written more compactly as

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{2.2}$$

where

$$\mathbf{H} = \begin{bmatrix} g\left(\mathbf{w}_1.\mathbf{x}_1 + b_1\right) & ... & g\left(\mathbf{w}_{\tilde{N}}.\mathbf{x}_1 + b_{\tilde{N}}\right) \\ \vdots & ... & \vdots \\ g\left(\mathbf{w}_1.\mathbf{x}_N + b_1\right) & ... & g\left(\mathbf{w}_{\tilde{N}}.\mathbf{x}_N + b_{\tilde{N}}\right) \end{bmatrix}_{N \times \tilde{N}}$$

is the output matrix of hidden layer in the neural network,

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N}\times m}$$

corresponds to the weights,

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N\times m} = \begin{bmatrix} t_{11} & \dots & t_{1m} \\ \vdots & & \vdots \\ t_{N1} & \dots & t_{Nm} \end{bmatrix}_{N\times m}$$

is the output values of neural network. The most common activation functions which can also be used in ELM are given in Table 2.1. In addition to these functions, multiquadratic and cosine functions are another types of activations functions and widely used for training the ELM algorithm. These functions are defined as $g(\mathbf{w},b,\mathbf{x}) = \left(\|\mathbf{x}-\mathbf{w}\|+b^2\right)^{1/2}$ and $g(\mathbf{w},b,\mathbf{x}) = \cos(\mathbf{w}.\mathbf{x}+b)$, respectively (Huang et al., 2015). By random selection of the learning parameters, the weights (β) can be obtained via a classical least-squares approach. Therefore, the estimation of $\boldsymbol{\beta}$ in ELM is equivalent to obtain the solution of the linear system expressed as

$$\left\|\mathbf{H}\hat{\boldsymbol{\beta}} - \mathbf{T}\right\| = \arg\min_{\boldsymbol{\beta}} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|. \tag{2.3}$$

The solution of Eq. (2.3) minimizes the following summation

$$\sum_{j=1}^{N}\left(\sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i g(\mathbf{w}_i.\mathbf{x}_j + b_i) - \mathbf{t}_j\right)^2. \tag{2.4}$$

The least-squares solution of $\boldsymbol{\beta}$ is calculated as $\widehat{\boldsymbol{\beta}}_{ELM} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$ unless $\tilde{N} \neq N$ and $\mathbf{H}^T\mathbf{H}$ is singular. Otherwise, Moore-Penrose inverse should be used

to get the optimal $\widehat{\boldsymbol{\beta}}$ vector which has the smallest norm least-squares solution. By considering Moore-Penrose inverse, the solution of Eq. (2.3) is defined as

$$\widehat{\boldsymbol{\beta}}_{ELM} = \mathbf{H}^{+}\mathbf{T} \tag{2.5}$$

where $\mathbf{H}^{+}$ is the Moore–Penrose inverse of matrix $\mathbf{H}$.

Several different methods are such as the orthogonalization method, iterative method and singular value decomposition exist to calculate the Moore-Penrose inverse of a matrix (Graybill, 2001; Rao and Mitra, 1971). According to the orthogonal projection method (Schott, 2005), the Moore-Penrose inverse of the $\mathbf{H}$ matrix is found as

$$\mathbf{H}^{+} = \begin{cases} \left(\mathbf{H}^{T}\mathbf{H}\right)^{-1}\mathbf{H}^{T}, & rank(\mathbf{H}) = \tilde{N} \\ \mathbf{H}^{T}\left(\mathbf{H}\mathbf{H}^{T}\right)^{-1}, & rank(\mathbf{H}) = N \end{cases} \tag{2.6}$$

By considering Moore-Penrose inverse in the estimation process of ELM, it has the following remarks (Huang et al., 2004, 2006):

(i) Minimum training error: Moore-Penrose inverse is equal to one of the possible least-squares solutions which provide the smallest training error. It guarantees for avoiding local minimum or infinite training processes.

(ii) Smallest norm of weights: Moore-Penrose inverse presents the solution which has the smallest norm among all possible least squares solutions of $\mathbf{H}\boldsymbol{\beta} = \mathbf{T}$.

(iii) Unique solution: The obtained solution via Moore-Penrose inverse is unique.

The steps of ELM algorithm proposed by Huang et al. (2006) are summarised in Algorithm 2.1.

Algorithm 2.1. *ELM*

For an arbitrary distinct training samples $\omega = \{(x_j,t_j)\,|x_j \in R^n,\ t_j \in R^m,\ j = 1,2,...,N\}$, *activation function* $g(x)$ *and the number of hidden layer nodes* $\tilde{N}$,

1 Randomly assign input weights (w_i) and bias (b_i) parameters, $i = 1,2,...,\tilde{N}$.

2 Calculate the hidden layer output matrix $\mathbf{H}$ by Eq. (2.6)

3 Calculate the output weights using Eq. (2.5).

2.3 Theoretical Properties of ELM

In theory, ELM has considerable properties from the points of some perspectives including interpolation, universal approximation capability, and generalizability (Huang et al., 2015).

2.3.1 Interpolation Theory

The following theorem proves that ELM can converge to any training set if the number of hidden neuron is large enough. It is clear that the theorem is valid even the number of hidden neuron $\left(\tilde{N}\right)$ is equal to size of training set (N).

Theorem 2.1. *(Huang et al., 2006) Given any small positive value* $\varepsilon > 0$ *and activation function* $g : R \to R$ *which is infinitely differentiable in any interval, there exists* $\tilde{N} \leq N$ *such that for N arbitrary distinct samples* (x_j,t_j), *where* $x_j \in R^n$ *and* $t_j \in R^m$, *for any* w_i *and* b_i, $i = 1,2,...,\tilde{N}$, *randomly chosen from any intervals of* R^n *and R, respectively, according to any continuous probability distribution, then with probability one,* $\left\|\mathbf{H}_{N\times\tilde{N}}\hat{\boldsymbol{\beta}}_{\tilde{N}\times m} - \mathbf{T}_{N\times m}\right\| < \varepsilon$.

2.3.2 Universal Approximation Capability

The activation function is an important component of neural networks, particularly SLFNs and effective on the capability of universal approximation. Generally, the main property of any universal learner (i.e. neural networks) is to have an activation function that is continuous and differentiable. This strict assumption is more flexible in ELM learning. As well as the classical and well-known activation functions which are given in Table 2.1 and suitable to conditions like continuity and differentiability, some alternatives including threshold function can be used in ELM. These interpretations are also supported by the following theorem which makes ELM a universal learner.

Theorem 2.2. *(Huang et al.,2006; Huang and Chen, 2007, 2008) Given any non-constant piecewise continuous function* $G : \mathbf{R}^d \to R$*, if span* $\{G(\mathbf{a},b,\mathbf{x}) : (a,b) \in \mathbf{R}^d \times R\}$ *is dense in* L^2*, for any continuous target function* f *and any function sequence* $\{G(\mathbf{a}_i,b_i,\mathbf{x})\}_{i=1}^{\tilde{N}}$ *randomly generated according to any continuous sampling distribution,* $lim_{L\to\infty}\|f - f_{\tilde{N}}\| = 0$ *holds with probability one if the output weights* $\boldsymbol{\beta}_i$ *are determined by ordinary least square to minimize* $\left\| f(\mathbf{x}) - \sum_{i=1}^{\tilde{N}} \boldsymbol{\beta}_i G(\mathbf{a}_i,b_i,\mathbf{x}) \right\|$.

The following theorem confirms that ELM can be used for not only regressional tasks but also classification tasks which have any complex decision hyperplane.

Theorem 2.3. *(Huang et al., 2012) Given any feature mapping* $\mathbf{h}(\mathbf{x})$*, if* $\mathbf{h}(\mathbf{x})$ *is dense in* $C(\mathbf{R}^d)$ *or in* $C(\mathbf{D})$*, which is a compact set of* $\mathbf{R}^d$*, then a generalized SLFN with such a random hidden layer mapping* $\mathbf{h}(\mathbf{x})$ *can separate arbitrary disjoint regions of any shapes in* $\mathbf{R}^d$ *or* $\mathbf{D}$.

2.3.3 Generalizability Capability

In real-world applications, the main goal is generally to obtain generalizable results which can provide more accurate and consistent results on new data sets. Various methods are used to examine the capability of the generalizability of any model. Bayesian framework, statistical learning theory (SLT) and cross-validation are the most well-known and used methods. Vapnik–Chervonenkis (VC) dimension theory which is defined in SLT is an effective method to determine the generalization bound of ELM for classification tasks (Huang et al., 2015). It is shown that the VC dimension of ELM is equal to the number of hidden layer neurons $\left(\tilde{N}\right)$ with probability one (Liu et al., 2012).

Theorem 2.4. *(Liu et al., 2012) The VC dimension of ELM with $\tilde{N}$ hidden nodes which are infinitely differentiable in any interval is equal to $\tilde{N}$ with probability one.*

As well as ELM has some superior properties from the perspective both theoretical and practical, Huang et. al. (2015) and Huang (2014, 2020) claimed that ELM provides more similar learning to biological learning than conventional learning like backpropagation, etc. Table 2.2 gives this comparison between conventional learning and ELM learning (Huang, 2020):

Table 2.2. Comparison of conventional learning and ELMs

Situation	Conventional learning	ELMs
Dependency to network size	Instable	Stable
Work load for parallel implementation	High	Very low
Work load for hardware implementation	High	Very low
Sensitivity to parameter selection	Very sensitive	Very low
Dependency on the type of application	High	None
Processing time	Low	Very fast
Work load for online sequantial learning	High	Very low
Possibility for achieving global optimum	Low	High

3. ESTIMATION METHODS IN LINEAR REGRESSION

Regression analysis is one of the major area in machine learning and has been widely used for different learning tasks in various disciplines due to the some superios properties like simplicity, interpretability and easy integrability. A classical linear regression model can be expressed in a matrix notation as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},\ i = 1,2,...,n \tag{3.1}$$

where $\mathbf{y}$ is an $n \times 1$ vector of the response variable, $\mathbf{X}$ is an $n \times p$ matrix of explanatory variables, $\boldsymbol{\beta}$ is a $p \times 1$ vector of coefficients to be estimated and $\boldsymbol{\varepsilon}$ is an $n \times 1$ vector of random errors. It is assumed that the errors are normally and identically distributed with zero mean and σ^2 constant variance and uncorrelated (i.e. independent). From the experimental view, explanatory variables are assumed to be fixed and measured without error (Montgomery et al., 2012).

There have been various methods to estimate the coefficients in a linear regression model. The most well-known method is the ordinary least squares (OLS) approach. In brief, OLS is based on minimization of the residuals (errors) sum of squares which is given as

$$\boldsymbol{C}(\boldsymbol{\beta}) = \sum_{i=1}^{n} \boldsymbol{\varepsilon}_i^2 = \boldsymbol{\varepsilon}^T \boldsymbol{\varepsilon} = (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^T (\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) \tag{3.2}$$

The $\boldsymbol{\beta}$ via OLS estimator can be estimated by using simple algebra as

$$\hat{\boldsymbol{\beta}}_{OLS} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{3.3}$$

Although the OLS estimator is commonly used, it has some drawbacks which are listed as

(i) The OLS is not suitable in terms of the capability of dealing with multicollinearity.

(ii) The OLS does not provide sparse solutions which means that the estimated coefficients are different from zero. In other words, it does not have a variable selection property.

(iii) The OLS solution is not unique in the presence of high dimensional data. It is possible to obtain many solutions which make the cost function zero. This result shows that there is high risk of over-fitting in OLS estimations (Friedman et al., 2001; James et al., 2013; Hastie et al., 2015).

The multicollinearity problem mentioned in (a) is defined as the presence of highly correlated explanatory variables unlike the independency assumption in theory. The main reason underlying the multicollinearity problem can be expressed as follows (Montgomery et al., 2012):

- The method choice for data collecting can adversely affect if the sample is mainly selected from a region of highly correlated features space.
- The dependency of model or population conditions like natural relationships that often occur in chemical or production processes.
- Determination of inappropriate model like adding some polynomial terms of explanatory variables may increase the level of multicollinearity.
- The usage of the over-defined model which is based on much more explanatory variables compared to the number of sample units. The general suggestions for avoiding over-defined model are the remove some variables, to

consider a subset of them, to determine the important variables from the preliminary studies or to use some alternative methods like principal components regression.

Multicollinearity can cause some adverse effects on the performance of linear regression models (Judge et al., 1988; Montgomery et al., 2012). These effects are (i) larger values of OLS coefficients, (ii) larger variance and covariances of OLS, (iii) model with high variance and (iv) misinterpreted model results and coefficients. If the multicollinearity exists between several variables, the separate effects of each variable can not be discriminated properly. Therefore, significant variables are labeled as insignificant. Estimation results may be too sensitive to the deletion of some variables (Judge et al., 1988).

In order to deal with the aforementioned drawbacks of OLS and to obtain more accurate and predictive models, there have been alternative estimators to OLS in the literature. In this section, the theoretical and conceptional properties of these estimators are given.

3.1 Ridge Estimator

The ridge estimator, which is the most well-known and used in machine learning and data-oriented studies as an alternative to the OLS estimator, is proposed by Hoerl and Kennard (1970) is defined as

$$\hat{\boldsymbol{\beta}}_R = \left(\mathbf{X}^T\mathbf{X} + k\mathbf{I}_p\right)^{-1}\mathbf{X}^T\mathbf{y},\ k \geq 0 \tag{3.4}$$

where k is called as ridge tuning parameter and $\mathbf{I}_p$ shows the identity matrix of dimension p. Ridge estimator deals with the multicollinearity problem by adding a small positive term (k) to the diagonal elements of $\mathbf{X}^T\mathbf{X}$ matrix. For a positive

optimal k value, the ridge estimator may provide better results than ordinary least squares. In short, ridge estimator improves OLS in terms of prediction accuracy and stability of coefficients for a certain amount of increasing on the bias.

3.1.1 The Properties of Ridge Estimator

- The ridge estimator can be written as a linear form of the ordinary least-squares as

$$\begin{aligned} \hat{\boldsymbol{\beta}}_R &= \left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-1}\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}} \\ &= \boldsymbol{\omega}_k\hat{\boldsymbol{\beta}}_{OLS} \end{aligned} \tag{3.5}$$

  where $\hat{\boldsymbol{\beta}}_{OLS}$ is the OLS solution given in Eq. (3.3). It is clear that $\hat{\boldsymbol{\beta}}_R = \hat{\boldsymbol{\beta}}_{OLS}$ for $k = 0$. By using the OLS solution, the following expectation can be defined (Judge et al., 1988)

$$E\left(\hat{\boldsymbol{\beta}}_{OLS}^T\hat{\boldsymbol{\beta}}_{OLS}\right) = \boldsymbol{\beta}^T\boldsymbol{\beta} + \sigma^2 tr\left(\mathbf{X}^T\mathbf{X}\right)^{-1} > \boldsymbol{\beta}^T\boldsymbol{\beta} + \frac{\sigma^2}{\lambda_k}$$

  where $\boldsymbol{\beta}$ is the true parameters, $tr(.)$ refers to the trace of the matrix and λ_k is the kth eigenvalue of $\mathbf{X}^T\mathbf{X}$ matrix. If the data matrix is ill-conditioned, the eigenvalues will be smaller with the proportionality of the level of ill-conditioning. Smaller eigenvalue increases the right part of the inequality (). Therefore, the difference between the estimated coefficients vector and the true parameter vector will be larger (Judge et al., 1988).

- The covariance matrix of $\hat{\boldsymbol{\beta}}_R$ is obtained as

$$Var\left(\hat{\boldsymbol{\beta}}_R\right) = \sigma^2\left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-1}\mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-1}$$

- The bias vector of $\hat{\boldsymbol{\beta}}_R$ is defined as

$$Bias\left(\hat{\boldsymbol{\beta}}_R\right) = -k\left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-1}\boldsymbol{\beta}$$

- The mean square error of ridge estimator (Judge et al., 1988) is equal to

$$\begin{aligned} MSE\left(\hat{\boldsymbol{\beta}}_R\right) &= Var\left(\hat{\boldsymbol{\beta}}_R\right) + \left[Bias\left(\hat{\boldsymbol{\beta}}_R\right)\right]\left[Bias\left(\hat{\boldsymbol{\beta}}_R\right)\right]^T \\ &= \sigma^2 tr\left[\left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-1}\mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-1}\right] \\ &\quad +k^2\boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-2}\boldsymbol{\beta} \\ &= \sigma^2\sum_{j=1}^{p}\frac{\lambda_j}{(\lambda_j+k)^2} + k^2\boldsymbol{\beta}^T\left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-2}\boldsymbol{\beta} \end{aligned}$$

It is clearly seen that as the k value increases, $Var\left(\hat{\boldsymbol{\beta}}_R\right)$ decreases, despite the increase in $Bias\left(\hat{\boldsymbol{\beta}}_R\right)$ term. From this viewpoint, an optimal k value which provides the reduction in the variance term is greater than the increase in the squared bias should be selected. This is the best possible way to get a smaller mean square error via ridge estimator than OLS.

- The residual sum of squares for ridge estimator is defined as

$$\begin{aligned} SS_{Res} &= \left(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}_R\right)^T\left(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}_R\right) \\ &= \left(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}_{OLS}\right)^T\left(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}_{OLS}\right) \\ &\quad +\left(\hat{\boldsymbol{\beta}}_R-\hat{\boldsymbol{\beta}}_{OLS}\right)^T\mathbf{X}^T\mathbf{X}\left(\hat{\boldsymbol{\beta}}_R-\hat{\boldsymbol{\beta}}_{OLS}\right) \end{aligned} \tag{3.6}$$

  where the first part of SS_{Res} corresponds to the residual sum of squares for the OLS estimates. It can be said that as k increases, the ridge residual sum of squares increases.

- As $k\to\infty$, $\hat{\boldsymbol{\beta}}_R^T\hat{\boldsymbol{\beta}}_R < \hat{\boldsymbol{\beta}}_{OLS}^T\hat{\boldsymbol{\beta}}_{OLS}$ and $\hat{\boldsymbol{\beta}}_R^T\hat{\boldsymbol{\beta}}_R \to 0$.

3.1.2 The Selection Methods for Ridge Tuning Parameter

The selection of ridge tuning parameters dramatically affects the performance of the estimator. However, there is no concensus about the best method related with choosing this parameter. Different researchers have been proposed different methods. Although the majority of these suggestions are based on minimizing the MSE criterion, there are some popular and alternative ways to choose the proper ridge tuning parameter are to use the Akaike information criterion (Akaike, 1973, 1998), Bayesian information criterion (Schwarz, 1978) and Cross-validation error (Hastie et al. 2001). In this section, some of suggestions for the ridge tuning parameters are taken into consideration.

- The first suggestion by Hoerl and Kennard (1970) is to use the ridge trace which is a plot of the estimated coefficients versus the ridge tuning parameter. The range of ridge tuning parameter (k) is generally taken as $[0,1]$. According to the level of multicollinearity, some coefficients significantly change as k increases. For some a certain range of k, the coefficient will be more stable as visually. The k points providing stable coefficients are seen as reasonable for obtaining smaller MSE than OLS.

- The second suggestion of Hoerl and Kennard (1970) is given as

$$\hat{k}_1 = \frac{\hat{\sigma}^2}{\hat{\alpha}^2_{\max}}. \tag{3.7}$$

  where $\hat{\sigma}^2$ is the OLS estimate of σ^2 and $\hat{\alpha}^2_{\max}$ is the maximum element of $\hat{\boldsymbol{\alpha}} = \mathbf{S}^T \hat{\boldsymbol{\beta}}_{OLS}$. Here $\mathbf{S}$ is an orthogonal matrix such that $\mathbf{S}^T\mathbf{X}^T\mathbf{X}\mathbf{S} = \boldsymbol{\Lambda}$ and $\boldsymbol{\Lambda} = diag\,(\lambda_1, \lambda_2, ..., \lambda_p)$ is a diagonal matrix whose diagonal elements are the

eigenvalues of $\mathbf{X}^T\mathbf{X}$. $\hat{\sigma}^2$ is computed as

$$\hat{\sigma}^2 = \frac{\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{OLS}\right)^T \left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}_{OLS}\right)}{n-p}. \tag{3.8}$$

- The third suggestion proposed by Hoerl et al. (1975) is defined as

$$\hat{k}_2 = \frac{p\hat{\sigma}^2}{\sum_{i=1}^{p}\hat{\alpha}_i^2} = \frac{p\hat{\sigma}^2}{\hat{\boldsymbol{\beta}}_{OLS}^T\hat{\boldsymbol{\beta}}_{OLS}}. \tag{3.9}$$

- Lawless and Wang (1976) suggested another estimator which is given as

$$\hat{k}_3 = \frac{p\hat{\sigma}^2}{\sum_{i=1}^{p}\lambda_i\hat{\alpha}_i^2} = \frac{p\hat{\sigma}^2}{\hat{\boldsymbol{\beta}}_{OLS}\mathbf{X}^T\mathbf{X}\hat{\boldsymbol{\beta}}_{OLS}}. \tag{3.10}$$

In addition to the analytical methods, the alternatives based on information measures such as AIC and BIC should be carefully considered. These criteria are basically based on searching for a balance between model performance and parsimony. Basically, these measures are calculated for each of a range of k values under consideration. Later on, the k value providing minimum AIC, BIC and CV error is selected and used for training the whole data set. AIC and BIC criteria are defined as

$$AIC = n.log(RSS) + 2.df \tag{3.11}$$

$$BIC = n.log(RSS) + df.log(n) \tag{3.12}$$

where n is the number of observations, df shows the degrees of freedom and RSS is the residual sum of squares. The degree of freedom is equal to the trace of quasi-projection matrices of the corresponding to the estimators used (Martínez et al., 2011; Wieringen, 2015; Breheny, 2020;).

In cross-validation process, the data are split into S folds, $S-1$ of the folds is used to fit the data and the rest is used to calculate the error. For S steps, this process is repeated. The overall test error is equal to the mean of these error values. While AIC and BIC are indirect estimates of test error, CV is calculated directly from the data set.

Due to the importance of parameter selection on the performance of ridge type estimators, there are many suggestions in the literature. Only some of them are given here. The reader is referred to Hocking et al. (1976) and Kibria (2003) to provide more information.

3.2 Generalized Ridge Estimator

In the ridge estimator, all coefficients are shrunk with the same ridge tuning parameter and the weight for each coefficient is equal to each other. In order to assign unequal weights (i.e. different parameters) to coefficients, the generalized ridge regression estimator is defined by Hoerl and Kennard (1970) as

$$\hat{\boldsymbol{\beta}}_{GR} = \left(\mathbf{X}^T\mathbf{X}+\mathbf{K}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{K} = diag\left(k_1, k_2, ..., k_p\right)$, is a diagonal matrix whose elements are the ridge tuning parameters, $k_i \geq 0,\ i = 1, ..., p$.

The MSE of generalized ridge estimator can be given as

$$\begin{aligned} MSE\left(\hat{\boldsymbol{\beta}}_{GR}\right) &= \sigma^2\left(\mathbf{X}^T\mathbf{X}+\mathbf{K}\right)^{-1}\mathbf{X}^T\mathbf{X}\left(\mathbf{X}^T\mathbf{X}+\mathbf{K}\right)^{-1} \\ &\quad + \left(\mathbf{X}^T\mathbf{X}+\mathbf{K}\right)^{-1}\mathbf{K}\boldsymbol{\beta}\boldsymbol{\beta}^T\mathbf{K}\left(\mathbf{X}^T\mathbf{X}+\mathbf{K}\right)^{-1}. \end{aligned}$$

3.3 Almost Unbiased Ridge Estimator

Singh et al. (1986) proposed the almost unbiased ridge estimator by jackknifing the ordinary ridge estimator and claimed that the proposed algorithms guar-

antee the reduction on the sum of square bias. The almost unbiased ridge estimator is defined as

$$\hat{\boldsymbol{\beta}}_{AUR} = \left[\boldsymbol{I} - k^2 \left(\mathbf{X}^T\mathbf{X} + k\mathbf{I}\right)^{-2}\right] \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{y},\ k \geq 0 \tag{3.13}$$

where k is the ridge tuning parameter.

3.4 Generalized Almost Unbiased Ridge Estimator

The generalized form of the almost unbiased ridge estimator is also proposed by Singh et al. (1986) as

$$\hat{\boldsymbol{\beta}}_{GAUR} = \left[\boldsymbol{I} - \left(\mathbf{X}^T\mathbf{X} + \boldsymbol{K}\right)^{-2} \boldsymbol{K}^2\right] \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{y}$$

where $\mathbf{K} = diag\left(k_1, k_2, ..., k_p\right),\ k_i \geq 0,\ i = 1, ..., p.$

3.5 Principal Components Regression Estimator

Principal components regression estimator (PCR) is another biased estimator that is based on the reduction of the number of variables by using newly created regressors called PCs. The advantage of this estimator to provide more stable and faster results than OLS and ridge type estimators. The main goal is to obtain fewer components explaining the variance as much as possible. At the end of the process, the PCs are orthogonal. Therefore, the PCR is considered as powerful to deal with multicollinearity. The estimator was first investigated by Hotelling (1933). The model (3.1) can be written in canonical form as

$$\mathbf{y} = \mathbf{C}\boldsymbol{\alpha} + \boldsymbol{\varepsilon} \tag{3.14}$$

where $\mathbf{C} = \mathbf{XS}$, $\boldsymbol{\alpha} = \mathbf{S}^T\boldsymbol{\beta}$ and $\mathbf{S}^T\mathbf{X}^T\mathbf{XS} = \mathbf{C}^T\mathbf{C} = \boldsymbol{\Lambda} = diag\left(\lambda_1, \lambda_2, ..., \lambda_p\right)$, and λ_i is the ith eigenvalue of $\mathbf{X}^T\mathbf{X}$ matrix and $\mathbf{S}$ is a $(p \times p)$ matrix whose columns are

the eigenvectors corresponding to each eigenvalue. After creating the PCs, OLS is applied to these PCs which are also the columns of the **C** matrix. In order to obtain the OLS solution on PCs, the elements in Eq. (3.14) should be partitioned as

$$\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_r & \mathbf{0} \\ \mathbf{0}^T & \boldsymbol{\Lambda}_{p-r} \end{pmatrix}, \ \boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\alpha}_r \\ \boldsymbol{\alpha}_{p-r} \end{pmatrix}, \ \mathbf{S} = \begin{pmatrix} \mathbf{S}_r & \mathbf{S}_{p-r} \end{pmatrix}$$

By using these partitions, Eq. (3.14) can be written as

$$\mathbf{y} = \mathbf{C}_r \boldsymbol{\alpha}_r + \mathbf{C}_{p-r} \boldsymbol{\alpha}_{p-r} + \boldsymbol{\varepsilon}$$

where $\mathbf{C}_r = \mathbf{X}\mathbf{S}_r$, $\boldsymbol{\alpha}_r = \mathbf{S}_r^T \boldsymbol{\alpha}$, $\mathbf{C}_{p-r} = \mathbf{X}\mathbf{S}_{p-r}$ and $\boldsymbol{\alpha}_{p-r} = \mathbf{S}_{p-r}^T \boldsymbol{\alpha}$.

The first r components $(\boldsymbol{\alpha}_r)$ should be used in the presence of multicollinearity. That's why, the first r components is estimated via OLS as

$$\hat{\boldsymbol{\alpha}} = \boldsymbol{\Lambda}_r^{-1} \mathbf{C}_r^T \mathbf{y} \tag{3.15}$$

By considering $\boldsymbol{\beta} = \boldsymbol{S}\boldsymbol{\alpha}$ in Eq. (3.15), the estimation of $\boldsymbol{\beta}$ coefficients by PCR estimator is defined as

$$\hat{\boldsymbol{\beta}}_{PCR} = \mathbf{S}_r \hat{\boldsymbol{\alpha}}_r = \mathbf{S}_r \boldsymbol{\Lambda}_r^{-1} \mathbf{C}_r^T \mathbf{y} = \mathbf{S}_r \mathbf{S}_r^T \hat{\boldsymbol{\beta}}_{OLS} \tag{3.16}$$

3.6 r-k Class Estimator

The r-k class estimator was proposed by Baye and Parker (1984) to eliminate the performance loss in the presence of multicollinearity in the data set. This estimator is a general form of the most well-known three estimators in linear regression which are OLS, ridge and PCR estimators. The r-k class estimator is defined as

$$\hat{\boldsymbol{\beta}}_r(k) = \mathbf{S}_r \left(\mathbf{S}_r^T \mathbf{X}^T \mathbf{X} \mathbf{S}_r + k \mathbf{I}_r \right)^{-1} \mathbf{S}_r^T \mathbf{X}^T \mathbf{y} \tag{3.17}$$

where r is the first r selected PCs and k is the ridge tuning parameter. This estimator is reduced to OLS, ridge or PCR estimators under some particular conditions:

- $\hat{\boldsymbol{\beta}}_p(0) = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$ corresponds the OLS estimator given by Eq. (3.3),
- $\hat{\boldsymbol{\beta}}_r(0) = \mathbf{S}_r\left(\mathbf{S}_r^T\mathbf{X}^T\mathbf{X}\mathbf{S}_r\right)^{-1}\mathbf{S}_r^T\mathbf{X}^T\mathbf{y}$ corresponds the PCR estimator given by Eq. (3.16),
- $\hat{\boldsymbol{\beta}}_p(k) = \left(\mathbf{X}^T\mathbf{X}+k\mathbf{I}_p\right)^{-1}\mathbf{X}^T\mathbf{y}$ corresponds the ridge estimator given by Eq. (3.4).

3.7 Liu Estimator

Liu estimator was proposed by Liu (1993) as an alternative to the ridge-type estimator to deal with multicollinearity. Although the idea behind the Liu estimator is similar in terms of shrinking the estimated with a small constant (i.e. Liu tuning parameter), the form of Liu tuning parameter in Liu estimator has a linear form, unlike the non-linear form in ridge estimator. The result of this situation is to be able to calculate easier and faster the Liu tuning parameter than the ridge tuning parameter. Another advantage of the Liu estimator over the ridge estimator is to be able to select the appropriate tuning parameter. The general form of the Liu estimator is given as

$$\hat{\boldsymbol{\beta}}_{Liu}^{(d)} = \left(\mathbf{X}^T\mathbf{X}+\mathbf{I}_p\right)^{-1}\left(\mathbf{X}^T\mathbf{y}+d\hat{\boldsymbol{\beta}}_{OLS}\right),\ 0<d<1 \tag{3.18}$$

where d refers to the Liu tuning parameter and $\hat{\boldsymbol{\beta}}_{OLS}$ is the OLS estimator. Actually, Liu estimator is a combination of the ridge estimator for $k=1$ and the OLS estimator which is defined as

$$\hat{\boldsymbol{\beta}}_{Liu}^{(d)} = (1-d)\,\hat{\boldsymbol{\beta}}_R(k=1) + d\hat{\boldsymbol{\beta}}_{OLS}$$

The general properties of the Liu estimator can be listed as (Liu, 1993):

- Liu estimator is reduced to the OLS estimator under a particular situation of $d = 1$ and can be written as a linear transform of OLS solution as

$$\hat{\boldsymbol{\beta}}_{Liu}^{(d)} = \left(\mathbf{X}^T\mathbf{X}+\mathbf{I}\right)^{-1}\left(\mathbf{X}^T\mathbf{y}+d\mathbf{I}\right)\hat{\boldsymbol{\beta}}_{OLS}$$

- The norm of Liu estimator is smaller than OLS: $\left\|\hat{\boldsymbol{\beta}}_{Liu}^{(d)}\right\| < \left\|\hat{\boldsymbol{\beta}}_{OLS}\right\|$

- $E\left(\hat{\boldsymbol{\beta}}_{Liu}^{(d)}\right) = \left(\mathbf{X}^T\mathbf{X}+\mathbf{I}\right)^{-1}\left(\mathbf{X}^T\mathbf{y}+d\mathbf{I}\right)\boldsymbol{\beta}$ and the Liu estimator is a biased estimator.

- There is at least one Liu tuning parameter providing a smaller MSE than that of the OLS estimator.

- The scalar mean squared error for the Liu estimator is given as

$$SMSE\left(\hat{\boldsymbol{\beta}}_{Liu}^{(d)}\right) = \sigma^2\sum_{i=1}^{p}\frac{\left(\lambda_i+d\right)^2}{\lambda_i\left(\lambda_i+1\right)} + \left(d-1\right)^2\sum_{i=1}^{p}\frac{\alpha_i^2}{\left(\lambda_i+1\right)}$$

- The optimal d parameter providing the smallest $SMSE\left(\hat{\boldsymbol{\beta}}_{Liu}^{(d)}\right)$ can be obtained as

$$d = 1-\sigma^2\left(\sum_{i=1}^{p}\frac{1}{\lambda_i\left(\lambda_i+1\right)}\Big/\sum_{i=1}^{p}\frac{\alpha_i^2}{\left(\lambda_i+1\right)}\right) \tag{3.19}$$

Eq. (3.19) can be written in a generalized form which gives the minimum SMSE value as

$$d_h = 1-h\sigma^2\left(\sum_{i=1}^{p}\frac{1}{\lambda_i\left(\lambda_i+1\right)}\Big/\sum_{i=1}^{p}\frac{\alpha_i^2}{\left(\lambda_i+1\right)}\right),\ h>0 \tag{3.20}$$

where h keeps the range of d parameters as $[0,1]$. Liu (1993) claimed the SMSE of the Liu estimator is smaller than that of the OLS estimator for all h values selected

from the range

$$0 < h < \frac{2(n-p)}{n-p+2}\left[1 - \frac{2}{\sum_{i=1}^{p} \frac{\lambda_p(\lambda_p+1)}{\lambda_i(\lambda_i+1)}}\right] \tag{3.21}$$

where λ_p is the largest eigenvalue of $\mathbf{X}^T\mathbf{X}$.

Some alternative suggestions on determining the appropriate Liu tuning parameter are proposed by Liu (1993) as

$$d_{CL} = 1 - \sigma^2\left(\sum_{i=1}^{p}\frac{1}{(\lambda_i+1)} \Big/ \sum_{i=1}^{p}\frac{\lambda_i\alpha_i^2}{(\lambda_i+1)}\right) \tag{3.22}$$

where d_{C_L} is the minimizer of C_L criterion defined as $C_L = SS_{res}(d)/\hat{\sigma}^2 + 2tr(\mathbf{H}_d) - (n-2)$ and $\mathbf{H}_d = \tilde{\mathbf{X}}^T\left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}+\mathbf{I}\right)^{-1}\left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}+d\mathbf{I}\right)\tilde{\mathbf{X}}^T$. Here $\tilde{\mathbf{X}}$ is the centered form of $\mathbf{X}$ matrix and $SS_{res}(d)$ is the residual sum of squares using the Liu solution of centered data. Liu (1993) showed that d_{C_L} also minimizes the generalized cross validation error which is defined as $GCV = SS_{res}(d)/(n-1-tr(\mathbf{H}_d))$.

3.8 Lasso Estimator

OLS estimator has some drawbacks because of producing less interpretable models with high variance in high dimensional and correlated data. Ridge estimator was seen as a remedy to high variance problems by dealing with multicollinearity. Ridge estimator has been widely used in different fields due to its superiorities like stability, predictive performance, and functionality on high dimensionality settings over OLS. However, ridge estimator does not present a sparse (i.e. compact) model which is quite important to deal with large data sets including irrelevant or noisy features. In other words, ridge estimator does not carry out a variable selection process and effect on the compactness of the model. This yields a less interpretable model compared to its competitors. In other words, the ridge estimator doesn't make any

contribution to the interpretable property of any model because it doesn't carry out a variable selection process. Therefore, the least absolute shrinkage and selection operator (Lasso) is proposed by Tibshirani (1996) in order to obtain a more predictive and sparse solution than OLS and ridge by carrying out variable selection. Lasso estimator is defined as

$$\hat{\boldsymbol{\beta}}_{Lasso} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2, \; subject\ to \; \sum_{j=1}^{p} \left|\beta_j\right| \leq t \tag{3.23}$$

where t corresponds to the bound on the sum of the absolute values of coefficients and corresponding the upper limit of maximum size for expanding.

$\hat{\boldsymbol{\beta}}_{Lasso}$ can be also written in Lagrangian form as

$$\hat{\boldsymbol{\beta}}_{Lasso} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \left|\beta_j\right| \tag{3.24}$$

where λ is the Lagrangian multiplier.

The first representation of Eq. (3.23) is known as a constrained optimization problem whereas the second one (Eq. (3.24)) is called a Lagrangian form. There is a λ value corresponding to each t value and the solutions obtained via any form of this optimization problem are exactly the same each other. In other words, the $\hat{\boldsymbol{\beta}}_{Lasso}$ solution corresponding any λ value in Lagrangian form solves the problem which has the bound of $t = \left\| \hat{\boldsymbol{\beta}}_{\lambda} \right\|$.

The advantage of the Lasso method is that some of the coefficients are shrunk to exactly zero. Thus, a much more sparse model is obtained. Both the ridge and Lasso regression model shrink the coefficients with the proportional tuning parameters but Lasso carries out the variable selection by setting some coefficients to zero. Due to these properties, Lasso and its variants have been attracted to many disciplines.

3.8.1 The Solution of Lasso Estimator

Because of the impossibility of closed-form solutions, several algorithms have been proposed to obtain solution in Lasso-type problems. Fu (1998) proposed an algorithm called as coordinate descent algorithm (i.e. shooting algorithm) by modifying with the structure of the bridge estimator which a kind of framework of estimators including OLS, ridge, Lasso, etc. Osborne et al. (2000) developed some new approaches to deal with the difficulties on implementing the Lasso. They presented a compact descent algorithm and suggested to use the modified Gram-Schmidt orthogonalization to improve the effectiveness. Rosset and Zhu (2007) developed an efficent and robust lasso algorithm providing the piecewise linear coefficient paths for all kind of the loss functions which are the members of "almost quadratic" family including L_1 norm. The almost quadratic term means that the part of cost functions except the L_1 norm is differentiable at almost all points in the range of corresponding variable. That is, the derivatives of the corresponding part of the cost function cannot be obtained for a certain value or range of variable. Efron et al. (2004) proposed least angle regression (LARS) which is one of the most well-known and used algorithm to obtain Lasso solutions. LARS is actually an individual algorithm proposed by model selection as an alternative to traditional forward selection methods and has some superior properties like computationally efficient and less greedy with compared to its competitors. The simple and efficient idea underlying the LARS algorithm is to project the residuals onto the active (selected at in iteration) variables and to update the residual direction until a new variable is seen as reasonable (i.e. reasonably correlated with residuals). By doing a simple modification on LARS, it can be applicable to Lasso-type problems. When LARS is

compared with the coordinate descent algorithm, it is more complex and less flexible (Breheny, 2020). That's why, the coordinate descent algorithm has been more widely used than LARS. It should be noted that the computational cost of coordinate descent algorithm could be higher than LARS because of the number of iteration for terminating the training process is not determined (Murphy, 2012). Friedman et al. (2010) proposed a group of algorithms based on cyclical coordinate descent for generalized linear models with convex penalties including OLS, ridge, Lasso and the elastic net.

In this study, we adopted the algorithm of Sjöstrand et al. (2018) which is a special form of piecewise linear coefficient paths proposed by Rosset and Zhu (2007). This algorithm is functional due to the property of applicability for both the Lasso and the elastic net models. The main difference between the Lasso and the elastic net is the number of tuning parameter. If the Lasso parameter is fixed to zero, the Lasso algorithm will provide the elastic net solutions. The details of coordinate descent, LARS and piecewise linear regularized solution paths algorithms are given in Algorithms 3.1, 3.2 and 3.3, respectively.

Algorithm 3.1. ***Coordinate descent algorithm (Fu, 1998; Murphy, 2012)***

set $\mathbf{w} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$

repeat

for $j = 1, ..., p$

$\alpha_j = 2\sum_{i=1}^{n} x_{ij}^2$

$c_j = 2\sum_{i=1}^{n} x_{ij}\left(y_i - \mathbf{w}^T\mathbf{x}_i + w_j x_{ij}\right)$

$w_j = soft\left(\frac{c_j}{\alpha_j}, \frac{\lambda}{\alpha_j}\right)$

until convergence is met

Algorithm 3.2. ***LARS algorithm (Efron et al., 2004; Sjöstrand et al., 2018)***.

Set the first coefficient and estimation vector as $\beta^{(0)} = \mathbf{0}$ and $\widehat{\mathbf{y}}^{(0)} = \mathbf{0}$

Set the active and inactive variable sets as $A = \varnothing$ and $I = \{1, ., , , .p\}$

for $k = 0$ to $p - 2$

do

Update the residual $\varepsilon = \mathbf{y} - \widehat{\mathbf{y}}^{(k)}$

Find the maximal correlation $c = \max_{i\varepsilon I}\left|x_i^T\varepsilon\right|$

Move variable corresponding to c from I to A

Calculate the least square solution $\beta_{OLS}^{(k+1)} = \left(X_A^T X_A\right)^{-1} X_A^T y$

Calculate the current direction $d = X_A\beta_{OLS}^{(k+1)} - \widehat{\mathbf{y}}^{(k)}$

Calculate the step length $\gamma = \min_{i\varepsilon I}\left\{\frac{x_i^T\varepsilon - c}{x_i^T d - c}, \frac{x_i^T\varepsilon + c}{x_i^T d + c}\right\}, 0 < \gamma \leq 1$

Update regression coefficients $\beta^{(k+1)} = (1-\gamma)\beta^{(k)} + \gamma\beta_{OLS}^{(k+1)}$

Update the fitted vector $\widehat{\mathbf{y}}^{(k+1)} = \widehat{\mathbf{y}}^{(k)} + \gamma d$

end

Let $\beta^{(p)}$ be the full least squares solution $\beta^{(p)} = \left(X^T X\right)^{-1} X^T y$

Output the series of coefficients $B = \left[\beta^{(0)}, ..., \beta^{(p)}\right]$

Algorithm 3.3. ***Piecewise linear regularized solution paths algorithm (Rosset and Zhu, 2007; Sjöstrand et al., 2018)***.

1 Initialize $\beta^{(0)} = 0$, $A = \arg\max_j \left|x_i^T y\right|$, $\bigtriangledown \widehat{\beta}_A^{(0)} = -sign\left(X_A^T y\right)$, $\bigtriangledown \widehat{\beta}_I^{(0)} = 0$, $k = 0$.

2 while $I \neq \varnothing$

3 do

4 $\gamma_j = \min_{j \varepsilon A} -\frac{\beta_j^{(k)}}{\bigtriangledown \widehat{\beta}_j^{(k)}}$

5 $\gamma_i = \min_{j \varepsilon I} \left\{ \frac{\left(x_i + x_j\right)^T \left(\mathbf{y} - \mathbf{X}\widehat{\beta}^{(k)}\right)}{\left(x_i + x_j\right)^T \left(\mathbf{X} \bigtriangledown \widehat{\beta}_j^{(k)}\right)}, \frac{\left(x_i - x_j\right)^T \left(\mathbf{y} - \mathbf{X}\widehat{\beta}^{(k)}\right)}{\left(x_i - x_j\right)^T \left(\mathbf{X} \bigtriangledown \widehat{\beta}_j^{(k)}\right)} \right\}$,where j is any index in A

6 $\gamma = \min\left\{\gamma_j, \gamma_i\right\}$

7 if $\gamma = \gamma_j$

8 move j from A to I

9 else

10 move i from I to A

11 end if

12 $\widehat{\beta}^{(k+1)} = \widehat{\beta}^{(k)} + \gamma \bigtriangledown \widehat{\beta}^{(k)}$

13 $\bigtriangledown \widehat{\beta}_A^{(k+1)} = -\left(2X_A^T X_A\right)^{-1} .sign\left(\widehat{\beta}_A^{(k+1)}\right)$

14 $k = k + 1$

15 end while

16 Output the series of coefficients $B = \left[\beta^{(0)}, ..., \beta^{(k)}\right]$

In Algorithms 3.1-3.3, A corresponds to the active set of coefficients which shows the selected variables at a certain iteration, I corresponds to the inactive set of coefficients which is updated at each iteration. Besides, $soft$ is the soft thresholding function which is defined as

$$soft\left(a;\lambda\right) \triangleq sign\left(a\right)\left(\left|a\right| - \lambda\right)_+$$

where $x_+ = \max(x, 0)$ is the positive part of x.

The appropriate value of Lasso tuning parameter (t or λ) is dramatically effective on the performance. Similar to the ridge tuning parameter, AIC, BIC and CV criteria can be used to determine the appropriate parameter. By starting a grid of λ values, the values of AIC and BIC criteria and cross-validation errors have been obtained and λ value corresponding to the smallest CV error, AIC or BIC value is selected for the final model. Finally, the entire data set is fitted by using the optimal λ value.

3.9 Elastic Net Estimator

Although Lasso is an effective solution in terms of sparsity performance, it should be noted that ridge estimator may overperform to Lasso in the presence of severe multicollinearity. Besides, there are some drawbacks of Lasso regression which are pointed out by Zou and Hastie (2005) as follows:

- In the presence of high pairwise correlations between variables, it is not a user-dependent process which variable to select. Lasso carries this selection randomly and does not care which one is selected.
- As mentioned before, Lasso is not competitive in comparison with ridge regression for dealing with severe multicollinearity
- In high dimensional setting ($p > N$), the number of selected variables is limited by $min(N; p)$. This problem may not convincing if many more variables are considered as important.

In order to overcome the drawbacks of Lasso, Zou and Hastie (2005) proposed the elastic net as a regularization and variable selection method. In the elastic

net, the superiorities of both ridge and Lasso methods have been used in a unified model. Thus, an effective variable selection process can be carried out by considering the grouping effect (the relationships between variables). The naive elastic net estimator proposed by Zou and Hastie (2005) is defined on a standardized data set as follows:

$$\hat{\boldsymbol{\beta}}_{ENet} = \arg\min_{\mathbf{fi}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \tag{3.25}$$

where λ_1 and λ_2 are the non-negative constants corresponding to the size of the L_1 norm of the coefficients and the size of L_2 norm of the coefficients, respectively. The solution can be written as a constrained form of the optimization problem as

$$\hat{\boldsymbol{\beta}}_{ENet} = \arg\min_{\mathbf{fi}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2, \; subject\ to \left[(1-\alpha) \sum_{j=1}^{p} |\beta_j| + \alpha \sum_{j=1}^{p} \beta_j^2 \right] \leq t$$

where $\alpha \in [0,1]$ and determines the closeness of the solution of the ridge or lasso estimators. For $\alpha = 1$, the solution is equivalent to the ridge solution, and with $\alpha = 0$, the solution is reduced to the lasso solution.

The naive elastic net solution can be obtained by reducing the estimation problem into a lasso-type optimization problem. Consider an augmented data set $(X^*, \mathbf{y}^*)$ created based on the original data set $(\mathbf{X}, \mathbf{y})$ and parameters (λ_1, λ_2) as

$$\mathbf{X}^*_{(n+p)\times p} = \frac{1}{\sqrt{1+\lambda_2}} \begin{pmatrix} \mathbf{X} \\ \lambda_2 \mathbf{I}_p \end{pmatrix} \text{ and } \mathbf{y}^*_{(n+p)} = \begin{pmatrix} \mathbf{y} \\ \mathbf{0}_p \end{pmatrix} \tag{3.26}$$

Assume that $\gamma = \lambda_1/\sqrt{1+\lambda_2}$ and $\boldsymbol{\beta}^* = \sqrt{1+\lambda_2}\boldsymbol{\beta}$, then the naive elastic net criterion can redefined as

$$\hat{\boldsymbol{\beta}}^* = \arg\min_{\boldsymbol{\beta}^*} \left\{ \left\| \mathbf{y}^* - \mathbf{X}^* \boldsymbol{\beta}^* \right\|^2 + \gamma \left| \boldsymbol{\beta}^* \right|_1 \right\} \tag{3.27}$$

The relationship between $\hat{\boldsymbol{\beta}}_{NaiveENet}$ and $\hat{\boldsymbol{\beta}}^{*}$ is

$$\hat{\boldsymbol{\beta}}_{NaiveENet} = \frac{1}{\sqrt{1+\lambda_2}}\hat{\boldsymbol{\beta}}^{*}. \tag{3.28}$$

By using the notation on the augmented data, the elastic net estimates are obtained as

$$\hat{\boldsymbol{\beta}}_{ENet} = \sqrt{1+\lambda_2}\hat{\boldsymbol{\beta}}^{*}. \tag{3.29}$$

In other words, the elastic net estimates can be written as a linear form of the naive elastic net estimates as

$$\hat{\boldsymbol{\beta}}_{ENet} = (1+\lambda_2)\,\hat{\boldsymbol{\beta}}_{NaiveENet}. \tag{3.30}$$

The superiorities of the elastic net over the naive elastic net are given as

- The 'naive' term in the naive elastic net is given because of that it may not overperform ridge or lasso estimators according to applications results except the situation that is the actual solution is close to ridge or lasso solutions. The performance of the naive elastic net is not good enough in terms of dealing with the severe multicollinearity and the variable selection between a group of highly correlated variables.

- Due to the double shrinkage, the naive elastic net causes unwanted bias although it doesn't reduce the variance enough compared to only ridge or lasso. That's why the elastic net is proposed by avoiding double shrinkage.

Zou and Hastie (2005) developed an algorithm called LARS-EN to solve the elastic net optimization problem. LARS-EN is a modification of the algorithm proposed by Efron et al. (2004) to obtain the lasso solution path. On augmented

data, the elastic net problem can be reduced to a lasso-type problem, so LARS-type algorithms can be used to find the entire solution path of the elastic net.

The parameter selection in the elastic net has a critical importance on the performance. The general expression of the elastic net optimization problem includes two parameters: λ_1 and λ_2. However, in practical application, the fraction of the L_1 norm of the coefficients (s) or the L_1 norm of the coefficients (t) is generally used as a tuning parameter. As a result, there is two possible parameter combination to estimate: (s, λ_2) or (t, λ_2). Among these parameters, the range of s is $[0, 1]$. Therefore, it is more suitable for comparative purposes. The general process for determining optimal parameters is to fix the ridge parameter (λ_2) and then search the best parameter(s) by using CV, Akaike information criterion (AIC) or Bayesian information criterion (BIC) with LARS-EN or similar algorithm.

4. THE PROPOSED EXTREME LEARNING MACHINE ALGORITHMS

ELM algorithm performs well on sampling data unless it is measured correctly and it is assumed that there is no multicollinearity or noises in variable values. It is impossible to make an accurate measurement of real-life applications. Furthermore, due to the presence of uncontrollable or non-measurable variables, it is assumed that there are always some noises in measures. In the linear model, these noises are called as errors. By considering this term in the classical ELM model, the model can be defined as

$$\mathbf{H}\boldsymbol{\beta} + \boldsymbol{\varepsilon} = \mathbf{T} \tag{4.1}$$

where $\mathbf{H}_{Nx\tilde{N}}$ is the hidden layer output matrix, $\boldsymbol{\beta}_{\tilde{N}x1}$ is the weight vector between the hidden layer and output nodes, $\mathbf{T}_{Nx1}$ is the observed target values and $\boldsymbol{\varepsilon}_{Nx1}$ is a vector of error terms with mean zero and variance-covariance matrix $\sigma^2\boldsymbol{I}$. Here $\boldsymbol{\varepsilon}$ is assumed to have the distribution of $N\left(0,\sigma^2\boldsymbol{I}\right)$. In the presence of noise terms, the estimator is calculated as follows:

$$\widehat{\boldsymbol{\beta}} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} = \frac{\sum_{l=1}^{m} h_l^T\left(h_l\beta_l + \varepsilon_l\right)}{\sum_{l=1}^{m} h_l^T h_l} = \boldsymbol{\beta} + \frac{\sum_{l=1}^{m} h_l^T\varepsilon_l}{\sum_{l=1}^{m} h_l^T h_l} \tag{4.2}$$

where h_l corresponds to the lth column of $\mathbf{H}$ matrix.

The important statistical properties of estimator in Eq. (4.2) are summarized by Zhang et al. (2016) as:

- $E\left(\widehat{\boldsymbol{\beta}}_{ELM}\right) = \boldsymbol{\beta}$: unbiased estimator
- $V\left(\widehat{\boldsymbol{\beta}}_{ELM}\right) = \sigma^2\sum_{l=1}^{\tilde{N}}\frac{1}{\lambda_l}$: variance

- $SMSE\left(\widehat{\boldsymbol{\beta}}_{ELM}\right) = \frac{1}{\tilde{N}} E\left[(\widehat{\boldsymbol{\beta}} - \beta)^T (\widehat{\boldsymbol{\beta}} - \beta)\right] = \frac{\sigma^2}{\tilde{N}} \sum_{l=1}^{\tilde{N}} \frac{1}{\lambda_l}$: mean of scalar mean square error

where λ_l is the lth eigenvalue of the $\mathbf{H}^T\mathbf{H}$ matrix.

The singularity of the matrix $\mathbf{H}^T\mathbf{H}$ may make the solution impossible. Multicollinearity is one of the main reasons for this situation. Multicollinearity or noises can affect dramatically ELM in terms of generalization and stability performance. When there are highly correlated variables in a data set, the values of eigenvalues shrinks to zero. As eigenvalues decrease, two main properties of $\widehat{\boldsymbol{\beta}}_{ELM}$, $V\left(\widehat{\boldsymbol{\beta}}_{ELM}\right)$ and $SMSE\left(\widehat{\boldsymbol{\beta}}_{ELM}\right)$ increase. This means that the performance of estimator decays (Zhang et al., 2016).

There are some alternative approaches preferred to deal with these issues. In this section, we deal with these issues.

4.1 ELM Based on Ridge Estimator

Although the ELM algorithm is an effective method in terms of learning ability and speed, there have been many studies in the literature for dealing with the drawbacks like instability, poor generalizability and insufficient sparsity. The efforts for improving ELM were effective but not enough to be sure about obtaining the optimal solution of the estimation problems. One of the main reasons for this situation is the problem called the ill-conditioning in a data set. If the hidden layer output matrix is ill-conditioned, the stability and the generalization performance of ELM can be dramatically affected. As a solution to this problem, some methods based on the ridge estimator have been proposed (Li and Niu, 2013) in the context of ELM. In the study of Li and Niu (2013), although several algorithms based ridge-

type estimator have been proposed, the method for the selection of appropriate ridge tuning parameter is limited. In addition to the analytical method which is used in the study of Li and Niu (2013), some alternatives such as based on information criteria like AIC and BIC, or based on CV error should be considered to give better insights about the performance of the proposed algorithms. That's why, we have proposed new selection methods in the context of ELM to present these insights. In this section, the details of the our are given.

4.1.1 Ridge Type Estimators

In order to address the problem of ill-conditioned **H** matrix, ridge estimator of Hoerl and Kennard (1970) is defined in the content of ELM by Li and Niu (2013) as

$$\hat{\boldsymbol{\beta}}^{k}_{RR-ELM} = \left(\mathbf{H}^T\mathbf{H}+k\mathbf{I}_{\tilde{N}}\right)^{-1}\mathbf{H}^T\mathbf{T},\ k\geq \mathbf{0} \tag{4.3}$$

where k is ridge tuning parameter. If k is equal to zero, the ELM solution is obtained. If $k\mathbf{I}_{\tilde{N}}$ is replaced by a diagonal matrix $\mathbf{K} = diag(k_1, k_2, \ldots, k_{\tilde{N}})$, the generalized ridge estimator in the context of ELM is obtained as

$$\hat{\boldsymbol{\beta}}_{GR-ELM} = \left(\mathbf{H}^T\mathbf{H}+\mathbf{K}\right)^{-1}\mathbf{H}^T\mathbf{T},\ k\geq \mathbf{0}$$

where $k_1, k_2, \ldots, k_{\tilde{N}}$ are the ridge tuning parameters.

On the other hand, the AUR and generalized AUR estimators defined by Singh et al. (1986) are given by Li and Niu (2013) in the context of ELM respectively as

$$\hat{\boldsymbol{\beta}}^{k}_{AUR-ELM} = \left[\mathbf{I}-k^2\left(\ \mathbf{H}^T\mathbf{H}+k\mathbf{I}_{\tilde{N}}\right)^{-2}\right]\hat{\boldsymbol{\beta}}_{ELM},\ k\geq \mathbf{0} \tag{4.4}$$

and

$$\hat{\boldsymbol{\beta}}_{AUGR-ELM} = \left[\mathbf{I}-\left(\mathbf{H}^T\mathbf{H}+\mathbf{K}\right)^{-2}\mathbf{K}^2\right]\hat{\boldsymbol{\beta}}_{ELM}$$

Li and Niu (2013) have used only one approach to select the appropriate ridge tuning parameter. However, the performance of these algorithms depends on the value of the ridge tuning parameter. If it is determined properly, ridge-based ELM solutions may provide smaller MSE than the ELM solution. However, selecting the appropriate value of it is a complicated and tough problem. Since there is no single best method for selecting the ridge tuning parameter in linear regression, we can carry the same idea to the ELM. In the next section, some of these are presented.

4.1.2 Determination of Appropriate Ridge Tuning Parameter

The ridge and AUR esimtators depend on the parameter k. The selection of this parameter affects the performance of the estimators and this parameter gives information about the regularization. There are a number of methods for the selection of the parameter k. Some popular ways including both the analytical methods and alternative based on information criteria like AIC and BIC. The details for choosing the proper ridge tuning parameter are given in Section (3.1.2).

The methods which are suggested and widely used in the literature of linear regression analysis can be adapted to the context of ELM. The main difference about the usage in ELM is to calculate the degrees of freedom which is based on the quasi projection matrix. This matrix is defined as $\mathbf{H}\left(\mathbf{H}^T\mathbf{H}+k\mathbf{I}_{\tilde{N}}\right)^{-1}\mathbf{H}^T$ for ridge estimator and $\mathbf{H}\left[\mathbf{I}-k^2\left(\mathbf{H}^T\mathbf{H}+k\mathbf{I}_{\tilde{N}}\right)^{-2}\right]\mathbf{H}^T$ for the AUR estimator in the context of ELM (Martínez et al., 2011; Wieringen, 2015; Breheny, 2020).

The following algorithm presents the learning process of the proposed ELM algorithm based on different selection criteria for the appropriate tuning parameter:

Algorithm 4.1. *RR-ELM and AUR-ELM with Different Selection Criteria.*

Input: *Training set* $\{(x_i,t_i)\}$*, the number of the hidden neurons* $\{\tilde{N}\}$*, an activation function* $\{g(.)\}$*, the number of trials* $\{L\}$*, a sequence of* $\{k\}$ *values.*

Output: *The* β *outweight vector.*

1 Generate the hidden layer parameters $\mathbf{w}_i$ and b_i, $1 \leq i \leq \tilde{N}$, randomly.

2 Calculate the hidden layer output matrix $\{\mathbf{H}\}$ via Eq. (2.6) and obtain the ELM solution via Eq. (2.5).

3 Calculate the weights vectors by $\hat{\boldsymbol{\beta}}^{k}_{RR-ELM} = \left(\mathbf{H}^T\mathbf{H}+k\mathbf{I}_{\tilde{N}}\right)^{-1}\mathbf{H}^T\mathbf{T}$ and $\hat{\boldsymbol{\beta}}^{k}_{AUR-ELM} = \left[\mathbf{I}-k^2\left(\ \mathbf{H}^T\mathbf{H}+k\mathbf{I}_{\tilde{N}}\right)^{-2}\right]\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$ by Eq. (4.3) and Eq. (4.4).

4 **for** $1 \leq t \leq length(k)$ **do**

5 Calculate AIC and BIC criteria based on each t value by Eq. (3.11) and Eq. (3.12).

6 Calculate the CV error based on each t value.

7 **end**

8 Store the t^*_{AIC}, t^*_{BIC} and t^*_{CV} corresponding to the minimum AIC, BIC or CV error value among all possible t values, respectively.

9 Calculate the mean performance metrics of all (L) trials.

4.2 ELM Based on Liu Estimator

In the primary ELM algorithm, the output weights of the hidden layer are calculated by using traditional Moore-Penrose inverse. In the case of severe collinearity, the rank of the matrix $\mathbf{H}$ will be less than $\tilde{N}$. This case leads to inability to calculate the inverse of $\mathbf{H}^T\mathbf{H}$. Or if there exists serious multicollinearity, the ELM solution $\hat{\boldsymbol{\beta}}_{ELM}$ may have a large variance and the distance from the ELM estimates $\hat{\boldsymbol{\beta}}_{ELM}$ to the true parameters $\boldsymbol{\beta}$ may be large. That is, opposed to Huang et al. (2012), ELM does not have the smallest training error and also the smallest norm

of output weights in the presence of multicollinearity. As an alternative to the ELM estimator, different employable methods can be applied. Ridge estimator of Hoerl and Kennard (1970) is defined in the context of ELM by Li and Niu (2013). Although Li and Niu (2013) have directly modified the ridge regression idea to ELM, the underlying objective function of ridge estimator can be written as (see the study of Huang et al. (2012) for example):

$$(\mathbf{H}\boldsymbol{\beta}-\mathbf{T})^T(\mathbf{H}\boldsymbol{\beta}-\mathbf{T})+\frac{k}{2}\boldsymbol{\beta}^T\boldsymbol{\beta}. \tag{4.5}$$

$\widehat{\boldsymbol{\beta}}^{k}_{RR-ELM}$ given by Eq. (4.3) is the solution of the objective function in Eq. (4.5).

One of the parameters which should be tuned in the RR-ELM is the ridge tuning parameter (k). However, there is no consensus on the appropriate selection of the ridge tuning parameter. There has been extensive literature for determination of optimal (k) parameter. As a remedy to this problem, an alternative method called Liu estimator was proposed by Liu (1993). Similar to ridge estimator, Liu can deal with multicollinearity problem by using a different parameter on the learning process. The difference between Liu and ridge is the form of the tuning parameter. Although ridge includes a quadratic form of (k) tuning parameters, Liu offers a linear form of its (d) tuning parameter. The linear form makes Liu better and easier than the ridge in terms of calculations and speed. Therefore, Liu estimator can be considered as an alternative to the ridge estimator.

The problem given by Eq. (2.3) says that we are looking for an estimator of $\boldsymbol{\beta}$ which minimizes the error sum of squares. The problem in Eq. (4.5) looks for an estimator having the minimum error sum of squares in a class of estimators of $\boldsymbol{\beta}$ whose length is equal to a constant, say c

$$\text{minimize } (\mathbf{H}\boldsymbol{\beta}-\mathbf{T})^T(\mathbf{H}\boldsymbol{\beta}-\mathbf{T}), \text{ subject to } \boldsymbol{\beta}^T\boldsymbol{\beta}=c.$$

Therefore, ridge estimator confines the length of the estimator which is far away from the true parameters in the case of multicollinearity. However, an estimator whose distance to a given point, say prior information, rather than the origin can be preferable because the origin is a biased estimate of the true parameters. If we want to search for an estimator that minimizes the error sum of squares in a class of estimators of $\boldsymbol{\beta}$ which are equal distance from $d\hat{\boldsymbol{\beta}}_{ELM}$ where $\hat{\boldsymbol{\beta}}_{ELM}$ is as given by Eq. (2.5) and d is a constant in the interval $(0,1)$, we write the objective function as

$$(\mathbf{H}\boldsymbol{\beta}-\mathbf{T})^T(\mathbf{H}\boldsymbol{\beta}-\mathbf{T})+\left(d\hat{\boldsymbol{\beta}}_{ELM}-\boldsymbol{\beta}\right)^T\left(d\hat{\boldsymbol{\beta}}_{ELM}-\boldsymbol{\beta}\right)$$

where $0<d<1$ is a parameter called as Liu tuning parameter. The Liu tuning parameter d shrinks each element of $\hat{\boldsymbol{\beta}}$ by the same d value; that is, the norm of $d\hat{\boldsymbol{\beta}}_{ELM}$ is less than the norm of $\hat{\boldsymbol{\beta}}_{ELM}$ for $0<d<1$.

Since the objective functions of the ridge and Liu regression are different from each other, their ability to learn the inherent patterns in the data will be different. Hence, algorithms based on ridge and Liu will act in a different way which will affect the generalization performance.

As a result of some basic mathematical operations, we obtain the estimator as

$$\hat{\boldsymbol{\beta}}^d_{Liu-ELM}=\left(\mathbf{H}^T\mathbf{H}+\mathbf{I}_{\tilde{N}}\right)^{-1}\left(\mathbf{H}^T\mathbf{T}+d\hat{\boldsymbol{\beta}}_{ELM}\right),\ 0<d<1. \tag{4.6}$$

Since the estimator in Eq.(4.6) is the analog form of the Liu estimator in linear regression model, we call $\hat{\boldsymbol{\beta}}^d_L$ as the ELM based on Liu (L-ELM) in the context of machine learning. Here, d is called as the Liu tuning parameter.

Equivalent to Eq.(4.6), L-ELM can be written as a linear transformation of ELM defined by Eq. (2.5) as

$$\hat{\boldsymbol{\beta}}^d_{Liu-ELM}=\left(\mathbf{H}^T\mathbf{H}+\mathbf{I}_{\tilde{N}}\right)^{-1}\left(\mathbf{H}^T\mathbf{H}+d\mathbf{I}_{\tilde{N}}\right)\hat{\boldsymbol{\beta}}_{ELM},\ 0<d<1. \tag{4.7}$$

One property of $\hat{\boldsymbol{\beta}}^{d}_{Liu-ELM}$ is that it can be written as a convex combination of $\hat{\boldsymbol{\beta}}_{ELM}$ and $\hat{\boldsymbol{\beta}}^{k}_{R}$ for $k=1$ which are given by Eq. (2.5) and Eq. (4.3), respectively

$$\hat{\boldsymbol{\beta}}^{d}_{Liu-ELM} = d\hat{\boldsymbol{\beta}}_{ELM} + (1-d)\hat{\boldsymbol{\beta}}^{(k=1)}_{R}.$$

This property does the estimator to trace a path between $\hat{\boldsymbol{\beta}}_{ELM}$ and $\hat{\boldsymbol{\beta}}^{(k=1)}_{R}$ as d goes from 1 to 0.

Another property of $\hat{\boldsymbol{\beta}}^{d}_{Liu-ELM}$ is that it is in the linear form of d. Hence, computing $\hat{\boldsymbol{\beta}}^{d}_{Liu-ELM}$ is much easier and faster than $\hat{\boldsymbol{\beta}}^{k}_{RR-ELM}$. From the view of machine learning, speed and easy computability provide $\hat{\boldsymbol{\beta}}^{d}_{Liu-ELM}$ superiority. We here propose the L-ELM algorithm for the case of single-output regression problem, but it can be modified to solve multi-output regression and classification problems.

4.2.1 Selection Criteria for Liu Tuning Parameter

The selection of Liu tuning parameter has an important role in the performance. Various approaches have been proposed in the literature of linear regression models. Following Liu (1993), we here define three estimates of Liu tuning parameter.

i) As the minimizer of the scalar mean square error of $\hat{\boldsymbol{\beta}}^{d}_{Liu-ELM}$, we give

$$\hat{d_1} = 1 - \hat{\sigma}^2 \left[\left(\sum_{i=1}^{\tilde{N}} \frac{1}{\lambda_i(\lambda_i+1)} \right) / \left(\sum_{i=1}^{\tilde{N}} \frac{\hat{\alpha}_i^2}{(\lambda_i+1)^2} \right) \right] \tag{4.8}$$

where $\lambda_1, \lambda_2, ..., \lambda_{\tilde{N}}$ are the eigenvalues of the $\mathbf{H}^T\mathbf{H}$ matrix. $\hat{\alpha}_i$ is the ith element of $\widehat{\alpha} = \mathbf{P}^T\hat{\boldsymbol{\beta}}_{ELM}$ where $\hat{\boldsymbol{\beta}}_{ELM}$ is as given in Eq. (2.5) and $\mathbf{P}_{\tilde{N}x\tilde{N}}$ is the orthogonal matrix whose columns correspond to the eigenvectors of $\mathbf{H}^T\mathbf{H}$. $\hat{\sigma}^2$ is the variance estimator of model errors. $\hat{d_1}$ in Eq. (4.8) can take values outside the interval $(0,1)$. Therefore,

a generalized form of Eq. (4.8) can be written as

$$\hat{d}_1 = 1 - h\hat{\sigma}^2 \left[\left(\sum_{i=1}^{\tilde{N}} \frac{1}{\lambda_i (\lambda_i + 1)} \right) / \left(\sum_{i=1}^{\tilde{N}} \frac{\hat{\alpha}_i^2}{(\lambda_i + 1)^2} \right) \right] \tag{4.9}$$

where $h > 0$ is an arbitrary constant which satisfies the following inequality $0 < h < h_1$ modified from Liu (1993) as

$$0 < h < \frac{2(n-p)}{n-p+2} \left(1 - \frac{2}{\sum_{i=1}^{\tilde{N}} \frac{\lambda_{\tilde{N}}(\lambda_{\tilde{N}}+1)}{\lambda_i(\lambda_i+1)}} \right) \tag{4.10}$$

where $\lambda_{\tilde{N}}$ is the largest eigenvalue of $\mathbf{H}^T\mathbf{H}$.

ii) As the minimizer of the $C_p = \frac{SS_{Res,d}}{\hat{\sigma}^2} + 2tr(\mathbf{M}_d) - (n-2)$ statistic, we give

$$\hat{d}_2 = 1 - \hat{\sigma}^2 \left[\left(\sum_{i=1}^{\tilde{N}} \frac{1}{(\lambda_i + 1)} \right) / \left(\sum_{i=1}^{\tilde{N}} \frac{\lambda_i \hat{\alpha}_i^2}{(\lambda_i + 1)^2} \right) \right] \tag{4.11}$$

where $M_d = \mathbf{H}\left(\mathbf{H}^T\mathbf{H} + \mathbf{I}_{\tilde{N}}\right)^{-1}\left(\mathbf{H}^T\mathbf{H} + d\mathbf{I}_{\tilde{N}}\right)\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T$ is the quasi-projection matrix and $SS_{Res,d}$ is the residual sum of squares using $\hat{\boldsymbol{\beta}}^d_{Liu-ELM}$.

iii) As the minimizer of the generalized cross validation

$$GCV(d) = \frac{SS_{Res,d}}{[n - 1 - tr(M_d)]^2} \tag{4.12}$$

statistic, we define $\hat{d}_3$ which is exactly same with $\hat{d}_2$.

We give the following steps for the L-ELM algorithm:

Algorithm 4.2. *L-ELM.*

Input: *Training set* $\{(x_i, t_i)\}$*, the number of the hidden neurons* $\{\tilde{N}\}$*, an activation function* $\{g(.)\}$*, the number of trials* $\{L\}$*.*

Output: *The* β *outweights vector.*

1 Generate the hidden layer parameters $\mathbf{w}_i$ and b_i, $1 \leq i \leq \tilde{N}$, randomly.

2 Calculate the hidden layer output matrix $\{\mathbf{H}\}$ via Eq. (2.6) and obtain the ELM solution via Eq. (2.5).

3 Find the appropriate value of Liu tuning parameter $\left(\hat{d}\right)$ by using Eq. (4.9) or Eq. (4.11), respectively.

4 Calculate the weights by $\hat{\boldsymbol{\beta}}^{d}_{Liu-ELM} = \left(\mathbf{H}^T\mathbf{H} + \mathbf{I}_{\tilde{N}}\right)^{-1}\left(\mathbf{H}^T\mathbf{H} + d\mathbf{I}_{\tilde{N}}\right)\hat{\boldsymbol{\beta}}_{ELM}$.

5 Obtain the mean performance metrics of all (L) trials.

4.3 ELM Based on r-k Class Estimator

4.3.1 The ELM algorithm based on principal components regression

In this section, we consider the PCR to deal with the undesirable affects of multicollinearity on ELM. Kim et al. (2007), Singh et al. (2012), Zhao et al. (2012) and Li et al. (2012) studied the PCA in the context of ELM for reducing the dimension of sampling data. The PCA is generally applied for three purposes: (1) reducing the dimension of data set, (2) identifying the highly correlated variables and (3) making visualization possible. PCA is a powerful and well-known statistical technique to reduce the number of dimensions by creating new variables referring to the linear form of highly correlated variables. The explained variance by new variables, i.e PCs, is as high as the amount by the correlated variables and the new dimension is much less than the former and from the statistical view, there is no significant information loss as long as the number of PCs is enough. Zhang et al. (2016) proposed an algorithm called PCA-ELM which is based on reducing the dimension of $\mathbf{H}$ matrix to create a new and the full column ranked $\mathbf{H}$ matrix. The estimated weight vector $\widehat{\boldsymbol{\beta}}$ in PCA-ELM algorithm is defined as follows:

$$\hat{\boldsymbol{\beta}}_{PCA-ELM} = \left(\mathbf{H}^{*T}\mathbf{H}^{*}\right)^{-1}\mathbf{H}^{*T}\mathbf{T} \tag{4.13}$$

where $\mathbf{H}^{*}_{Nxr} = \mathbf{H}_{Nx\tilde{N}}\mathbf{P}^{*}_{\tilde{N}xr}$ is the new transformed hidden layer matrix. In the calculation of $\mathbf{H}^{*}$ matrix, r is the number of PCs retained in the model, $\mathbf{P}^{*}_{\tilde{N}xr}$ is a matrix consisting of first r eigenvectors for significant eigenvalues of $\mathbf{H}^{T}\mathbf{H}$ matrix. The steps of the PCA-ELM algorithm can be found in the study of Zhang et al. (2016).

In PCA-ELM proposed by Zhang et al. (2016), the reduced form of $\mathbf{H}$ hidden layer output matrix is used to calculate $\hat{\boldsymbol{\beta}}_{PCA-ELM}$. As a result, the size of the obtained weight vector is equal to the number of significant components. Unlike PCA-ELM, the PCR estimator is based on a slightly different calculation. Although PCR also uses PCs, the estimates of weight vector is found by reducing the number of components but the size of it is constant. This difference can be important in practice. Therefore, the r-k class estimator including the PCR estimator is proposed in the following section.

4.3.2 Proposed Algorithm: ELM Based on the R-K Class Estimator

In this section, we propose a new ELM algorithm which is based on the r-k class estimator and the selection of optimal parameters is discussed.

The r-k class estimator was proposed by Baye and Parker (1984) to eliminate the performance loss in the presence of multicollinearity in the data set. This estimator is a general form of the most well-known three estimators in linear regression: OLS, ridge and PCR estimators.

Let $\mathbf{P}$ is an orthogonal matrix whose columns correspond to the eigenvectors of $\mathbf{H}^{T}\mathbf{H}$ with $\mathbf{H}^{T}\mathbf{H} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}^{T}$ where $\boldsymbol{\Lambda}$ is a diagonal matrix with eigenvalues, $\lambda_1, ..., \lambda_{\tilde{N}}$, of the $\mathbf{H}^{T}\mathbf{H}$ matrix. Then each column of $\mathbf{HP}$ stands for the

PCs. The PCR of $\boldsymbol{\beta}$ in linear regression is obtained by deleting one or more PCs. Thus, we partition $\mathbf{P}$ matrix as $\mathbf{P}=[\mathbf{P}_r\mathbf{P}_{\tilde{N}-r}]$ where $\mathbf{P}_r=[p_1,p_2,\ldots,p_r]$ is the remaining columns of $\mathbf{P}$ having deleted $\tilde{N}-r$ columns with $r\leq\tilde{N}$. Thus, $\mathbf{P}_r^T\mathbf{H}^T\mathbf{H}\mathbf{P}_r=\boldsymbol{\Lambda}_r$=$diag\,(\lambda_1,...,\lambda_r)$ and $\mathbf{P}_{\tilde{N}-r}^T\mathbf{H}^T\mathbf{H}\mathbf{P}_{\tilde{N}-r}=\boldsymbol{\Lambda}_{\tilde{N}-r}$=$diag\,(\lambda_{r+1},...,\lambda_{\tilde{N}})$ where $\mathbf{P}_{\tilde{N}-r}=[p_{r+1},p_{r+2},\ldots,p_{\tilde{N}}]$. Then, Eq. (2.2) becomes

$$\mathbf{H}\boldsymbol{\beta}=\mathbf{H}\mathbf{P}\mathbf{P}^T\boldsymbol{\beta}=\mathbf{H}\mathbf{P}_r\mathbf{P}_r^T\boldsymbol{\beta}+\mathbf{H}\mathbf{P}_{\tilde{N}-r}\mathbf{P}_{\tilde{N}-r}^T\boldsymbol{\beta}. \tag{4.14}$$

Omitting the term $\mathbf{H}\mathbf{P}_{\tilde{N}-r}\mathbf{P}_{\tilde{N}-r}^T\boldsymbol{\beta}$, we get

$$\mathbf{H}\mathbf{P}_r\mathbf{P}_r^T\boldsymbol{\beta}=\boldsymbol{T}. \tag{4.15}$$

Under the model (4.15), we apply the ELM idea and get

$$\widehat{\boldsymbol{\beta}}_r=\mathbf{P}_r\left(\mathbf{P}_r^T\mathbf{H}^T\mathbf{H}\mathbf{P}_r\right)^{-1}\mathbf{P}_r^T\mathbf{H}^T\mathbf{T} \tag{4.16}$$

which is the PCR-ELM estimator. When we apply the ridge idea to the model (4.15), we obtain

$$\widehat{\boldsymbol{\beta}}_r(k)=\mathbf{P}_r\left(\mathbf{P}_r^T\mathbf{H}^T\mathbf{H}\mathbf{P}_r+k\mathbf{I}_r\right)^{-1}\mathbf{P}_r^T\mathbf{H}^T\mathbf{T},\ k>0. \tag{4.17}$$

Since this estimator carries the idea of the r-k class estimator of Baye and Parker (1984), we call $\widehat{\boldsymbol{\beta}}_r(k)$ as the r-k class-ELM (abbreviated as RK-ELM) estimator.

As previously mentioned, the corresponding ELM algorithms based on each of three well-known estimators can be obtained from the RK-ELM estimator as:

- $\widehat{\boldsymbol{\beta}}_{\tilde{N}}(0)=\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$ is equivalent to the preliminary ELM algorithm proposed by Huang et al. (2006).
- $\widehat{\boldsymbol{\beta}}_{\tilde{N}}(k)=\widehat{\boldsymbol{\beta}}_{RR-ELM}(k)=\left(\mathbf{H}^T\mathbf{H}+k\mathbf{I}_{\tilde{N}}\right)^{-1}\mathbf{H}^T\mathbf{T}$ is equivalent to the ELM agorithm based on ridge estimator proposed by Li and Niu (2013).

- $\widehat{\boldsymbol{\beta}}_r(0) = \widehat{\boldsymbol{\beta}}_r = \mathbf{P}_r \left(\mathbf{P}_r^T \mathbf{H}^T \mathbf{H} \mathbf{P}_r\right)^{-1} \mathbf{P}_r^T \mathbf{H}^T \mathbf{T}$ corresponds to the our novel ELM algorithm based on the PCR estimator.

4.3.3 Parameter Selection for the Proposed Algorithm

Parameter selection in any algorithm has an important impact on the performance. In our proposed algorithm in Eq. (4.17), the number of hidden layer nodes $\left(\tilde{N}\right)$, the number of PCs (r) and the ridge tuning parameter (k) should be tuned and determined carefully.

Selection of the Number of Hidden Layer Nodes $\left(\tilde{N}\right)$

Instead of determining a fixed value of hidden layer neurons, error-minimized extreme learning machine (EM-ELM) proposed by Feng et al. (2009) is used to find the appropriate number of the hidden layer nodes. EM-ELM is based on updating hidden layer output matrix by adding nodes (one by one or in groups) randomly and calculating the weight vector incrementally. Depending on a certain threshold value for convergence, this process is continued. The details of the EM-ELM algorithm (growing the hidden layer matrix and fast way of updating weight vector) can be found in Feng et al. (2009).

Selection of the Number of PCs

In order to select the number of PCs (r), there are several ways. In this study, we utilize from the cumulative contribution rate (CCR) which is the ratio of explained variance by the first *r* PCs. Eigenvalues correspond to the measure of the variation keeping by each PC. Therefore, eigenvalues are used as a representation of

variance. CCR measure is defined as

$$CCR = \frac{\sum_{l=1}^{r} \frac{1}{\lambda_l}}{\sum_{l=1}^{\tilde{N}} \frac{1}{\lambda_l}} \geq \eta$$

where η is the pre-determined threshold value. The value of η depends on the structure of the data set, the type of study and field (Yan, 2006; Zhang et al., 2016). However, the ratios between %80 and %95 are generally used in practical studies (Joliffe, 1986). We have taken this threshold as %95 in calculations.

Selection of k

There are miscellaneous ways to choose the optimal value of the tuning parameter (k). The most commonly used methods rely on minimizing SMSE of that estimator. Hoerl and Kennard (1970), Hoerl et al. (1975), Kibria (2003), etc. utilized from this idea and gave several estimators of tuning parameters for the ridge estimator. We here use the same approach, the SMSE of RK-ELM is used to determine the optimum k value. The SMSE of $\widehat{\boldsymbol{\beta}}_r(k)$ is

$$\begin{aligned} SMSE(\widehat{\boldsymbol{\beta}}_r(k)) &= E\left[(\widehat{\boldsymbol{\beta}}_r(k)-\boldsymbol{\beta})^T(\widehat{\boldsymbol{\beta}}_r(k)-\boldsymbol{\beta})\right] \\ &= \frac{1}{\tilde{N}}\left(\sum_{l=1}^{r}\frac{k^2\alpha_l^2+\sigma^2\lambda_l}{(\lambda_l+k)^2}+\sum_{l=r+1}^{\tilde{N}}\alpha_l^2\right). \end{aligned}$$

The derivative of $SMSE(\widehat{\boldsymbol{\beta}}_r(k))$ with respect to k,

$$\frac{dSMSE(\widehat{\boldsymbol{\beta}}_r(k))}{dk} = 2\sum_{l=1}^{r}\frac{\lambda_l}{(\lambda_l+k)^3}(\sigma^2-k\alpha_l^2)$$

shows that $SMSE(\widehat{\boldsymbol{\beta}}_r(k))$ is decreasing for $k < \frac{\sigma^2}{\alpha_{\max}^2}$ where $\alpha = P^T\boldsymbol{\beta}$. Then, an appropriate k value can be selected as

$$\hat{k}_1 = \frac{\hat{\sigma}^2}{\hat{\alpha}_{\max}^2 - \frac{\hat{\sigma}^2}{\lambda_{\max}}} \tag{4.18}$$

where $\hat{\sigma}^2$ is the residual mean squares and $\hat{\alpha}_{\max}$ is the maximum element of the vector $\hat{\alpha} = P^T \hat{\boldsymbol{\beta}}_r$. $\hat{\sigma}^2$ is an unbiased estimate of σ^2 and corresponds the residual sum of squares using PCR-ELM solution. $\hat{\alpha}^2_{\max} - \frac{\hat{\sigma}^2}{\lambda_{\max}}$ is an unbiased estimate of $\alpha^2_{\max}$.

The $k_l = \frac{\hat{\sigma}^2}{\alpha_l^2}$ value for all $l = 1,2,...,r$ minimizes $SMSE(\widehat{\boldsymbol{\beta}}_r(k))$. Then by following Hoerl and Kennard (1975) and Kibria (2003), we here respectively give the following estimators of tuning parameter for the RK-ELM algorithm:

$\hat{k}_2 = \frac{r\hat{\sigma}^2}{\widehat{\boldsymbol{\beta}}_r' \widehat{\boldsymbol{\beta}}_r}$ as the harmonic mean of k_l,

$\hat{k}_3 = \frac{\hat{\sigma}^2}{\left(\prod_{l=1}^r \widehat{\alpha}_l^2\right)^{1/r}}$ as the geometric mean of k_l.

Algorithm 4.3. *RK-ELM.*

Input: *Training set* $\{(x_i,t_i)\}$, *an initial large number of the hidden neurons* $\{\tilde{N}\}$, *an activation function* $\{g(.)\}$, *the CCR threshold value* $\{\eta\}$, *the number of trials* $\{L\}$.

Output: *The* β *outweights vector.*

1 Generate the hidden layer parameters $\mathbf{w}_i$ and b_i, $1 \leq i \leq \tilde{N}$, randomly.

2 Calculate the initial hidden layer output matrix $\{\mathbf{H}\}$ based on Eq. (2.6) predefined $\tilde{N}$ value.

3 Find $\mathbf{H}_{Best}$ corresponding to the optimal size of $\{\mathbf{H}\}$ matrix calculated via the EM-ELM algorithm.

4 Obtain the first r principal components explaining the variance by the ratio of η in $\mathbf{H}_{Best}$.

5 Find the appropriate value of ridge biasing parameter(k) by using $\hat{k}_1$, $\hat{k}_2$ or $\hat{k}_3$.

6 Calculate the weights vector by $\widehat{\boldsymbol{\beta}}_r(k) = \mathbf{P}_r \left(\mathbf{P}_r^T \mathbf{H}^T \mathbf{H} \mathbf{P}_r + k\mathbf{I}_r\right)^{-1} \mathbf{P}_r^T \mathbf{H}^T \mathbf{T}$.

7 Obtain the mean performance metrics of all (L) trials.

4.4 ELM Based on Liu and Lasso Estimators

Ridge estimator significantly improves the performance of Lasso within the elastic net training process. In this study, we propose a new regularization and variable selection method which is based on Liu and Lasso methods. In the following section, we present some details of Liu estimator and the form of the proposed method called Liu-Lasso extreme learning machine (LL-ELM).

4.4.1 The Proposed Algorithm: LL-ELM

Liu and ridge deal with multicollinearity by shrinking coefficient with a tuning parameter and provide more stable and generalizable results than classical ELM model. Although both estimators are effective on accounting the variables with high correlations, Liu estimator is faster and easier than ridge in terms of selecting tuning parameters because of having linear form of tuning parameter. The objective

function of Liu estimator can be also defined (Yıldırım and Özkale, 2020) as

$$\hat{\boldsymbol{\beta}}_{Liu-ELM} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2 + \left\| d\hat{\boldsymbol{\beta}} - \boldsymbol{\beta} \right\|_2 \right\},\ 0 < d < 1 \tag{4.19}$$

where d refers to the Liu tuning parameter.

Similar to elastic net method, the solution of Eq. (4.19) can be obtained via an augmented data set which is defined as follows:

$$\tilde{\mathbf{H}} = \begin{pmatrix} \mathbf{H} \\ \mathbf{I}_{\tilde{N}} \end{pmatrix} \text{ and } \tilde{\mathbf{T}} = \begin{pmatrix} \mathbf{T} \\ d\hat{\boldsymbol{\beta}}_{ELM} \end{pmatrix}. \tag{4.20}$$

Then the Liu estimator in Eq. (4.19) can be redefined in augmented form as

$$\hat{\boldsymbol{\beta}}^{d}_{Liu-ELM} = \arg\min_{\boldsymbol{\beta}} \left\{ \left\| \tilde{\mathbf{H}}\boldsymbol{\beta} - \tilde{\mathbf{T}} \right\|_2 \right\}. \tag{4.21}$$

When multicollinearity exists, the ELM estimates often have low bias but large variance which results in prediction difficulty. Besides, when there exist a large number of nodes, interpolation problem occurs with the ELM estimates. Shrinkage estimation methods and variable selection methods are the standard techniques for improving the ELM in these cases. RR-ELM and Liu-ELM are such shrinkage estimation methods. Although RR-ELM and Liu-ELM give more stable results, they do not set any weight to zero and do not give an easily interpretable model. Lasso-ELM was proposed by (Miche et al., 2011; Martinez et al., 2011) as a competitor to RR-ELM which shrinks some weights and sets others to zero. Similar to RR-ELM, Liu-ELM shrinks the weights resulting in good prediction accuracy but does not select weights; in other words does not set some weights to zero. To obtain more interpretable estimates, we here aim to combine the idea of Lasso-ELM and Liu-ELM and propose a new algorithm called Liu-Lasso ELM (LL-ELM). The objective

function of our proposed method is as

$$\hat{\boldsymbol{\beta}}_{LL-ELM} = \arg\min_{\boldsymbol{\beta}} \left\{ \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2 + \left\| d\hat{\boldsymbol{\beta}} - \boldsymbol{\beta} \right\|_2 + \lambda \|\boldsymbol{\beta}\|_1 \right\}. \tag{4.22}$$

With the sparsity property of Lasso, LL-ELM can provide more stable, generalizable and faster results than its competitors.

The solution of the proposed method can also be calculated via the augmented data as

$$\hat{\boldsymbol{\beta}}_{LL-ELM} = \arg\min_{\boldsymbol{\beta}} \left\{ \left\| \tilde{\mathbf{H}}\boldsymbol{\beta} - \tilde{\mathbf{T}} \right\|_2 + \lambda \|\boldsymbol{\beta}\|_1 \right\}. \tag{4.23}$$

By defining the proposed method (LL-ELM) in Eq. (4.23), the problem is reduced to a Lasso problem. So that similar to the approach of elastic net, LARS-EN algorithm (Zou and Hastie, 2005) can be used to estimate the $\boldsymbol{\beta}$. In our study, we adopted the approach of Sjöstrand et al. (2018) which is based the LARS algorithm (Efron et al., 2004) and the algorithm with piecewise linear regularization path proposed by Rosset and Zhu (2007). The details of these algorithms are given as Algorithm 3.2 and Algorithm 3.3 in Section (3.8.1).

4.4.2 Parameter and Model Selection

It is clear that the selection of Liu tuning parameter is effective on the performance on the performance of Liu-ELM and LL-ELM. In the context of ELM, Yıldırım and Özkale (2020) proposed $\hat{d}_1$ and $\hat{d}_2$ which are given respectively by Eqs. (4.9) and (4.11).

In Liu-ELM (a.k.a L-ELM), $\hat{d}_1$ and $\hat{d}_2$ values given in Eq. (4.9) and Eq. (4.11) are obtained using training data and used for measuring testing performance. For the proposed algorithm (LL-ELM), same $\hat{d}_1$ and $\hat{d}_2$ values with Liu-ELM can be

used for the initial parameters. For each fixed $\hat{d_1}$ and $\hat{d_2}$, the λ parameter is needed to be tuned carefully. There are various ways like AIC, BIC and CV to determine it. All of these methods have been widely used in the literature. Among these methods, BIC tends to produce more parsimonious models (i.e. more compact). This property of BIC may guarantee an optimal model instead of underfitted or overfitted one. The optimal λ value is determined via BIC which is defined as follows:

$$\boldsymbol{BIC}_t = \|\mathbf{T} - \mathbf{H}\boldsymbol{\beta}_t\|_2^2 + \log(N)\,\hat{\sigma}^2 L_t \tag{4.24}$$

where $\boldsymbol{\beta}_t$ is the tth model obtained with each possible combination of (d,λ), N is the size of training set, $\hat{\sigma}^2$ is mean of squares of residuals and L_t is the number of positive elements in $\boldsymbol{\beta}_t$ vector. The (d,λ) combination providing minimum BIC value is selected for each d value and an overall examination among all (d) values is carried out. The best (d,λ) pair is selected for final model and used for obtaining testing results. The following algorithm can be used for experiments:

Algorithm 4.4. *LL-ELM.*

Input: *Training set* $\{(x_i,t_i)\}$*, the maximum number of the hidden neurons* $\{\tilde{N}\}$*, an activation function* $\{g(.)\}$*, the number of trials* $\{L\}$*.*

Output: *The* β *weight matrix.*

1 Generate the initial parameters $\mathbf{w}_i$ and b_i, $1 \leq i \leq \tilde{N}$, randomly.

2 Calculate the hidden layer output matrix $\{\mathbf{H}\}$ via Eq. (2.6) and obtain the ELM solution via Eq. (2.5)

3 Find the optimal Liu parameters $\left\{\hat{d}\right\}$ via Eq. (4.9) or Eq. (4.11).

4 Solve the equation $\boldsymbol{\beta}_{LL-ELM} = \arg\min_{\boldsymbol{\beta}} \left\{ \left\| \tilde{\mathbf{H}}\boldsymbol{\beta} - \tilde{\mathbf{T}} \right\|_2^2 + \lambda \left\| \boldsymbol{\beta} \right\|_1 \right\}$ by using Algorithm 3.2 and 3.3.

5 for $1 \leq t \leq size\,(\beta)$ do

6 Calculate BIC(t) using each possible model using Eq. (4.24).

7 end

8 Find the t^* corresponding the minimum BIC value among all possible models of β vector.

9 Select the optimum weights vector as $\beta_{Best} = \beta\,(t^*)$.

5. NUMERICAL ANALYSIS

Numerical analysis process for each algorithm proposed with the scope of the thesis study is carried out by using different data sets. The structural properties of the data sets are important on the performance of the proposed algorithms. The learning (i.e. training) process for RR-ELM and L-ELM is similar and not sensitive to the type of scaling. However, the normalization for both inputs and outputs is suggested (Huang, 2020). The RK-ELM produces consistent results when the centering or standardization is applied to the data sets. The main advantages of this approach are to discriminate the effect of each components and to achieve a more balanced components in terms of variance explanation capability. Otherwise, the high level variability in a data set adversely affects to the results and cause relatively few components to be used. For LL-ELM, the effect of standardization is to avoid from the adverse effect of magnitude of each variable and to get more reasonable constraints which is also effective on the performance of the model. Standardization has a common usage for training Lasso based models and suggested (Hastie et al., 2009, 2015). Besides, the other reasons of these selections are to avoid the usage of the data sets including many categorical variables for testing RK-ELM and LL-ELM algorithms and to compare the algorithms under same conditions which they are firstly proposed.

5.1 Performance Evaluation of RR-ELM and AUR-ELM

In this section, the performance of different estimation algorithms (Yıldırım and Özkale, 2019) is investigated and compared with each other on several benchmark regression problems including eight data sets that have been widely known and

used in machine learning studies. These data sets, Servo, Abalone, Boston housing, Auto-MPG, Delta ailerons, Bank domains, House census(8L) and Delta elevators which are obtained from the University of California at Irvine (UCI) Machine Learning Repository (Dua and Graff, 2020) and a different source (Torgo, 2019). ELM, RR-ELM, and AUR-ELM algorithms have been applied to these data sets and they are compared to each other. Each data set, the inputs have been normalized into the interval [-1,1] and the outputs into the interval [0,1]. Each data set has been split into the training and testing data randomly. The size of each part is given in Table 5.1.

Table 5.1. The benchmark data sets for comparison of ridge-type ELM algorithms

Data sets	Attributes	Sample size	Training set	Testing set
Boston housing	13	506	250	256
Servo	4	167	80	87
Abalone	8	4177	2000	2177
Auto mpg	7	398	200	198
Delta ailerons	5	7129	3000	4129
Delta elevators	6	9517	4000	5517
Bank domains	8	8192	4500	3692
House census (8L)	8	22784	10000	12784

The experimental settings are followed by Li and Niu (2013). One hundred trials have been conducted for each data set. The number of hidden neurons is equal to 30 or 50 depending on the data set. The sigmoidal activation function is used for Servo, Abalone and Boston housing; sine activation function is used for Auto-MPG, Delta ailerons, Bank domains, California housing, House census(8L) and Delta elevators.

In every trial, the ridge tuning parameter is determined by minimizing the AIC, BIC and CV criteria. These criteria are applied to each of RR-ELM and AUR-ELM algorithms by considering RR and AUR estimators, respectively. The number of hidden neurons, the RMSE and the standard deviations of the predicted values for both training and testing data sets are given in Table 5.2.

According to the results in Table 5.2, it can be said that the choice of algorithms and methods for the determination of ridge tuning parameters are effective on the performance of training and testing. The processing time is longer than ELM because of considering all combinations of the estimation methods and criteria like AIC, BIC, and CV. However, the performance of ELM is improved especially in terms of testing performance in most data sets. In Figs. 5.1-5.3, we give the testing errors for several data sets including Auto Mpg, Boston and Abalone by using AIC, BIC, and CV criteria, respectively. The mean value of the ridge tuning parameter for each criterion is used for evaluation.

Table 5.2. Performance comparison of ridge-type ELM algorithms

Data Sets	Algorithms	#Hidden Nodes	Method	Training RMSE	Testing RMSE		k Value	
					Mean	Deviation	Mean	Deviation
	ELM	50	-	0,06805	0,11103	0,01405	-	-
Boston Housing	RR-ELM	50	AIC	0,06978	0,09955	0,01090	0,0086	0,0077
			BIC	0,07711	0,09868	0,00613	0,0620	0,0463
			CV	0,07195	0,09899	0,00859	0,0159	0,0138
	AUR-ELM	50	AIC	0,07347	0,09471	0,00832	0,6993	0,4255
			BIC	0,07227	0,09401	0,00635	0,7076	0,4321
			CV	0,07316	0,09449	0,00616	0,7223	0,4300
Servo	ELM	30	-	0,08002	0,11525	0,02056	-	-
	RR-ELM	30	AIC	0,07375	0,13046	0,02089	0,0741	0,0729
			BIC	0,07457	0,12967	0,02394	0,0717	0,0668
			CV	0,07800	0,13436	0,02366	0,1200	0,1057
	AUR-ELM	30	AIC	0,08432	0,11906	0,01345	0,1946	0,1434
			BIC	0,09348	0,12419	0,03687	0,2245	0,1463
			CV	0,08520	0,11857	0,03522	0,1360	0,0982
Abalone	ELM	30	-	0,07512	0,07847	0,00262	-	-
	RR-ELM	30	AIC	0,07537	0,07765	0,00143	0,0064	0,0091
			BIC	0,07604	0,07750	0,00083	0,0288	0,0137
			CV	0,07561	0,07749	0,00078	0,0091	0,0084
	AUR-ELM	30	AIC	0,07652	0,07695	0,00312	2,7210	1,4402
			BIC	0,07682	0,07721	0,00523	2,7378	1,4368
			CV	0,07592	0,07633	0,00133	1,6372	1,2753
Auto Mpg	ELM	30	-	0,07454	0,08985	0,00830	-	-
	RR-ELM	30	AIC	0,07674	0,08480	0,00500	0,4141	0,3956
			BIC	0,08104	0,08521	0,00498	1,4301	0,6162
			CV	0,07745	0,08456	0,00552	0,6339	0,4672
	AUR-ELM	30	AIC	0,07146	0,09203	0,00581	6,0758	2,6241
			BIC	0,07266	0,09289	0,00626	6,3213	2,6129
			CV	0,07122	0,09198	0,00620	4,3012	2,8983

Table 5.2. Continued

Data Sets	Algorithms	#Hidden Nodes	Method	Training RMSE	Testing RMSE		k Value	
					Mean	Deviation	Mean	Deviation
Delta Ailerons	ELM	30	-	0,03828	0,03968	0,00026	-	-
	RR-ELM	30	AIC	0,03834	0,03972	0,00029	0,0061	0,0062
			BIC	0,03836	0,03961	0,00022	0,0568	0,0540
			CV	0,03829	0,03963	0,00030	0,0248	0,0333
	AUR-ELM	30	AIC	0,03830	0,03966	0,00027	1,0434	0,5330
			BIC	0,03829	0,03968	0,00030	1,0434	0,5330
			CV	0,03822	0,03966	0,00023	0,9853	0,6031
Delta Elevators	ELM	30	-	0,05287	0,05557	0,00067	-	-
	RR-ELM	30	AIC	0,05284	0,05548	0,00076	0,4160	0,2810
			BIC	0,05295	0,05561	0,00066	0,5108	0,2809
			CV	0,05281	0,05548	0,00060	0,3784	0,2779
	AUR-ELM	30	AIC	0,05334	0,05548	0,00086	24,7555	15,7008
			BIC	0,05350	0,05559	0,00093	24,8080	15,6353
			CV	0,05329	0,05541	0,00083	21,8505	14,6620
Bank Domains	ELM	30	-	0,10553	0,11212	0,01627	-	-
	RR-ELM	30	AIC	0,10252	0,10887	0,01859	0,8594	0,5796
			BIC	0,10430	0,11055	0,01833	0,8594	0,5796
			CV	0,10287	0,10903	0,01683	0,9604	0,6079
	AUR-ELM	30	AIC	0,10140	0,10747	0,01951	117,4400	46,2350
			BIC	0,10327	0,10932	0,01871	116,3025	46,7128
			CV	0,10319	0,10957	0,01851	115,5925	44,3151
House Census	ELM	30	-	0,07500	0,07528	0,00175	-	-
	RR-ELM	30	AIC	0,07518	0,07533	0,00180	0,9336	0,7391
			BIC	0,07513	0,07520	0,00181	1,0104	0,7505
			CV	0,07470	0,07498	0,00156	0,9900	0,7655
	AUR-ELM	30	AIC	0,07554	0,07500	0,00152	5,0460	2,7851
			BIC	0,07546	0,07483	0,00168	5,0520	2,8080
			CV	0,07527	0,07475	0,00171	4,7730	2,7506

The results from Table 5.2 can be summarized as follows:

(i) RR-ELM for Servo, AUR-ELM for Auto Mpg and Bank Domains data sets perform better than classical ELM in terms of training performance (RMSE).

The performances for Abalone, Delta Ailerons and Delta Elevators in both algorithms are similar to ELM. ELM outperforms for the rest of the data sets than RR-ELM and AUR-ELM. When examined the results between RR-ELM and AUR-ELM, AUR-ELM provides better results than RR-ELM for Auto Mpg, Delta Ailerons, and Bank Domains. For Boston, Servo, Abalone, Delta Elevators and House Census (8L), RR-ELM is better than AUR-ELM.

(ii) RR-ELM and AUR-ELM algorithms perform better than ELM in terms of testing performance (RMSE) in all data sets with the exception of Servo. When the results between RR-ELM and AUR-ELM are examined, AUR-ELM provides better results than RR-ELM for Boston Housing (see Fig. 5.2), Delta Elevators, Bank Domains, and House Census (8L). For Auto Mpg (see Fig. 5.1) and Delta Ailerons, RR-ELM is better than AUR-ELM. The performances of RR-ELM and AUR-ELM algorithms are similar for the Abalone data set (see Fig. 5.3).

(iii) When the standard deviation of testing RMSE is considered individually, RR-ELM and AUR-ELM algorithms provide more stable results, in other words, less standard deviations results, than ELM for Boston, Abalone, Auto Mpg and House Census(8L). ELM performs better than RR-ELM and AUR-ELM for some data sets like Servo, Delta Ailerons, Delta Elevators and Bank Domains in terms of stability. When the results between RR-ELM and AUR-ELM are examined, AUR-ELM provides better results than RR-ELM for Boston, Bank Domains, and House Census(8L). For Abalone, Auto Mpg, Servo, Delta Ailerons and Delta Elevators, RR-ELM is better than AUR-ELM.

(iv) The criteria for choosing the ridge tuning parameter are considered individually in the data sets which are improved or similar with ELM in terms of training performance. According to the results, while AIC is better for Servo, Abalone, and Bank Domains, CV is better for Auto Mpg, Delta Ailerons, and Delta Elevators. For the data sets which are improved in terms of testing performance, AIC is better for Bank Domains, BIC is better for Boston Housing and Delta Ailerons, CV for Abalone, Auto Mpg, Delta Elevators and House Census(8L). By examining the standard deviations, it can be seen that AIC is the best for House Census(8L), BIC for Auto Mpg and Delta Ailerons and CV for Abalone, Delta Elevators and Bank Domains. Based on these results, it can be said that there is not any single best method for choosing the ridge tuning parameter. Also, the selection criterion is effective on the performance of training and testing data sets. Additionally, the range of ridge tuning parameter in the AUR-ELM algorithm is more stable and near-valued for each of the criteria than RR-ELM.

In order to calculate the percentages of improvement in RR-ELM and AUR-ELM with regard to the values obtained with ELM, the following equation is used:

$$Percentage = \frac{RMSE_{ELM} - RMSE_{RR-ELM}}{RMSE_{ELM}} \times 100$$

$$Percentage = \frac{RMSE_{ELM} - RMSE_{AUR-ELM}}{RMSE_{ELM}} \times 100$$

Figs. 5.4 and 5.5 show the changes of improvement. In these figures, it can be seen that the performance of RR-ELM and AUR-ELM depends on the value of the ridge tuning parameter. While RR-ELM outperforms to AUR-ELM in some cases, AUR-ELM is better than RR-ELM for other cases.

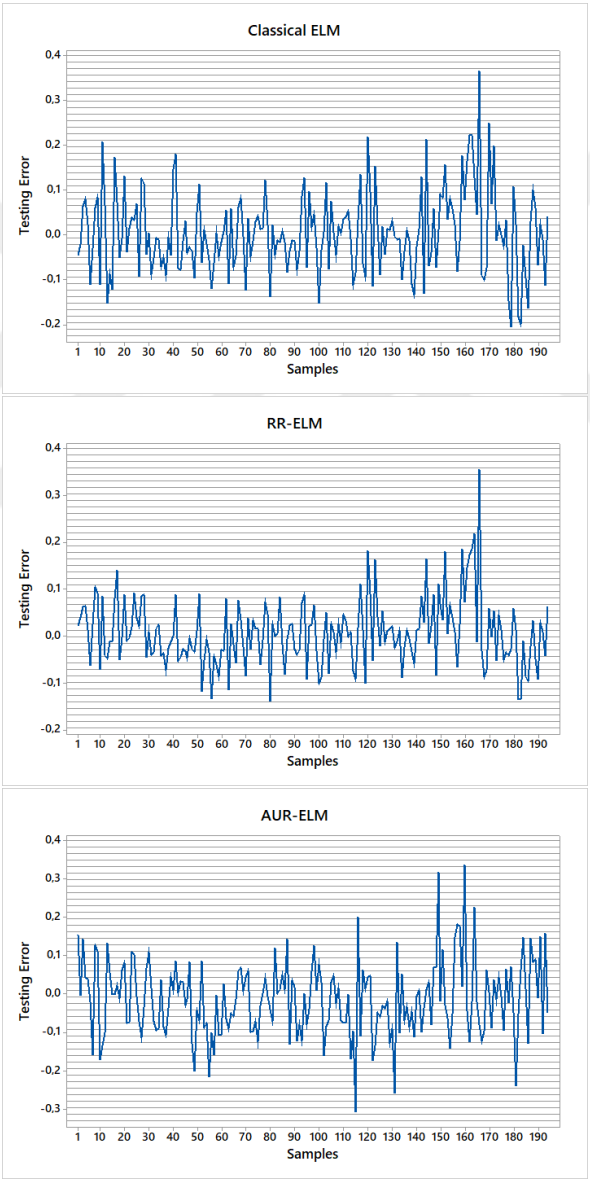

Figure 5.1. The testing errors for Auto mpg data set

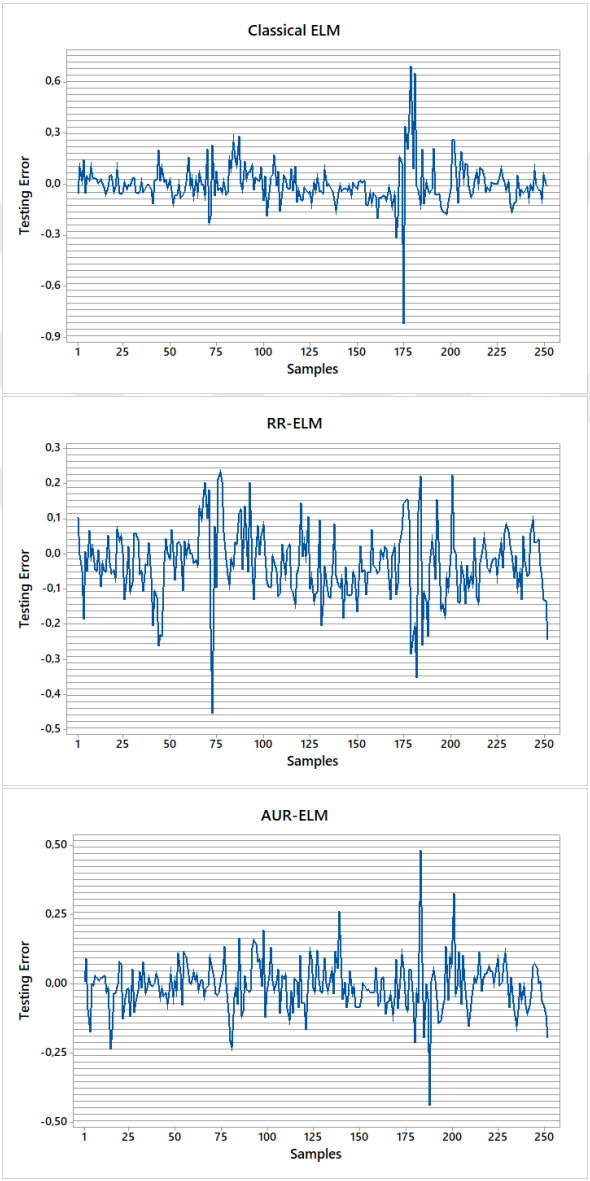

Figure 5.2. The testing errors for Boston housing data set

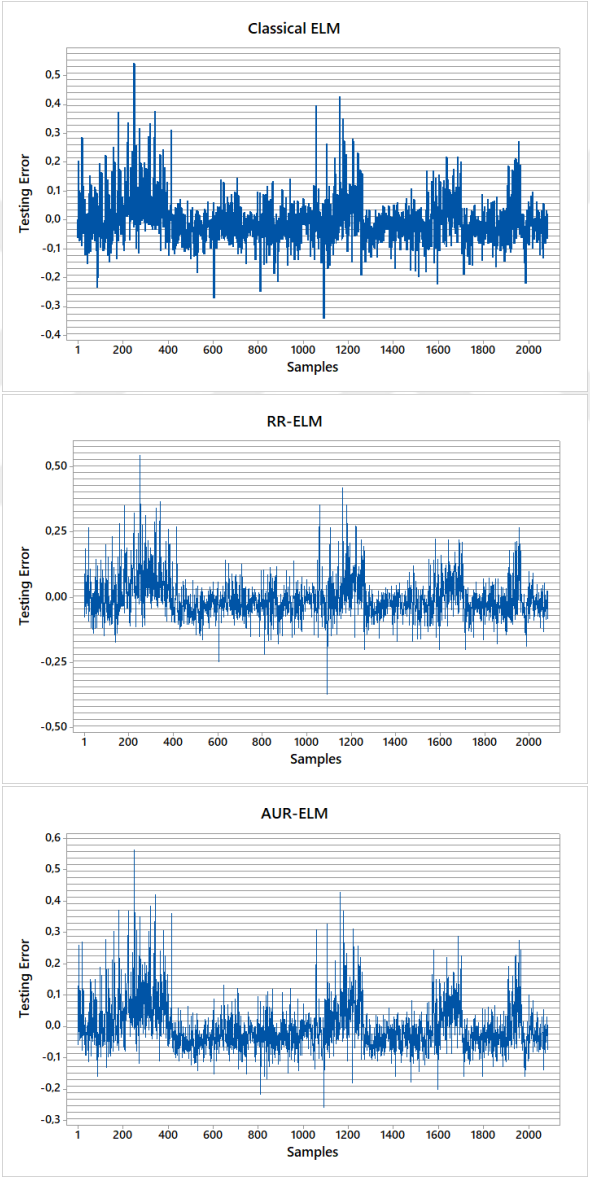

Figure 5.3. The testing errors for Abalone data set

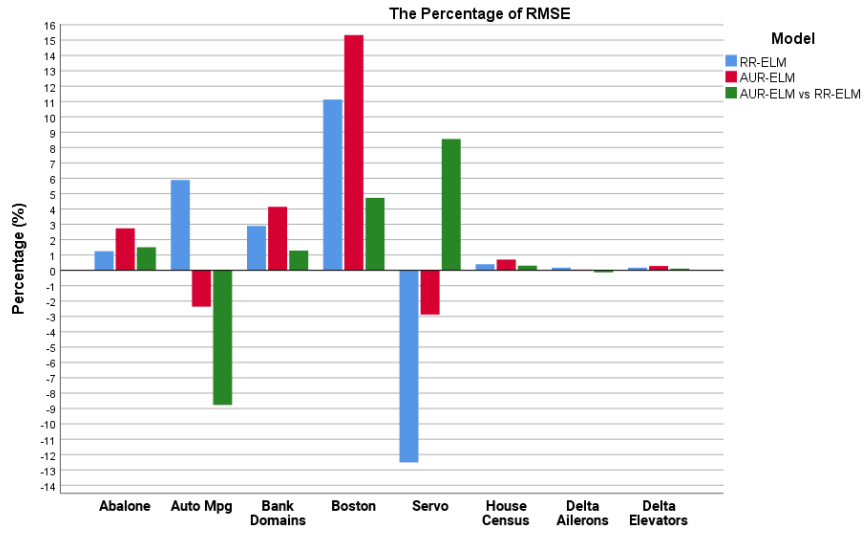

Figure 5.4. Comparison of models in terms of testing RMSE

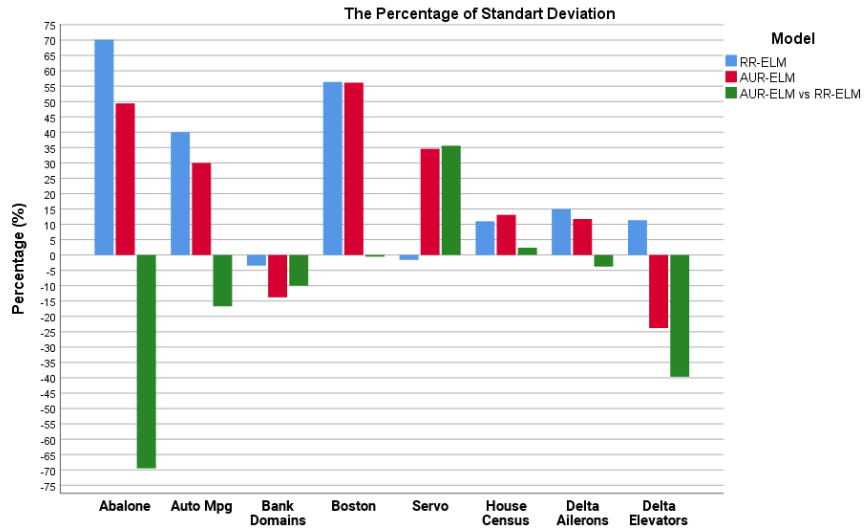

Figure 5.5. Comparison of models in terms of standard deviation of testing RMSE

5.2 Performance Evaluation of L-ELM

This section shows the performance results of the basic ELM and the proposed algorithm (Yıldırım and Özkale, 2020) by using several regression tasks. Nine benchmark data sets are chosen to compare the performance of these algorithms. The data sets are collected from two well-known sources, UCI machine learning repository (Dua and Graff, 2020) and Torgo (2019). As suggested by the author of the preliminary ELM (Huang, 2020), all inputs for each data set have been normalized into the interval [-1,1] and the outputs into the interval [0,1]. Each data set has been split to train and to test the results by %70 and %30, respectively. The properties of these data sets are given in Table 1.

Table 5.3. The benchmark data sets for Liu-based ELM algorithm

Data Sets	Attributes	Sample Size	Training Set	Testing Set
Boston Housing	13	506	356	150
Servo	4	167	119	48
Abalone	8	4177	2925	1252
Auto Mpg	7	398	282	116
Delta Ailerons	5	7129	4992	2137
Delta Elevators	6	9517	6664	2853
Bank Domains	8	8192	5736	2456
California Housing	8	20640	14449	6191
House Census (8L)	8	22784	15951	6833

The codes of each algorithm are written from the scratch based on the source codes provided by Huang et al. (2006) and evaluated in the R software environment running Core i7 CPU 2.30 GHz with 12 GB RAM. In order to reduce the effect

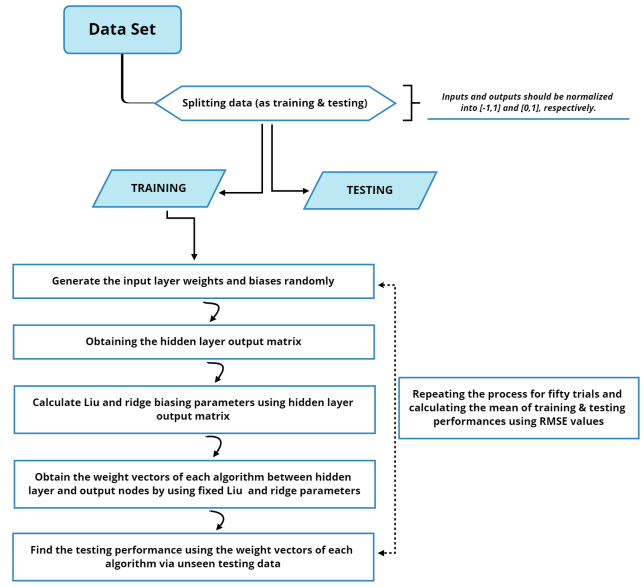

Figure 5.6. The flow chart of experimental process on training and testing phases for L-ELM

of randomness, fifty trials have been carried out each data set. The number of the hidden neurons is fixed and took as 50 for all different data sets. In OP-ELM, since the number of hidden nodes are different in each trial, the mean value of all trials has been reported in hidden nodes column of Table 5.4. The sigmoidal activation function is used for all data sets.

Two different Liu tuning parameters $\hat{d}_1$ and $\hat{d}_2$ are calculated for each trial and data set. Only for the Servo data set, Eq. (4.9) is used to calculate $\hat{d}_1$ because of the calculation of Eq.(4.8) gave a d value outside the interval $(0,1)$. The mean h value in the range is used in calculation $\hat{d}_1$ for the Servo data set. By using the

optimal $\hat{d}_1$ and $\hat{d}_2$ according to the training set, Liu regression models have been trained and tested. The k tuning parameter for RR-ELM and AUR-ELM is set as a fixed value and obtained by using $k = (\widetilde{N} \times \widehat{\sigma}^2) / \sum_{i=1}^{\widetilde{N}} \widehat{\alpha}_i{}^2$ which is exactly same with the proposed by Li and Niu (2013) as $k = (\tilde{N} \times \widehat{\sigma}^2) / \widehat{\boldsymbol{\beta}}' \widehat{\boldsymbol{\beta}}$.

The mean d values for each choice of Liu tuning parameter, the mean k values for the ridge tuning parameter and the corresponding mean of the RMSE values for the training and testing data have been reported. The standart deviations of testing RMSE have been calculated to evaluate the stability performance of each algorithm. The summary of the training and testing phases is given by Figure 5.6. Besides, all the aforementioned results are given in Table 5.4. To compare the performance of L-ELM over the other algorithms in percentage, the reduction rate is calculated as $RR = \frac{(any\ algorithm) - (L-ELM)}{(any\ algorithm)} \times 100$ and reported by Table 5.4.

When Table 5.4 is examined, the results can be summarised as follows:

- The mean of d values that are found as the best in each trial is reported. The type of Liu tuning parameters affects both the training and testing performances. According to the training performance, $\hat{d}_2$ is generally superior to $\hat{d}_1$ in the sense of providing smaller RMSE values, except Bank Domains, California Housing, and House Census data sets. When testing performance is examined, $\hat{d}_1$ is found as better than $\hat{d}_2$ for data sets except Boston Housing, Servo and Auto Mpg data sets.

- The training time of the basic ELM is lower than L-ELM in all data sets. The selection of the Liu tuning parameters affects the processing time. Likewise, this effect is variable depending on the type of Liu tuning parameters. There are not serious differences between the amount of time spent. While $\hat{d}_1$ takes

Table 5.4.: The performance results of ELM and L-ELM algorithms

Data set	Algorithm	#Hidden Nodes	d or k	Mean (d or k)	Training Time(sec)	RMSE (Train)	RMSE (Test)					
							Mean	*RR*(%) d_1	*RR*(%) d_2	*S.Dev.*	*RR*(%) d_1	*RR*(%) d_2
Boston	ELM	50	–	–	0.0093	0.07142	0.09586	−1.1370	2.9939	0.00729	27.2976	29.4924
Housing	L-ELM	50	d_1	0.64010	0.0505	0.07680	0.09695	–	4.0854	0.00530	–	3.0188
			d_2	0.93681	0.0521	0.07125	**0.09299**	−4.2585	–	**0.00514**	−3.1128	–
	RR-ELM	50	k	0.00498	0.0831	0.07603	**0.08643**	−12.1717	−7.5899	**0.00599**	11.5192	14.1903
	AUR-ELM	50	k	0.00498	0.0825	0.07552	0.09307	−4.1689	0.0859	0.00884	10.0452	41.8552
	OP-ELM	24.5	–	–	0.1395	0.11901	0.11712	17.2216	20.6028	0.00983	46.0834	47.7110
Servo	ELM	50	–	–	0.0062	0.05802	0.11682	0.0171	2.9703	0.01629	−73.3579	8.7170
	L-ELM	50	d_1	0.61739	0.0383	0.08304	0.11680	–	2.9537	0.02824	–	47.3441
			d_2	0.99960	0.0397	0.05763	0.11335	−3.0436	–	0.01487	−89.9126	–
	RR-ELM	50	k	0.00967	0.0671	0.08975	**0.11188**	−4.3975	−1.3139	**0.01078**	−161.967	−37.9406
	AUR-ELM	50	k	0.00967	0.0670	0.05537	0.13636	14.3443	16.8744	0.02975	5.0756	50.0168
	OP-ELM	35.5	–	–	0.0496	0.12255	0.14593	4.6920	7.5071	0.02217	−27.3793	32.9274
Abalone	ELM	50	–	–	0.0448	0.07382	0.08013	2.9452	−0.7862	0.00598	51.5050	−4.8495
	L-ELM	50	d_1	0.59479	0.1608	0.07459	**0.07777**	–	−3.8446	0.00290	–	−116.207
			d_2	0.99379	0.1441	0.07397	0.08076	3.7023	–	0.00627	53.7480	–
	RR-ELM	50	k	0.00085	0.1815	0.07337	0.08006	2.8603	−0.8743	**0.00289**	−0.3460	−116.955
	AUR-ELM	50	k	0.00085	0.1748	0.07315	0.08386	7.2621	3.6966	0.01019	71.5407	38.4690
	OP-ELM	32.3	–	–	0.8065	0.08086	0.22849	65.9635	64.6549	0.28481	98.9817	97.7985
Auto	ELM	50	–	–	0.0052	0.06290	0.08006	−3.3474	3.2100	0.00586	16.0409	24.2320
Mpg	L-ELM	50	d_1	0.23742	0.0452	0.07148	0.08274	–	6.3451	0.00492	–	9.7560
			d_2	0.95268	0.0471	0.06264	**0.07749**	−6.7750	–	0.00444	−10.8108	–
	RR-ELM	50	k	0.00376	0.0709	0.05702	0.08941	7.4600	13.3318	**0.00359**	−37.0474	−23.6769
	AUR-ELM	50	k	0.00376	0.0699	0.05527	0.09433	12.2866	17.8522	0.00508	3.1496	12.5984
	OP-ELM	28.3	–	–	0.0494	0.08162	0.09529	13.1703	18.6798	0.00743	33.7819	40.2422

Table 5.4. The performance results of ELM and L-ELM algorithms (Cont'ed)

Data set	Algorithm	#Hidden Nodes	*d or k*	Mean (*d or k*)	Training Time(sec)	RMSE (Train)	RMSE (Test)					
							Mean	*RR*(%) d_1	*RR*(%) d_2	*S.Dev.*	*RR*(%) d_1	*RR*(%) d_2
Delta	ELM	50	–	–	0.0786	0.03826	0.03813	0.8130	−0.2622	0.00027	18.5185	−37.037
Ailerons	L-ELM	50	d_1	0.78434	0.2656	0.03839	**0.03782**	–	−1.0840	**0.00022**	–	−68.1818
			d_2	0.99875	0.2279	0.03830	0.03823	1.0724	–	0.00037	40.5405	–
	RR-ELM	50	k	0.00006	0.2407	0.03772	0.03929	3.7414	2.6978	0.00025	12.0000	−48.0000
	AUR-ELM	50	k	0.00006	0.2489	0.03760	0.03960	4.4949	3.4595	0.00040	45.0000	7.5000
	OP-ELM	35.5	–	–	1.5327	0.03974	0.04125	8.3151	7.3212	0.00170	87.0588	78.2352
Delta	ELM	50	–	–	0.0954	0.05215	0.05463	0.3844	0.0549	0.00014	35.7142	0
Elevators	L-ELM	50	d_1	0.65952	0.3483	0.05226	**0.05442**	–	−0.3307	**0.00009**	–	−55.5556
			d_2	0.98068	0.3022	0.05219	0.05460	0.3296	–	0.00014	35.7142	–
	RR-ELM	50	k	0.00008	0.3007	0.05221	0.05454	0.2200	−0.1100	0.00016	43.75	12.5000
	AUR-ELM	50	k	0.00008	0.3129	0.05219	0.05452	0.1834	−0.1467	0.00011	18.1818	−27.2727
	OP-ELM	36.8	–	–	1.6218	0.05447	0.05511	1.2520	0.9254	0.00057	84.2105	75.4386
Bank	ELM	50	–	–	0.0844	0.05159	0.05138	0.7006	0.3114	0.00184	24.4565	8.1521
Domains	L-ELM	50	d_1	0.90925	0.2889	0.05120	**0.05102**	–	−0.3920	**0.00139**	–	−21.5827
			d_2	0.94901	0.2656	0.05140	0.05122	0.3904	–	0.00169	17.7514	–
	RR-ELM	50	k	0.00029	0.2667	0.05159	0.05239	2.6150	2.2332	0.00140	0.7142	−20.7143
	AUR-ELM	50	k	0.00029	0.2982	0.05127	0.05161	1.1431	0.7556	0.00151	7.9470	−11.9205
	OP-ELM	44.8	–	–	1.1314	0.12556	0.12546	59.3336	59.1742	0.00639	78.2472	73.5524
California	ELM	50	–	–	0.2046	0.13086	0.13197	0.3788	0.0530	0.00148	7.4324	20.2702
Housing	L-ELM	50	d_1	0.95081	0.7760	0.13070	**0.13147**	–	−0.3270	0.00137	–	13.8686
			d_2	0.99690	0.7153	0.13094	0.13190	0.3260	–	**0.00118**	−16.1017	–
	RR-ELM	50	k	0.00030	0.7322	0.13013	0.13222	0.5672	0.2420	0.00146	6.1643	19.1780
	AUR-ELM	50	k	0.00030	0.7687	0.13083	0.13286	1.0462	0.7225	0.00184	25.5434	35.8695
	OP-ELM	32.5	–	–	5.4219	0.15032	0.16173	18.7102	18.4443	0.03626	96.2217	96.7457
House	ELM	50	–	–	0.2269	0.06942	0.07074	1.0319	−0.5089	0.00122	27.0491	−49.1803
Census	L-ELM	50	d_1	0.97637	0.8150	0.06926	0.07001	–	−1.5569	**0.00089**	–	−104.494
			d_2	0.99691	0.8054	0.06963	0.07110	1.5330	–	0.00182	51.0989	–
	RR-ELM	50	k	0.00092	0.7666	0.06978	**0.06968**	−0.4735	−2.0378	0.00183	51.3661	0.5464
	AUR-ELM	50	k	0.00092	0.8480	0.06970	0.07048	0.6668	−0.8796	0.00260	65.7692	30.000
	OP-ELM	30.1	–	–	5.3775	0.08210	0.08121	13.7914	12.4492	0.00314	71.6560	42.0382

less time than $\hat{d}_2$ for Boston Housing, Servo, Auto Mpg, and Delta Ailerons, $\hat{d}_2$ provides faster results for the rest data sets. When compared to other algorithms, L-ELM algorithms for $\hat{d}_1$ and $\hat{d}_2$ are much faster than RR-ELM and AUR-ELM for Boston Housing, Servo, Abalone, and Auto Mpg and L-ELM algorithm for $\hat{d}_2$ is faster than RR-ELM and AUR-ELM for Delta Ailerons, Bank Domains, and California Housing. RR-ELM and AUR-ELM algorithms providing competitive and similar results for Delta Elevators, Bank Domains, California Housing, and House Census data sets when compared to L-ELM. OP-ELM is the slowest algorithm for all data sets.

- When L-ELM is compared to the training performance of basic ELM, L-ELM for at least one d parameter is better for the most of data sets except Abalone, Delta Ailerons and Delta Elevators. When compared to other algorithms, L-ELM for at least one d parameter gives smaller training RMSE than RR-ELM, AUR-ELM and OP-ELM for Boston Housing, Bank Domains and House Census data sets. According to the testing performance, L-ELM for at least one d parameter is better than ELM for all cases providing smaller testing RMSE values on average. This result shows that L-ELM improves the generalization performance of the basic ELM. Besides, L-ELM is found as better than OP-ELM for all cases.

- In Table 5.4, the standard deviations of testing RMSE are given to investigate the stability of L-ELM performance. By observing these values, it is seen that L-ELM provides smaller standard deviations for at least one d parameter. Besides, the selection of the Liu tuning parameter is effective on the performance of the standard deviation. According to the standard deviations, L-ELM with

$\hat{d}_2$ parameter gives better results for Boston Housing, Servo, Auto Mpg and California Housing data sets than $\hat{d}_1$ which is better for all other data sets. When L-ELM compared with RR-ELM, AUR-ELM, and OP-ELM, it is more stable for at least one Liu tuning parameter for all data set except Auto Mpg and Abalone data sets. It should be noted that the standard deviation of L-ELM is slightly larger than RR-ELM for the latter data set.

In order to determine whether there is statistical significance between the results of algorithms, a one-way analysis of variance (ANOVA) test has been conducted on each data set using testing RMSE. Based on ANOVA results, the differences between RMSE scores of fifty trials are found statistically significant (for each data set, $p < 0.05$). These results mean that at least two algorithms are significantly different from each other for each data set. As well as the normality and homogeneity of variances assumptions, the existence of outliers is critical in terms of the validity and significance of the results in ANOVA tests. Therefore, these assumptions have been checked via appropriate tests and approaches. It is found that there are no outliers in data sets according to box-plot and z-scores. The normality assumption of ANOVA has been met by the Kolmogorov-Smirnov test (for each data set, $p > 0.05$); but the assumption related to homogeneity of variances has been violated according to Levene's test (for each data set, $p < 0.05$). Because of violation of variance assumption, the Games-Howell post hoc test has been carried out to determine which algorithms are different from each other. [1] The comparison results based on the Games-Howell post hoc test between the proposed algorithms and the competitors have been given in Table 5.5.

[1] Only the main results of the whole statistical process have been given for the sake of simplicity and keeping the study shorter.

Based on the results in Table 5.5, the following interpretations for L-ELM(d_1) and/or L-ELM(d_2) can be presented:

- L-ELM is statistically superior to ELM for Servo, Abalone, Auto Mpg, Delta Ailerons, Delta Elevators, and House Census.
- L-ELM is statistically superior to RR-ELM for Boston, Abalone, Auto Mpg, Delta Ailerons, Delta Elevators, and Bank Domains.
- L-ELM is statistically superior to AUR-ELM for Servo, Abalone, Auto Mpg, Delta Ailerons, Delta Elevators, and California Housing.
- L-ELM is statistically superior to OP-ELM for Boston, Servo, Abalone, Auto Mpg, Delta Ailerons, Delta Elevators, Bank Domains, California Housing, and House Census.

As a result, L-ELM (d_1) and/or L-ELM (d_2) may overcome to ELM, RR-ELM, AUR-ELM, and OP-ELM as statistically in terms of testing performance for at least one data set. To sum up, according to testing performance in Table 5.4 and Table 5.5, it can be said that the L-ELM algorithm for at least one Liu tuning parameter provides generally better results for both the mean and the standard deviation of testing RMSE than ELM, RR-ELM, AUR-ELM, and OP-ELM. The proposed algorithms have poorer performance than only for RR-ELM but better than the others in terms of standard deviations for Servo, Auto and Abalone data sets. The difference is slight for the latter data set. These results are also consistent with the results in Yıldırım and Özkale (2019) and show that the ELM based on Liu estimator may present faster, more generalizable and more stable results than ELM and its variants.

Table 5.5. The post hoc results on the testing RMSE to show the statistical significance for each data set

Data set	Algorithm	Statistics	ELM	RR-ELM	AUR-ELM	OP-ELM	L-ELM(d_2)
Boston	L-ELM(d_1)	t	-2.022	7.938	-2.180	-10.369	3.791
		p	0.35	$<$ **0.001**	0.269	$<$ **0.001**	$<$ **0.001**
	L-ELM(d_2)	t	-0.260	4.997	0.045	-12.460	–
		p	1.000	$<$ **0.001**	1.000	$<$ **0.001**	–
Servo	L-ELM(d_1)	t	2.901	1.103	2.902	-5.124	0.763
		p	0.059	0.878	0.056	$<$ **0.001**	0.973
	L-ELM(d_2)	t	3.641	0.510	3.949	-7.142	–
		p	**0.011**	0.996	**0.004**	$<$ **0.001**	–
Abalone	L-ELM(d_1)	t	4.900	-3.438	3.202	-2.898	-3.065
		p	$<$ **0.001**	**0.013**	**0.033**	0.086	**0.035**
	L-ELM(d_2)	t	1.537	0.679	1.505	-2.840	–
		p	0.642	0.984	0.663	0.079	–
Auto Mpg	L-ELM(d_1)	t	9.680	-6.978	9.990	-8.231	5.602
		p	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**
	L-ELM(d_2)	t	15.949	-13.125	15.018	-11.904	–
		p	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**	–
Delta Ailerons	L-ELM(d_1)	t	1.252	-26.754	22.230	-10.960	-6.662
		p	0.809	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**
	L-ELM(d_2)	t	17.464	-15.567	15.304	-9.587	–
		p	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**	$<$ **0.001**	–
Delta Elevators	L-ELM(d_1)	t	3.558	-3.779	4.008	-6.532	-7.311
		p	**0.012**	**0.006**	**0.003**	$<$ **0.001**	$<$ **0.001**
	L-ELM(d_2)	t	-1.723	1.626	-2.646	-4.808	–
		p	0.523	0.585	0.100	$<$ **0.001**	–
Bank Domains	L-ELM(d_1)	t	0.913	-4.246	1.744	-62.928	-0.633
		p	0.941	**0.001**	0.509	$<$ **0.001**	0.988
	L-ELM(d_2)	t	0.376	-3.351	1.081	-62.352	–
		p	0.999	**0.016**	0.887	$<$ **0.001**	–
California Housing	L-ELM(d_1)	t	2.371	-2.283	3.570	-4.568	-1.686
		p	0.657	0.218	**0.010**	**0.001**	0.544
	L-ELM(d_2)	t	1.062	-1.020	2.542	-4.504	–
		p	0.894	0.909	0.134	**0.001**	–
House Census	L-ELM(d_1)	t	-1.296	0.947	0.946	-19.087	-3.785
		p	0.785	0.931	0.931	$<$ **0.001**	**0.004**
	L-ELM(d_2)	t	-4.137	3.376	-1.150	-16.101	–
		p	**0.001**	**0.016**	0.858	$<$ **0.001**	–

The testing errors of ELM and L-ELM algorithms for Boston Housing, Auto Mpg, House Census, and Bank Domains data sets are given by Figures 5.7-5.10. It is seen from Figures 5.7-5.10 that the testing error values of L-ELM results are slightly more compact around zero than ELM. This shows that L-ELM provides more stable results than ELM. The results in Figures 5.7-5.10 for OP-ELM are obtained by using the mean node number of all trials. Similarly, the mean values of ridge parameters for all trials are used to get RR-ELM and AUR-ELM results.

In Figure 5.11, the RMSE performances of the data sets for d values in the interval $[0,1]$ are given. As seen from Figure 5.11 that the RMSE values decrease as d values increase and there is one d value minimizing RMSE. When L-ELM is compared with ELM by obtaining $d = 1$ in L-ELM, it has a smaller RMSE value for the optimum d parameter. This means that L-ELM clearly improves the ELM for at least one d value and takes the minimum RMSE for one global d value.

It should be noted that ELM provides successful results like low training error, good generalization performance and low computational cost in many real-world applications. Therefore, it is hard to compete with ELM in practice and improving ELM in a certain amount is noteworthy. In Figure 5.11, it is clearly seen that one can get better performance than ELM as long as the Liu tuning parameter d is determined carefully.

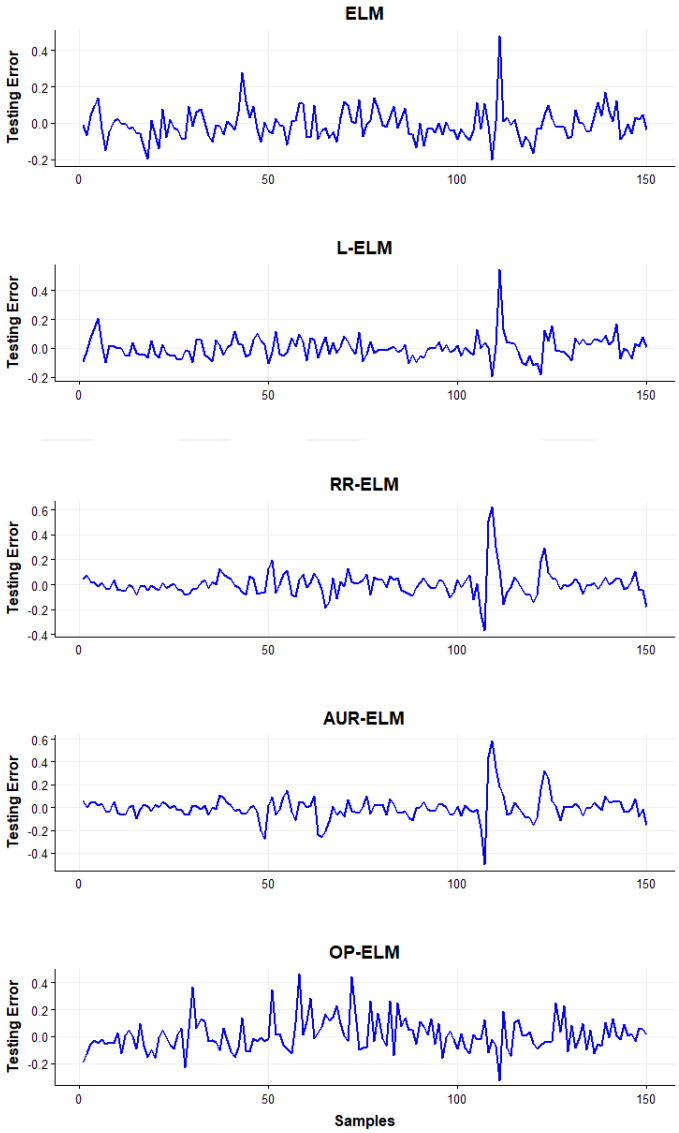

Figure 5.7. The testing errors of ELM, RR-ELM, AUR-ELM, OP-ELM and L-ELM for $\hat{d}_2$ algorithms for Boston Housing data set

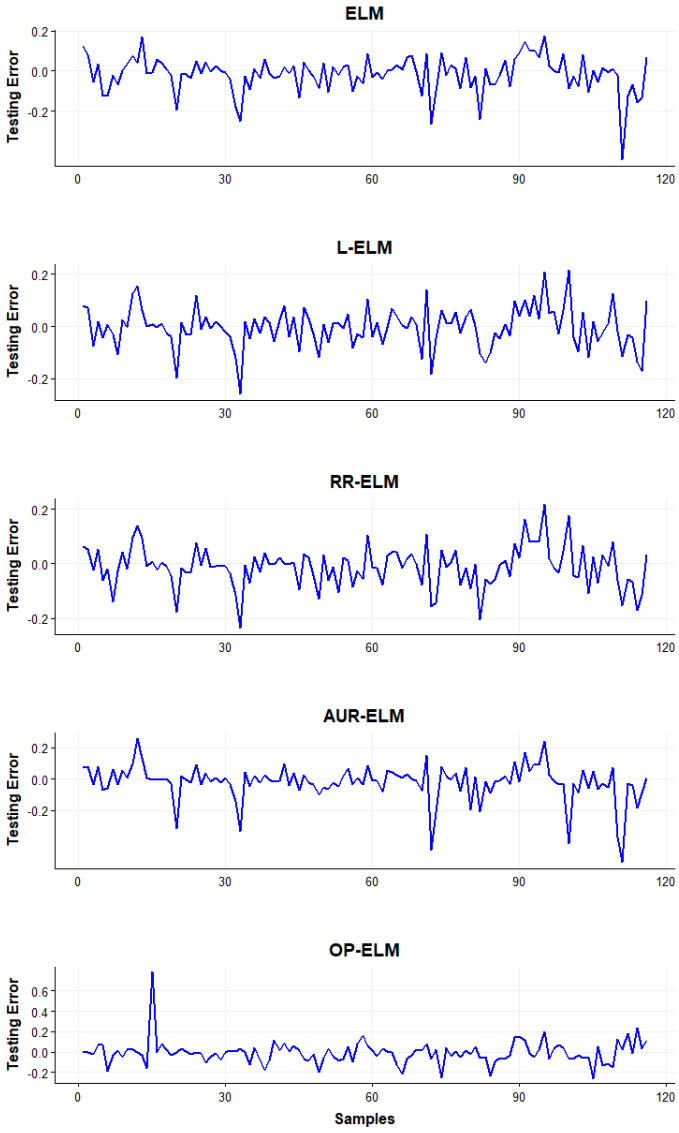

Figure 5.8. The testing errors of ELM, RR-ELM, AUR-ELM, OP-ELM and L-ELM for $\hat{d}_2$ algorithms for Auto Mpg data set

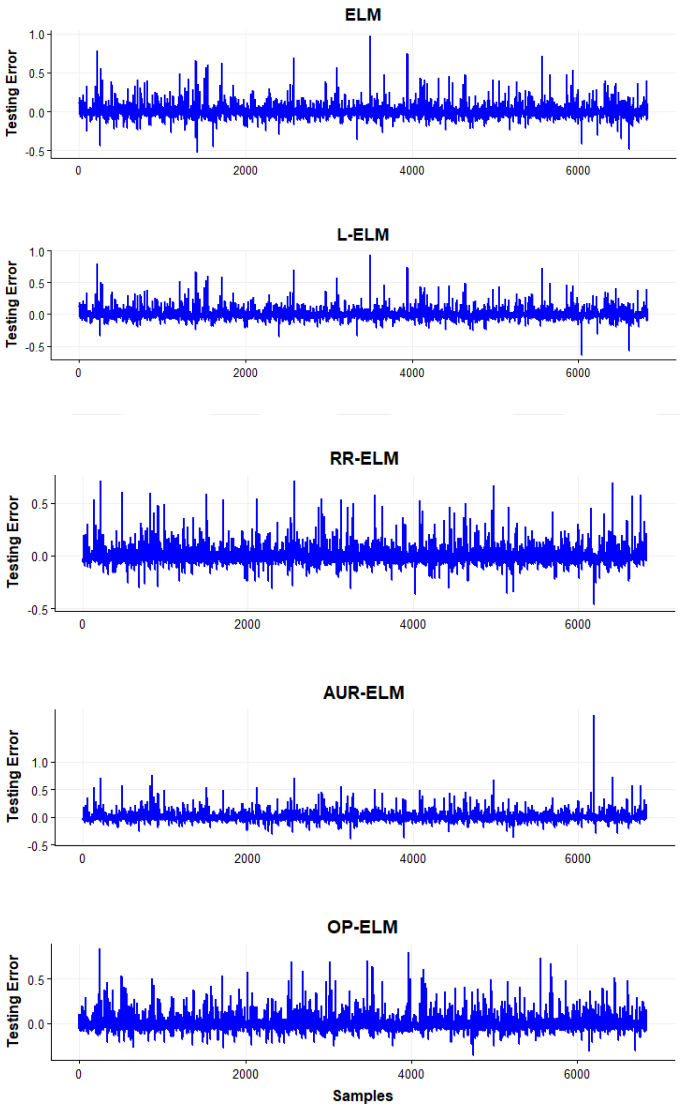

Figure 5.9. The testing errors of ELM, RR-ELM, AUR-ELM, OP-ELM and L-ELM for $\hat{d}_1$ algorithms for House Census data set

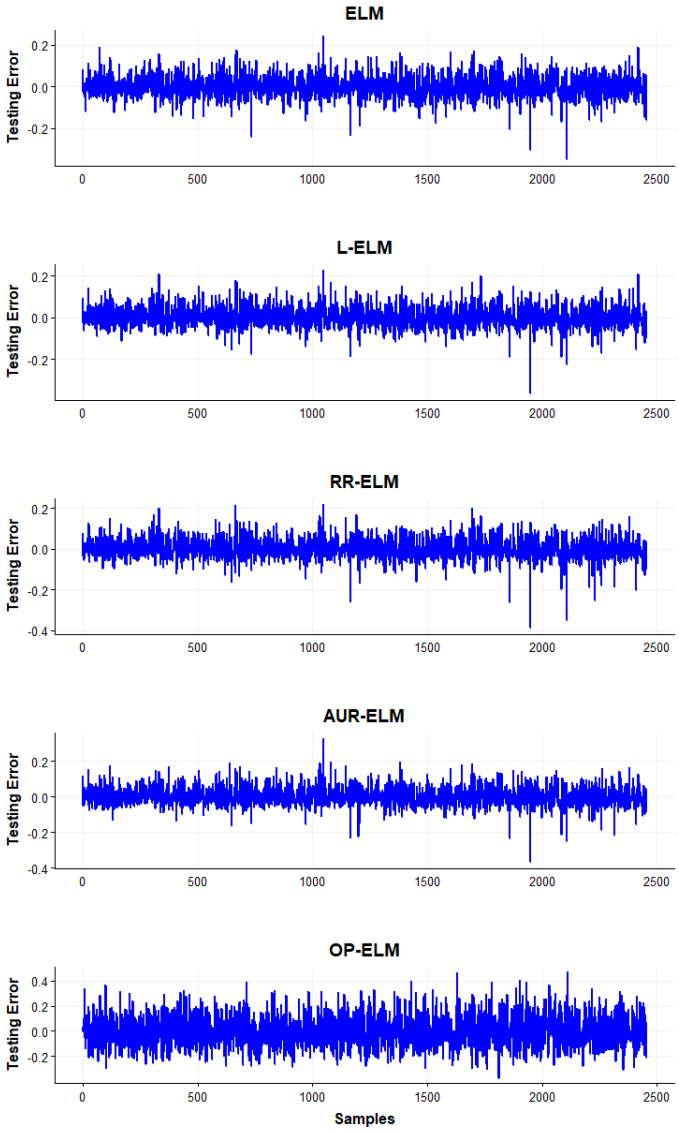

Figure 5.10. The testing errors of ELM, RR-ELM, AUR-ELM, OP-ELM and L-ELM for $\hat{d}_1$ algorithms for Bank Domains data set

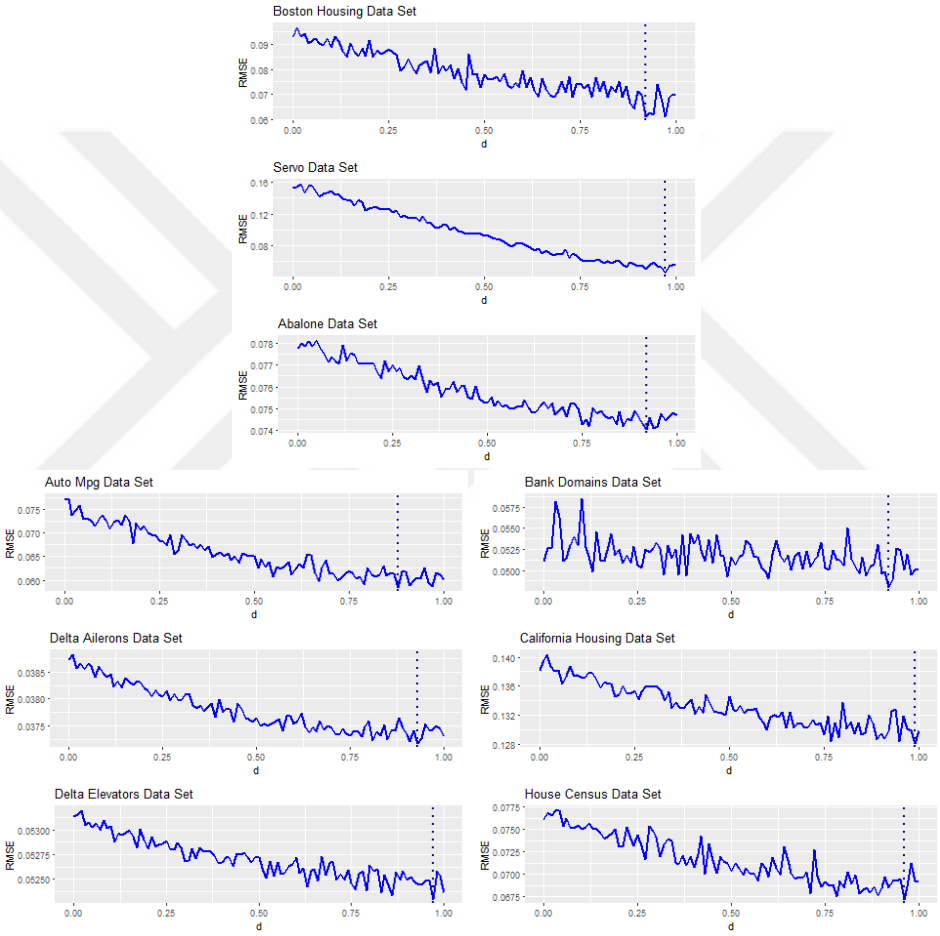

Figure 5.11. The L-ELM performance change for each data set based on sequential tuning parameter in the range of $[0, 1]$

5.3 Performance Evaluation of RK-ELM

In this section, the numerical performance of the proposed PCR-ELM and RK-ELM algorithms are investigated and compared with the classical ELM algorithm for regression problems. Three real-world data sets have been used to determine the efficiency of the aforementioned algorithms. Auto Mpg and Boston Housing data sets are collected from UCI (Dua and Graff, 2020) repository. Body Fat data set which was donated for publicl usage was firstly used in the study of Penrose et al. (1985). The properties of all data sets have been summarized in Table 5.6. Each data set has been split to training and testing sets with the ratio of %75 and %25, respectively. Thirty trials have been carried out to eliminate the effect of initial assignments of weighs and bias values in the input layer. The RMSE is used as the performance criterion. The mean value of RMSE for all trials has been reported. In order to examine the stability of training and testing performance, the standard deviations of RMSE have been calculated. Sigmoidal activation function has been used in experiments for all data sets.

Table 5.6. The properties of benchmark data sets

Data Sets	Attributes	Sample Size	Training Set	Testing Set
Auto Mpg	9	392	295	97
Boston Housing	14	506	381	125
Body Fat	15	252	192	60

All the experiments have been conducted with the R software environment on an i7 2.3GHZ CPU and 12GB RAM computer. The source code of the proposed algorithm has been written from scratch by considering the base code provided by Huang et al. (2006).

The number of hidden layer neurons has been determined via EM-ELM proposed by Feng et al. (2009). The proposed algorithms have been applied to the hidden layer matrix obtained by EM-ELM. The threshold value for CCR is taken as %95 and the number of PCs is found by using this value. The numbers of PCs are variable for each trial but the mean of them for all trials has been found as 6, 8 and 10 for Auto Mpg, Boston Housing, and Body Fat, respectively. The tuning parameters $\hat{k}_1$, $\hat{k}_2$ and $\hat{k}_3$ which are given in Section (4.3.3) have been considered for all data sets and algorithms. Besides three types of $\hat{k}$ values, random values of tuning parameter between 0 and 1 have been used for comparison purposes. The final solution using tuned parameters has been obtained. The general steps of process flow is explained in Figure 5.12.

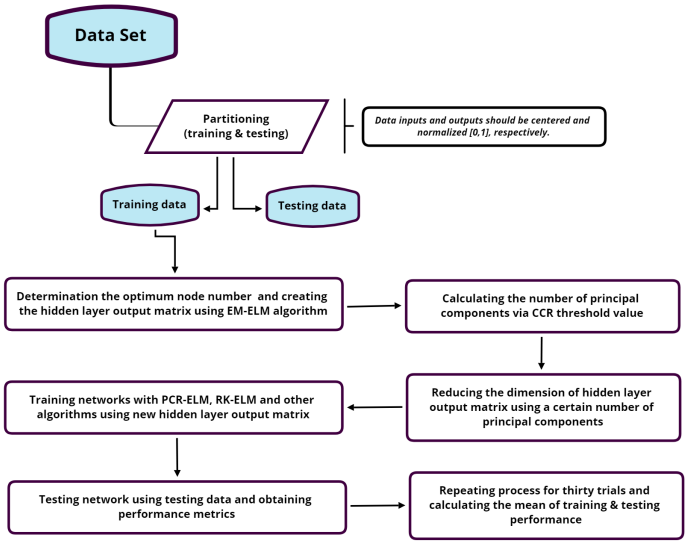

Figure 5.12. The general architecture of the PCR-ELM and RK-ELM algorithms

The estimation results of the experimental study have been given in Table 5.7. The visual representation of improvement for each algorithm over the RK-ELM on testing RMSE and standard deviation of it are given by Figures 5.13 and 5.14. The changes in comparison are based on the reduction rate representing the relative efficiency and calculated via

$$RR = \frac{RMSE_{any\ alg.} - RMSE_{RK-ELM}}{RMSE_{any\ alg.}} \times 100. \tag{5.1}$$

The following conclusions are obtained from the results:

- The proposed algorithms improve testing performance. RK-ELM outperforms ELM, PCR-ELM, PCA-ELM, and RR-ELM based on random tuning parameter selection. However, RR-ELM can produce better results than RK-ELM for the proposed k selection methods. The improvements in the testing RMSE are quite variable and this means that the performance of the RK-ELM algorithm is data-dependent.

- The improvement on the stability (i.e. variability) is measured by standard deviation of testing RMSE. When examined the last two columns of Table 5.7, it can be said that RK-ELM algorithm provides more stable results than ELM, PCR-ELM, PCA-ELM and RR-ELM algorithms for Auto Mpg, and Body Fat data sets. RR-ELM is more generalizable than RK-ELM only for Boston Housing data sets.

- According to the training results in Table 5.7, the ELM outperforms RK-ELM in terms of training performance. The ELM provides significantly better results than the others. However, a similar interpretation is not valid for the

Table 5.7. Performance comparison between RK-ELM and other algorithms

Data set	Algorithm	k type	k		#Hidden Nodes	RMSE (Train)		RMSE (Test)	
			Mean	S.Dev.	Mean	Mean	S.Dev.	Mean	S.Dev.
Auto Mpg	ELM	–	–	–	258	0.35168	0.03747	0.62266	0.43646
	PCR-ELM	–	–	–	258	0.38405	0.01802	0.40493	0.01686
	RK-ELM	k_1	16.93	17.07	258	0.38424	0.01804	0.40352	0.01675
		k_2	45.36	44.17	258	0.38470	0.01783	0.40248	0.01718
		k_3	245.62	304.34	258	0.38666	0.01701	0.40148	0.01746
		Random	0.505	0.29	258	0.38139	0.01508	0.33127	0.00930
	RR-ELM	k_1	16.93	17.07	258	0.37061	0.02466	0.39577	0.02080
		k_2	45.36	44.17	258	0.37670	0.02153	0.39713	0.01961
		k_3	245.62	304.34	258	0.38403	0.01818	0.39956	0.01826
		Random	0.505	0.29	258	0.28285	0.07963	0.38714	0.05127
	PCA-ELM	–	–	–	258	0.38405	0.01802	0.40483	0.01710
Boston Housing	ELM	–	–	–	114	0.15414	0.07927	0.84449	0.56631
	PCR-ELM	–	–	–	114	0.26026	0.03297	0.22180	0.03868
	RK-ELM	k_1	1.63	0.76	114	0.26029	0.03298	0.22096	0.03867
		k_2	6.29	3.34	114	0.26062	0.03296	0.21916	0.03891
		k_3	20.23	11.66	114	0.26315	0.03274	0.21709	0.03917
		Random	0.505	0.29	114	0.26924	0.04427	0.24887	0.03828
	RR-ELM	k_1	1.63	0.76	114	0.18097	0.05669	0.19845	0.03194
		k_2	6.29	3.34	114	0.19758	0.04802	0.18508	0.03683
		k_3	20.23	11.66	114	0.21928	0.04197	0.18528	0.03836
		Random	0.505	0.29	114	0.19959	0.07855	0.25265	0.03659
	PCA-ELM	–	–	–	114	0.26026	0.03297	0.42441	0.06937
Body Fat	ELM	–	–	–	76	0.31971	0.05865	0.59884	0.08675
	PCR-ELM	–	–	–	76	0.38205	0.02210	0.58419	0.01693
	RK-ELM	k_1	1.42	0.81	76	0.38230	0.02216	0.58316	0.01660
		k_2	6.82	3.03	76	0.38517	0.02196	0.58188	0.01584
		k_3	33.18	17.26	76	0.39841	0.01796	0.58559	0.01309
		Random	0.505	0.29	76	0.34115	0.05364	0.42438	0.05914
	RR-ELM	k_1	1.42	0.81	76	0.33152	0.04889	0.55249	0.03619
		k_2	6.82	3.03	76	0.35531	0.03655	0.55973	0.02834
		k_3	33.18	17.26	76	0.38817	0.02271	0.57710	0.01858
		Random	0.505	0.29	76	0.22779	0.11577	0.46382	0.08877
	PCA-ELM	–	–	–	76	0.38505	0.02210	0.63105	0.02973

stability of training performance. RK-ELM algorithm presents more stable results than the ELM algorithm. Actually, RK-ELM is significantly better than all other algorithms in terms of stability of training performance.

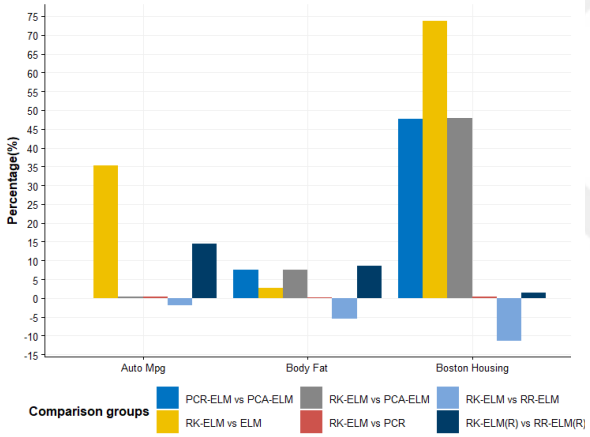

Figure 5.13. The comparison based on testing RMSE

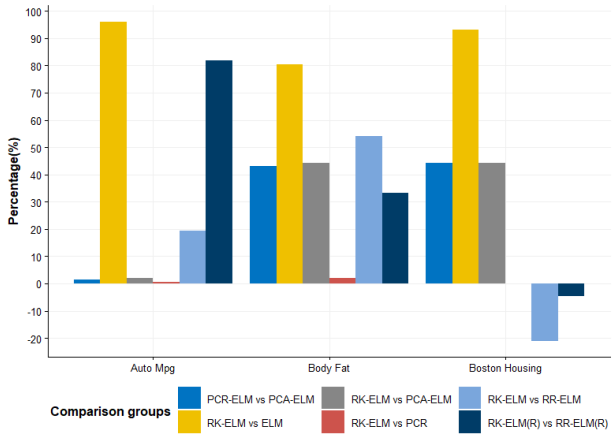

Figure 5.14. The comparison based on standard deviations of testing RMSE

In order to investigate the effects of proposed algorithms on the generalization (i.e. testing) and stability performance, the residuals (i.e. testing error) for Auto Mpg, Boston Housing, and Body Fat data sets have been shown in Figures 5.15, 5.16 and 5.17, respectively. The mean node value has been considered in calculations. According to Figures 5.15, 5.16 and 5.17, it is shown that the testing errors (i.e. residuals) obtained by the RK-ELM algorithm are more stable around zero than ELM, PCR-ELM, RR-ELM and PCA-ELM algorithms for all data sets. This means that the proposed algorithm estimates the data with lower and less variable errors. From the machine learning view, the generalization performance of the proposed algorithm is better than the rest of all algorithms.

According to the statistical theory of biased estimators, RK-ELM may provide more generalizable and stable results than RR-ELM if it can be tuned in a careful manner. The selection of the k tuning parameter is critical to the performance of RK-ELM. A range of k values from 0 to 1 is used to validate that RK-ELM may outperform RR-ELM in a certain range. The results are given in Figure 5.18. It is shown that there is at least one point where RK-ELM gives smaller test MSE then RR-ELM for all data sets. This interpretation is valid when the performances based on random k values in Table 5.7 are examined.

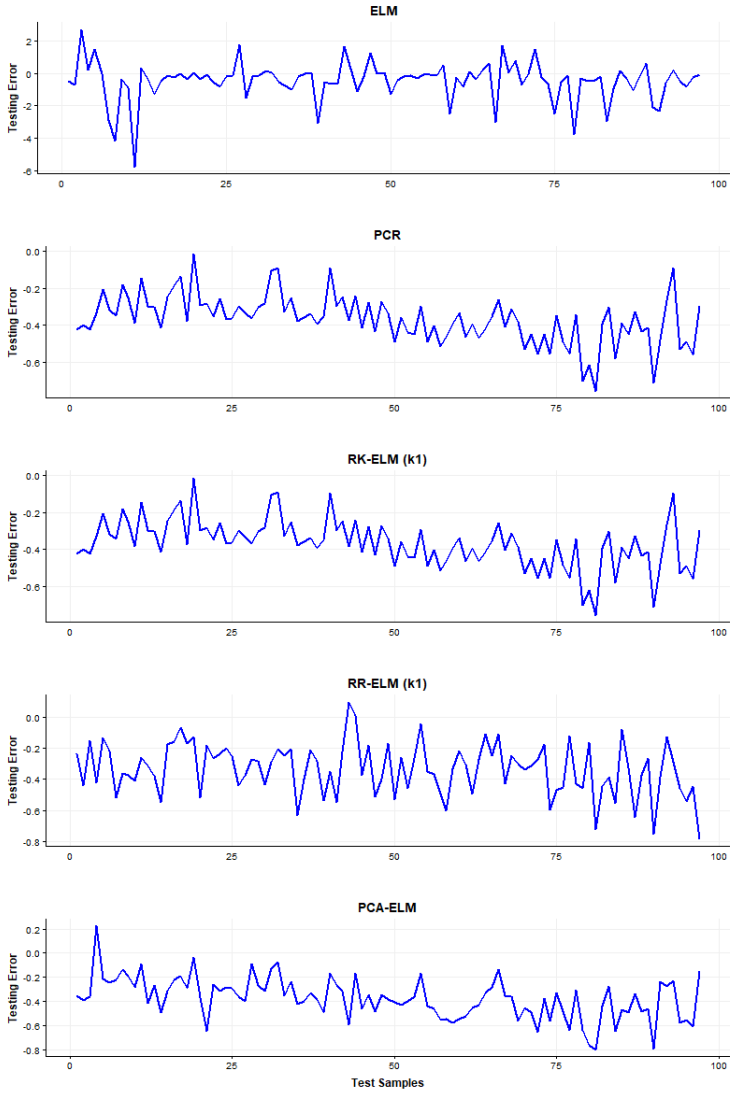

Figure 5.15. The residuals based on testing RMSE for Auto Mpg ($m = 258$)

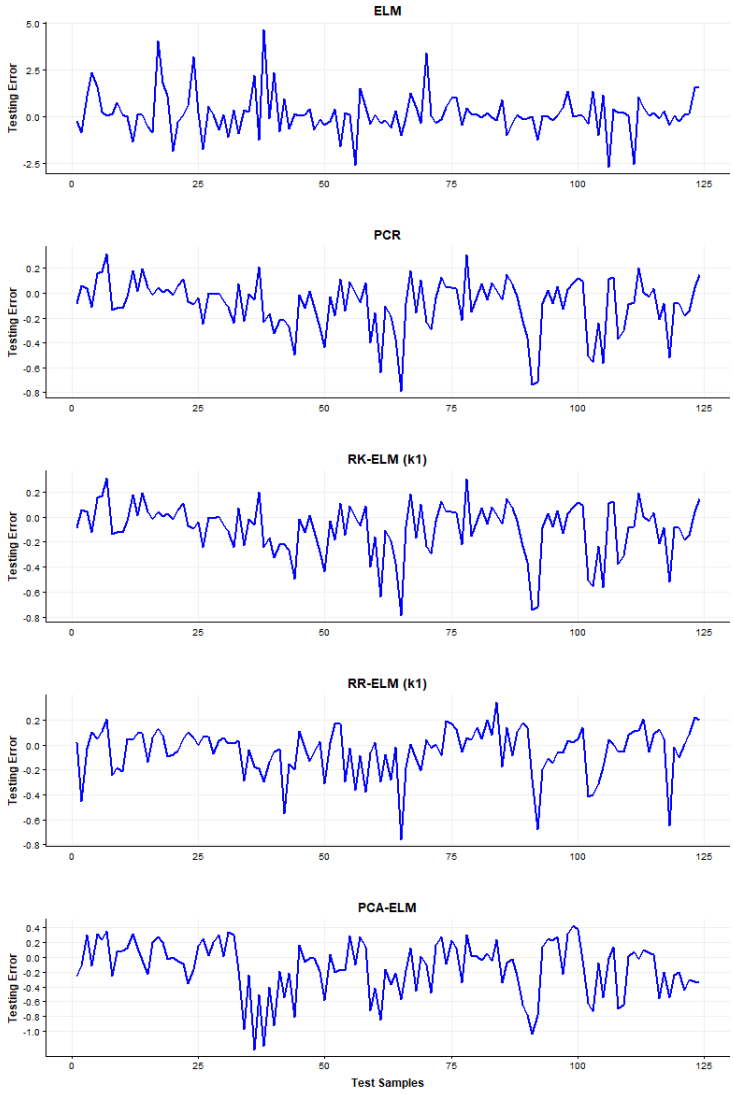

Figure 5.16. The residuals based on testing RMSE of Boston ($m = 114$)

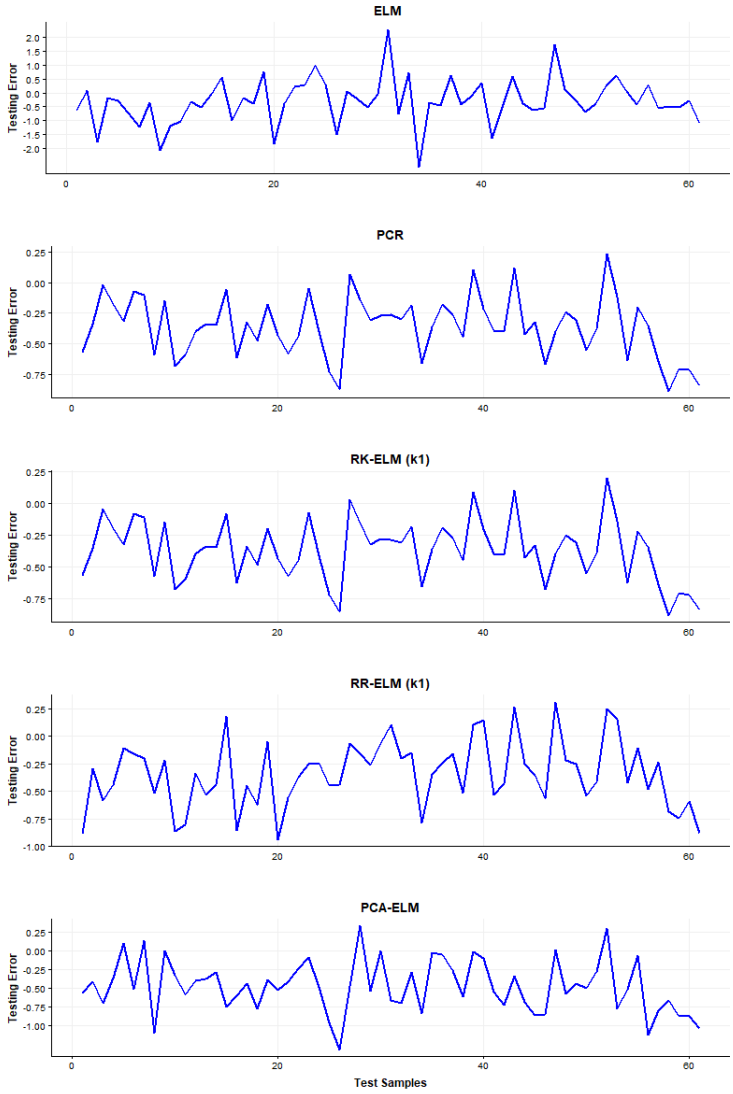

Figure 5.17. The residuals based on testing RMSE of Body Fat ($m = 76$)

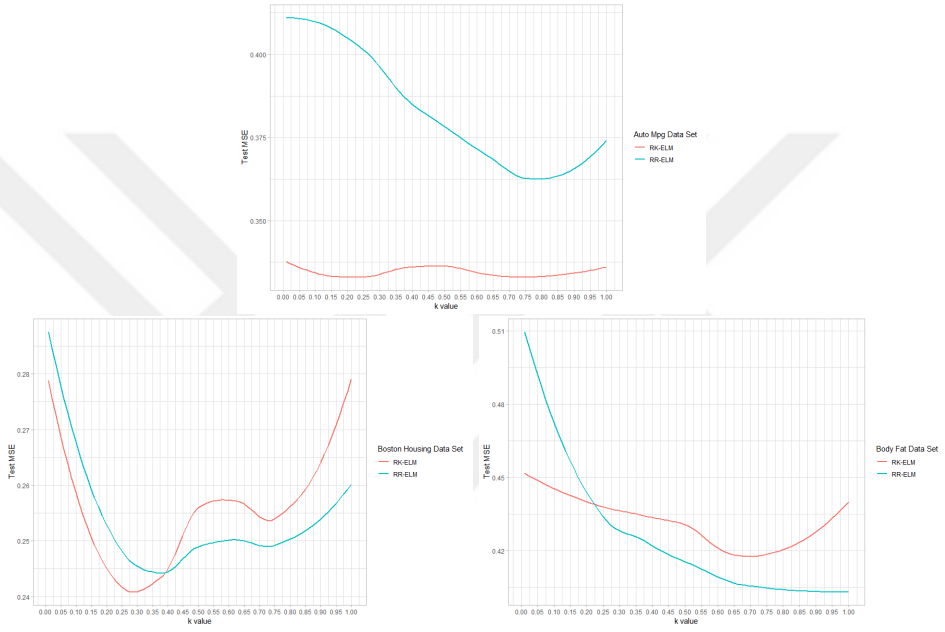

Figure 5.18. The comparison of RK-ELM and RR-ELM for each data set for arbitrary k values from 0 to 1

5.4 Performance Evaluation of LL-ELM

In this section, a performance comparison has been carried out on several benchmark data sets to investigate the effectiveness of the algorithms. All data sets have been obtained from UCI repository (Dua and Graff, 2020). The properties of data set are given in Table 5.8. All data sets have been standardized to have zero mean and unit variance. Each data set is split into training and testing data sets with the ratios %70 and %30, respectively. Twenty trials have been conducted to eliminate the randomness effect of initial parameters assignments. The initial number of the hidden layer nodes is fixed as 100 and sigmodial activation function is used for all data sets and experiments. The experiments have been carried out via R software. In order to train LL-ELM, ELM variants based on Lasso and Elastic net algorithms, LARS-EN algorithm with piecewise linear regularization path proposed by Rosset and Zhu (2007) has been used. All coding processes of each algorithm have been carried out from scratch in the R platform.

Table 5.8. The benchmark data sets for Liu-Lasso based algorithm

Data Sets	Attributes	Sample Size	Training Set	Testing Set
Body fat	14	252	179	73
Energy	8	768	540	228
Fish	6	908	637	271
Bank domains	8	8192	5736	2456
Abalone	8	4177	2925	1252

In Lasso-ELM, instead of tuning λ, it is suggested to use the fraction of L_1 norm of coefficients (s) which is defined as $\left\|\hat{\boldsymbol{\beta}}_{ELM}\right\| / \max\left(\hat{\boldsymbol{\beta}}_{ELM}\right)$. Here, $\max\left(\hat{\boldsymbol{\beta}}_{ELM}\right)$ actually corresponds the ELM solution which is the L_1 norm of low-

bias model solution. From the point of optimization perspective, there is a λ value corresponding to each s value and the solutions obtained via any form of this optimization problem are exactly the same each other. In other words, the $\hat{\boldsymbol{\beta}}$ solution corresponding any λ value in Lagrangian form solves the problem which has the bound of $s = \left\| \hat{\boldsymbol{\beta}}_{\lambda} \right\|$. The advantage of s over λ is to have values within $[0,1]$. Similar the process in LL-ELM, BIC criterion is used to determined optimal s value by using each fixed $\hat{d}_1$ and $\hat{d}_2$ parameters as initial parameters. On the other hand, the ridge tuning parameter k is selected from the sequence of $\left(10^{-15}, 10^{-14}, ..., 10^{1}\right)$. In RR-ELM, the k value minimizing the training error is used to obtain training and testing results. In order to get the ELM results based on elastic net, for each fixed k value, the optimal s value minimizing BIC criterion is calculated and the (k,s) values giving the global minimum among all possible combinations are used for the final model's performance.

In addition to the performance results of each algorithms, the optimal parameters are presented in Table 5.9. In Table 5.9, the third and fourth columns correspond the average d values and its standard deviations which are calculated based on all trials. The k and s columns refer to the best parameters corresponding to the optimal (k,s) or (d,s) combination giving the overall minimum value of BIC criterion. In the last column, the mean node number throughout all trials is reported.

Table 5.9. The values of tuning parameters

Data set	Algorithm	d	SD	k	s	$Node$
	ELM	0	0	0	0	100
	Ridge-ELM	0	0	10^{-12}	0	100
	Liu-ELM (d_1)	0,63	0,07	0	0	100
Body Fat	Liu-ELM (d_2)	0,89	0,02	0	0	100
	Lasso-ELM	0	0	0	0,71	81,25
	ENet-ELM	0	0	10^{-2}	0,91	93,30
	LL-ELM (d_1)	0,63	0,07	0	0,98	97,75
	LL-ELM (d_2)	0,89	0,02	0	0,98	98,70
	ELM	0	0	0	0	100
	Ridge-ELM	0	0	10^{-12}	0	100
	Liu-ELM (d_1)	0,84	0,06	0	0	100
Energy	Liu-ELM (d_2)	0,99	0,00	0	0	100
	Lasso-ELM	0	0	0	0,08	43,45
	ENet-ELM	0	0	10^{-1}	0,80	71,10
	LL-ELM (d_1)	0,84	0,06	0	0,998	99,60
	LL-ELM (d_2)	0,99	0,00	0	0,998	99,45
	ELM	0	0	0	0	100
	Ridge-ELM	0	0	10^{-13}	0	100
	Liu-ELM (d_1)	0,27	0,26	0	0	100
Fish	Liu-ELM (d_2)	0,93	0,02	0	0	100
	Lasso-ELM	0	0	0	0,03	16,30
	ENet-ELM	0	0	0,1	0,13	16,60
	LL-ELM (d_1)	0,27	0,26	0	0,61	62,60
	LL-ELM (d_2)	0,93	0,02	0	0,99	98,45

Table 5.9. The values of tuning parameters (Cont'ed)

Data set	Algorithm	d	SD	k	s	$Node$
	ELM	0	0	0	0	100
	Ridge-ELM	0	0	10^{-13}	0	100
	Liu-ELM (d_1)	0,10	0,29	0	0	100
Bank	Liu-ELM (d_2)	0,26	0,16	0	0	100
Domains	Lasso-ELM	0	0	0	0,94	94,30
	ENet-ELM	0	0	10^{-3}	0,99	99,05
	LL-ELM (d_1)	0,10	0,29	0	0,94	93,65
	LL-ELM (d_2)	0,26	0,16	0	0,90	89,80
	ELM	0	0	0	0	100
	Ridge-ELM	0	0	10^{-12}	0	100
	Liu-ELM (d_1)	0,15	0,19	0	0	100
Abalone	Liu-ELM (d_2)	0,83	0,04	0	0	100
	Lasso-ELM	0	0	0	0,11	37,20
	ENet-ELM	0	0	10^{-1}	0,38	44,00
	LL-ELM (d_1)	0,15	0,19	0	0,34	35,80
	LL-ELM (d_2)	0,83	0,04	0	0,98	98,10

Table 5.10. Comparison of all algorithms in terms of training and testing RMSE

Data set	Algorithm	*Training*	*SD*	*Testing*	*SD*
	ELM	0,05924	0,00810	0,26786	0,02831
	Ridge-ELM	0,05920	0,00810	0,26785	0,02828
	Liu-ELM (d_1)	0,06821	0,00962	0,24545	0,02318
Body Fat	Liu-ELM (d_2)	0,06018	0,00828	0,26040	0,02670
	Lasso-ELM	0,07899	0,01808	0,25684	0,02666
	ENet-ELM	0,06059	0,01480	0,25794	0,02799
	LL-ELM (d_1)	0,06564	0,00944	0,24460	0,02270
	LL-ELM (d_2)	0,05092	0,00742	0,26018	0,02699
	ELM	0,16976	0,00991	0,21523	0,01617
	Ridge-ELM	0,16976	0,00988	0,21522	0,01613
	Liu-ELM (d_1)	0,17248	0,01095	0,20932	0,01498
Energy	Liu-ELM (d_2)	0,16977	0,00992	0,21466	0,01598
	Lasso-ELM	0,24887	0,02330	0,27770	0,01627
	ENet-ELM	0,21968	0,00861	0,26511	0,00957
	LL-ELM (d_1)	0,15944	0,01042	0,20917	0,01517
	LL-ELM (d_2)	0,16673	0,03764	0,22294	0,04242
	ELM	0,49160	0,00698	0,77594	0,06101
	Ridge-ELM	0,49158	0,00690	0,77591	0,06100
	Liu-ELM (d_1)	0,52827	0,02571	0,66426	0,02918
Fish	Liu-ELM (d_2)	0,49194	0,00707	0,75992	0,05782
	Lasso-ELM	0,55793	0,01405	0,67020	0,00983
	ENet-ELM	0,55376	0,01901	0,67272	0,01126
	LL-ELM (d_1)	0,52782	0,05457	0,67233	0,02690
	LL-ELM (d_2)	0,45960	0,00785	0,75835	0,05758

Table 5.10. Comparison of all algorithms in terms of training and testing RMSE (Cont'ed)

Data set	Algorithm	*Training*	*SD*	*Testing*	*SD*
	ELM	0,21702	0,00401	0,21852	0,00511
	Ridge-ELM	0,21702	0,00400	0,21849	0,00510
	Liu-ELM (d_1)	0,21738	0,00413	0,21843	0,00493
Bank	Liu-ELM (d_2)	0,21725	0,00406	0,21838	0,00495
Domains	Lasso-ELM	0,21624	0,00460	0,21948	0,00551
	ENet-ELM	0,21576	0,00421	0,21895	0,00524
	LL-ELM (d_1)	0,21835	0,00509	0,21907	0,00492
	LL-ELM (d_2)	0,21829	0,00475	0,21998	0,00526
	ELM	0,63230	0,00221	0,64882	0,00455
	Ridge-ELM	0,63230	0,00220	0,64880	0,00454
	Liu-ELM (d_1)	0,64343	0,00561	0,63915	0,00243
Abalone	Liu-ELM (d_2)	0,63277	0,00228	0,64449	0,00380
	Lasso-ELM	0,65770	0,01374	0,65823	0,01016
	ENet-ELM	0,65028	0,01082	0,65149	0,00845
	LL-ELM (d_1)	0,66915	0,02135	0,66020	0,01307
	LL-ELM (d_2)	0,62302	0,00281	0,64465	0,00402

Table 5.10 shows the training and testing results of all algorithms based on data sets given in Table 5.8. The training and testing performance with their standard deviations are calculated by taking the averages of twenty trials. Based on the results in Table 5.9, the following interpretations can be said:

- LL-ELM for at least one Liu tuning parameter can overperform to all algorithms in terms of training performance except auto mpg and bank domains data sets.

- According to the testing RMSE values, the proposed algorithm with $\hat{d}_1$ parameter is more generalizable than other algorithms for body fat and energy data sets. Similarly, LL-ELM is seen as stable in terms of standard deviation of testing performance. It provides best results for body fat and bank domains data sets. Liu-ELM is better than LL-ELM in terms of testing performance for auto mpg, fish, bank domains and abalone data sets.

- In order to present an insight into the regularization level of each algorithm, the norm values of coefficients obtained via Liu-ELM and LL-ELM are calculated for these data sets and are given in Table 5.11. Although Liu-ELM has lower testing RMSE value, the mean norm value for Liu-ELM is higher than LL-ELM for all data sets. This means that the proposed algorithm shrinks more severe than Liu-ELM. Additionally, Liu-ELM and LL-ELM are compared based on one hundred trials with the random assignments of Liu tuning parameters within range $[0, 1]$ and the results are given in Fig. 5.19. In Figure 5.19, it is seen that there is at least one Liu tuning parameter where LL-ELM overperforms Liu-ELM for all data sets except Bank Domains.

- When the sparsity results are examined, Lasso-ELM and ENet-ELM generally give more parsimonious models than LL-ELM. For bank domains and abalone data, LL-ELM based on $\hat{d}_2$ provides more sparse models.

Table 5.11. Norm comparison of Liu-ELM and LL-ELM algorithms

Data set	Algorithm	*Norm*	*SD*
Body Fat	Liu-ELM (d_1)	2,9124	0,3426
	Liu-ELM (d_2)	3,6674	0,3904
	LL-ELM (d_1)	2,8741	0,3457
	LL-ELM (d_2)	3,6368	0,4137
Energy	Liu-ELM (d_1)	41,4017	11,8085
	Liu-ELM (d_2)	48,1587	10,4254
	LL-ELM (d_1)	41,3467	11,8364
	LL-ELM (d_2)	48,1413	10,4214
Fish	Liu-ELM (d_1)	17,3667	13,5730
	Liu-ELM (d_2)	44,6700	9,9829
	LL-ELM (d_1)	15,3633	14,9289
	LL-ELM (d_2)	44,3644	10,1091
Auto Mpg	Liu-ELM (d_1)	23,0984	12,9695
	Liu-ELM (d_2)	99,3208	10,0434
	LL-ELM (d_1)	16,6475	16,5184
	LL-ELM (d_2)	97,7532	9,8360
Bank Domains	Liu-ELM (d_1)	2,6077	0,2595
	Liu-ELM (d_2)	2,6741	0,2455
	LL-ELM (d_1)	2,5216	0,3471
	LL-ELM (d_2)	2,5139	0,4060
Abalone	Liu-ELM (d_1)	10,1211	6,0219
	Liu-ELM (d_2)	30,6996	5,6065
	LL-ELM (d_1)	6,3967	6,8261
	LL-ELM (d_2)	30,4090	5,8022

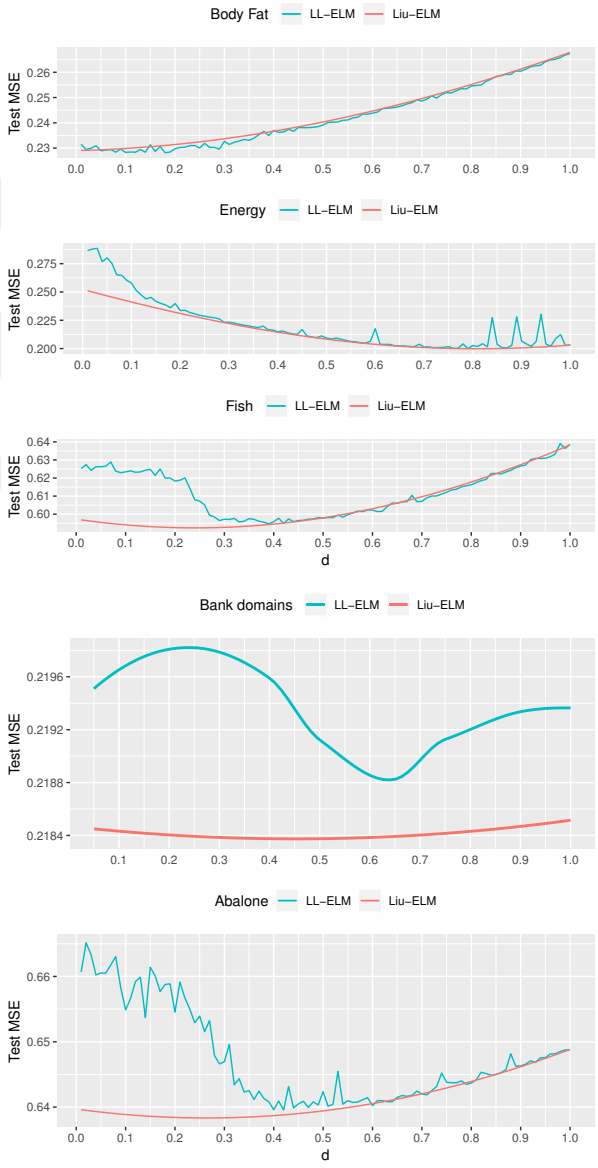

Figure 5.19. The performance comparison of Liu-ELM and LL-ELM based on random tuning parameter

6. CONCLUSIONS AND DISCUSSIONS

6.1 Conclusions

In the information area, the huge amount of data increasing exponentially every day should be analyzed effectively to uncover valuable knowledge inside it. The most prominent way to realize this goal is to benefit from machine learning which provides an automated processes to give some insight for accurate decision making etc. purposes. The neural networks, particularly feedforward neural networks, which is also a sub-field of machine learning serve these purposes in terms of speed, wide applicability and presenting solutions for complex problems. Extreme learning machine which is a type of feedforward neural network has been extensively used and got attention by researchers from a variety of disciplines. Although ELM has some superior properties including extremely fast learning speed, least human intervention for tuning parameters and easy integrability to different tasks and platforms, it has some drawbacks like instability, poor generalizability, and unconvincing sparsity properties.

In this thesis study, some new and novel learning algorithms have been proposed in order to deal with ELM's drawbacks. In the first part of the study, we investigated some alternative ways of selecting optimal tuning parameters for ELM algorithms based on ridge-type estimators. The usage of AIC, BIC and CV criteria are suggested instead of analytical methods. The performance comparison based on different selection criterion and algorithms have been conducted on real-world data sets. The results show that the selection method is effective in both training and testing performances of ELM and its variants like ridge-based ones. Experimental results demonstrate that the selection method of the tuning parameter significantly

affects the performance of RR-ELM and AUR-ELM against the ELM. Especially, the effect on the stability of testing results (i.e. generalization performance) is remarkable. Also, the difference between different selection methods for RR-ELM and AUR-ELM is notable. It should be noted that the attentive selection of the tuning parameter could provide more accurate and stable results on the examination of the data.

In the second part of the study, we have proposed a novel ELM algorithm based on Liu estimator. The main motivation for this study is to benefit from the advantages of Liu estimator like easier tuning parameter selection. It is considered to present an alternative estimator to ridge estimator that has no exact solution for parameter selection in common sense. We also suggested two different methods for determining the optimal Liu parameters for Liu-based ELM. A comprehensive comparative study has been carried out by using L-ELM (a.k.a Liu-ELM), RR-ELM, OP-ELM and the basic ELM via several benchmark data sets. The results show that L-ELM for at least one d parameter is better in terms of testing performance than ELM, RR-ELM, AUR-ELM and OP-ELM. Statistical significance tests based on testing performance also support the most data sets. According to the training time results, L-ELM is generally faster than RR-ELM, AUR-ELM, and OP-ELM. Also, the variability of testing performance in L-ELM for at least one d parameter is generally lower than the opponent. Consequently, L-ELM for at least one d parameter provides better results on the generalization and stability performance than the basic ELM with a certain amount of loss on processing speed.

In the third part of the study, an ELM algorithm based on an r-k class estimator which provides a unified framework of ridge and PCR estimators. PCR is

considered as an effective way to deal with multicollinearity like ridge estimator. In the context of ELM, ELM based on r-k class estimator and PCR which is a special form of it are proposed and compared with RR-ELM, the basic ELM by using different selection criteria of ridge tuning parameters. According to the comparison results, the proposed algorithms may provide better and stables results in terms of RMSE criterion and standard deviation of RMSE. RK-ELM generally outperforms all other algorithms in terms of testing performance and stability. Consequently, the proposed algorithms may be satisfactory tools to obtain more generalizable and stable results in regression studies.

In the last part of the study, we proposed a novel regularization and variable selection algorithm to improve the conventional extreme learning machine and its variants. The proposed algorithm combines the benefits of Liu estimator and Lasso regression methods to deal with the drawbacks of ELM like instability, poor generalizability and under or over-fitting problems. The experimental study shows that LL-ELM improves the training and testing performance of ELM and overperforms well-known competitors. It is seen that LL-ELM has a notable shrinkage property compared with other algorithms, particularly Liu-ELM. Although LL-ELM does not carry out a hard variable selection (i.e. node selection) process like Lasso or Elastic Net, the level of shrinkage with an amount of sparsity is better. The norm of estimated coefficients is lower than other algorithms. This means that LL-ELM may guarantee a lower norm estimated which can provide more stable and accurate results in terms of generalization performance. Besides, the proposed algorithm is a powerful tool for both regressional and classification tasks in data-oriented studies.

6.2 Recommendations and Future Works

The recommendations with the scope of this thesis study can be summarized as follows:

- Although the selection method of the tuning parameter provides some useful insights, there exist some limitations on the usage of ridge based ELM algorithms. Firstly, the selection of the tuning parameter may be difficult for some studies. Therefore some alternative estimators based on simpler tuning parameters can be used. Secondly, some dimension reduction methods like principal component analysis can be functional on reducing the impact of multicollinearity in the ridge based ELM and may provide better estimates. Lastly, to determine the number of hidden layer neurons via an iterative error minimized approach may improve the results and create a more compact network in ridge based ELM algorithms.

- An alternative algorithm to EM-ELM and an appropriate selection method of the tuning parameter can be presented for the RK-ELM. Also, the ways for improving the training performance in RK-ELM should be investigated.

- It should be noted that the level of LL-ELM's sparsity property can be improved by considering alternative ways of parameter selection methods.

REFERENCES

Akaike, H., 1973.Information theory and an extension of the maximum likelihood principle.In B. N. Petrov & B. F. Csaki (Eds.),Second International Symposium on Information Theory, 267-281.Academiai Kiado:Budapest.

Akaike, H., 1998. Information theory and an extension of the maximum likelihood principle. In Selected papers of hirotugu akaike (pp. 199-213). Springer, New York, NY.

Allen, D. M., 1974. The Relationship Between Variable Selection and Data Agumentation and a Method for Prediction. Technometrics, 16(1), 125–127.

Ambrose, S.A., Bridges, M.W., DiPietro, M., Lovett, M.C., and Norman, M.K., 2010. How learning works: Seven research-based principles for smart teaching. San Francisco, CA: Jossey-Bass.

Anderson, J. A., 1972. A simple neural network generating an interactive memory. Mathematical biosciences, 14.3:4 197-220.

Barea, R., Boquete, L., Ortega, S., López, E., and Ascariz, J. M. R., 2012. EOG-based eye movements codification for human computer interaction. Expert Systems with Applications, 39(3), 2677-2683.

Baye, M. R., and Parker, D. F., 1984. Combining ridge and principal component regression:a money demand illustration. Communications in Statistics - Theory and Methods, 13(2), 197–205.

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.

Breheny, P., 2020. Lecture notes: High Dimensional Data Analysis. https://myweb.uiowa.edu/pbreheny/7600/ (Accessed on 10 April 2020).

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.

Cao, F., Liu, B., and Park, D. S., 2013. Image classification based on effective extreme learning machine. Neurocomputing, 102, 90-97.

Castaño, A., Navarro, F. F., and Martínez, C. H., 2013. PCA-ELM: A Robust and Pruned Extreme Learning Machine Approach Based on Principal Component Analysis. Neural Processing Letters, 37(3), 377–392.

Chen, F. L., and Ou, T. Y., 2011. Sales forecasting system based on Gray extreme learning machine with Taguchi method in retail industry. Expert Systems with Applications, 38(3), 1336-1345.

Chen, K., Lv, Q., Lu, Y., and Dou, Y., 2017. Robust regularized extreme learning machine for regression using iteratively reweighted least squares. Neurocomputing, 230, 345–358.

Choi, K., Toh, K. A., and Byun, H., 2012. Incremental face recognition for large-scale social network services. Pattern Recognition, 45(8), 2868-2883.

Creech, G., and Jiang, F., 2012. The application of extreme learning machines to the network intrusion detection problem. In AIP Conference Proceedings, 1479(1), 1506-1511.

Deng, C., Huang, G., Xu, J., and Tang, J., 2015. Extreme learning machines: new trends and applications. Science China Information Sciences, 58(2), 1-16.

Deng, W., Zheng, Q., and Chen, L., 2009. Regularized Extreme Learning Machine. 2009 IEEE Symposium on Computational Intelligence and Data Mining, 389–395.

Ding, S., Zhao, H., Zhang, Y., Xu, X., and Nie, R., 2015. Extreme learning ma-

chine: Algorithm, theory and applications. Artificial Intelligence Review, 44(1), 103–115.

Dua, D. and Graff, C., 2020. UCI Machine Learning Repository. [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California,School of Information and Computer Science. (Accessed on 05 March 2020)

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R., 2004. Least angle regression. The Annals of statistics, 32(2), 407-499.

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.

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.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.

Friedman, J., Hastie, T., and Tibshirani, R., 2010. Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1).

Fu, W. J., 1998. Penalized regressions: the bridge versus the lasso. Journal of computational and graphical statistics, 7(3), 397-416.

Graybill, F. A., 2001. Matrices with applications in statistics. Cengage Learning. Second edition Duxbury Classic.

Grossberg, S., 1976. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. Biological cybernet-

ics, 23.3: 121-134.

Hagan, M. T., and Menhaj, M. B., 1994. Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks, 5(6), 989–993.

Hagan, M. T., Demuth, H. B., and Beale, M. H., and Jesus, O. D., 2014. Neural Network Design, 2nd Edition, Cengage Learning.

Hastie, T., Tibshirani, R., and Friedman, J., 2009. The elements of statistical learning: data mining, inference and prediction. Springer.

Hastie, T., Tibshirani, R., and Wainwright, M., 2015. Statistical learning with sparsity: the lasso and generalizations. CRC press.

He, B., Sun, T., Yan, T., Shen, Y., and Nian, R., 2017. A pruning ensemble model of extreme learning machine with $L_{1/2}$ regularizer. Multidimensional Systems and Signal Processing, 28(3), 1051–1069.

Hebb, D. O., 1949. The organization of behavior: A neuropsychological theory. Wiley, New York.

Hocking, R. R., Speed, F. M., and Lynn, M. J., 1976. A class of biased estimators in linear regression. Technometrics 18:425–438.

Hoerl, A. E., and Kennard, R. W., 1970. Ridge Regression: Applications to Nonorthogonal Problems. Technometrics, 12(1), 69–82.

Hoerl, A. E., Kennard, R. W., and Baldwin, K. F., 1975. Ridge regression:some simulations. Communications in Statistics, 4(2), 105–123.

Hopfield, J. J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79.8:2554-2558.

Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. Neural Networks, 4(2), 251â257

Hotelling, H., 1933. Analysis of a Complex of Statistical Variables into Principal Components. Journal of Educational Pschology, 24, 417-441 and 489-520.

Huang, G. B., 2014. An insight to extreme learning machines: Random neurons, random features and kernels. Cognitive Computation (Online).

Huang, G., B., 2020. Extreme learning machines. http://www.ntu.edu.sg/home/egbhuang. (Accessed 05 March 2020).

Huang, G. B., 2020. Extreme learning machines (ELM):Filling the Gap between Frank Rosenblatt's Dream and John von Neumann's Puzzle? - Tutorial. Nanyang Technological University, Singapore, https://www.ntu.edu.sg/home/egbhuang. (Accessed 15 April 2020).

Huang, G. B., and Babri, H. A., 1998. Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Transactions on Neural Networks, 9(1), 224–229.

Huang, G. B., and Chen, L., 2007. Convex incremental extreme learning machine. Neurocomputing, 70(16), 3056–3062.

Huang, G. B., and Chen, L., 2008. Enhanced random search based incremental extreme learning machine. Neurocomputing, 71(16), 3460–3468.

Huang, G., Huang, G. B., Song, S., and You, K., 2015. Trends in extreme learning machines: A review. Neural Networks, 61, 32–48.

Huang, G. B., Wang, D. H., and Lan, Y., 2011. Extreme learning machines: a survey. International Journal of Machine Learning and Cybernetics, 2(2), 107-122.

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.

Huang, G. B., Zhu, Q. Y., and Siew, C. K., 2004. Extreme learning machine: A new

learning scheme of feedforward neural networks. 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), 2, 985–990.

Huang, G. B., Zhu, Q. Y., and Siew, C. K., 2006. Extreme learning machine: Theory and applications. Neurocomputing, 70(1–3), 489–501.

James, G., Witten, D., Hastie, T., and Tibshirani, R., 2013. An introduction to statistical learning (Vol. 112, pp. 3-7). New York: Springer.

Joliffe, I., 1986. Principal Component Analysis, Springer, Verlag.

Judge, G. G., Hill, R. C., Griffiths, W. E., Lütkepohl, H., and Lee, T. C., 1988. Introduction to the Theory and Practice of Econometrics. John Wiley.

Kibria, B. M. G., 2003. Performance of Some New Ridge Regression Estimators. Communications in Statistics - Simulation and Computation, 32(2), 419–435.

Kim, J., Shin, H., Lee, Y., and Lee, M., 2007. Algorithm for Classifying Arrhythmia using Extreme Learning Machine and Principal Component Analysis. 2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 3257–3260.

Kohonen, T., 1972. Correlation matrix memories. IEEE transactions on computers, 100.4: 353-359.

Lawless, J., F., and Wang, P., 1976. A simulation study of ridge and other regression estimators. Communications in Statistics-theory and Methods, 5(4), 307-323.

Leshno, M., Lin, V. Ya., Pinkus, A., and Schocken, S., 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Networks, 6(6), 861–867.

Li, G., and Niu, P., 2013. An enhanced extreme learning machine based on ridge regression for regression. Neural Computing and Applications, 22(3–4), 803–

810.

Li, K., Peng, J. X., and Irwin, G. W., 2005. A fast nonlinear model identification method. IEEE Transactions on Automatic Control, 50(8), 1211–1216.

Li, L. N., Ouyang, J. H., Chen, H. L., and Liu, D. Y., 2012. A Computer Aided Diagnosis System for Thyroid Disease Using Extreme Learning Machine. Journal of Medical Systems, 36(5), 3327–3337.

Liu, K., 1993. A new class of blased estimate in linear regression. Communications in Statistics-Theory and Methods, 22(2), 393-402.

Liu, X. Y., Gao, C. H., and Li, P., 2012. A comparative analysis of support vector machines and extreme learning machines. Neural Networks, 33, 58–66.

Lu, S., Wang, X., Zhang, G., and Zhou, X., 2015. Effective algorithms of the Moore-Penrose inverse matrices for extreme learning machine. Intelligent Data Analysis, 19(4), 743–760.

Martínez, J. M. M., Escandell, P. M., Soria, E. O., Martín, J. D. G., Magdalena, R. B., and Gómez, J. S., 2011. Regularized extreme learning machine for regression problems. Neurocomputing, 74(17), 3716–3721.

McCulloch, W. S., and Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), 115–133.

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.

Miche, Y., Heeswijk, M. V., Bas, P., Simula, O., and Lendasse, A., 2011. TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization. Neurocomputing, 74(16), 2413–2421.

Minsky, M., and Papert, S., 1969. Perceptrons. Cambridge, MA. MIT Press.

Montgomery, D. C., Peck, E. A., and Vining, G. G., 2012. Introduction to linear regression analysis (Vol. 821). John Wiley & Sons.

Murphy, K. P., 2012. Machine learning: a probabilistic perspective. MIT press.

Nian, R., He, B., and Lendasse, A., 2013. 3D object recognition based on a geometrical topology model and extreme learning machine. Neural Computing and Applications, 22(3-4), 427-433.

Osborne, M. R., Presnell, B., and Turlach, B. A., 2000. A new approach to variable selection in least squares problems. IMA journal of numerical analysis, 20(3), 389-403.

Penrose, K., Nelson, A., and Fisher, A., 1985. Generalized Body Composition Prediction Equation for Men Using Simple Measurement Techniques (abstract), Medicine and Science in Sports and Exercise, 17(2), 189.

Poria, S., Cambria, E., Winterstein, G., and Huang, G. B., 2014. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. Knowledge-Based Systems, 69, 45-63.

Rao, C.R., and Mitra, S.K, 1971. Generalized Inverse of Matrices and its Applications, Wiley, New York.

Reilly, D. L., Cooper, L. N., and Elbaum, C., 1982. A neural model for category learning. Biological cybernetics, 45.1: 35-41.

Rong, H. J., Suresh, S., and Zhao, G. S., 2011. Stable indirect adaptive neural controller for a class of nonlinear system. Neurocomputing, 74(16), 2582-2590.

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review, 65.6: 386.

Rosset, S., and Zhu, J., 2007. Piecewise linear regularized solution paths. The

Annals of Statistics, 35(3), 1012–1030.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., 1988. Learning representations by back-propagating errors. Cognitive modeling, 5.3: 1.

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural networks, 61, 85–117.

Schott, J. R., 2005. Matrix analysis for statistics. John Wiley & Sons, Hoboken.

Schwarz, G., 1978. Estimating the dimension of a model. The annals of statistics, 6(2), 461-464.

Singh, B., Chaubey, Y. P., and Dwivedi, T. D., 1986. An Almost Unbiased Ridge Estimator. Sankhyā: The Indian Journal of Statistics, Series B (1960-2002), 48(3), 342–346.

Singh, L., Chetty, G., and Sharma, D., 2012. A Novel Approach to Protein Structure Prediction Using PCA or LDA Based Extreme Learning Machines. In T. Huang, Z. Zeng, C. Li, and C. S. Leung (Eds.), Neural Information Processing (Vol. 7666, pp. 492–499). Springer Berlin Heidelberg.

Sjöstrand, K., Clemmensen, L. H., Larsen, R., Einarsson, G., and Ersbøll, B., 2018. SpaSM: A MATLAB Toolbox for Sparse Statistical Modeling. Journal of Statistical Software, 84(10).

Stein, C., 1956. Inadmissibility of Usual Estimator for the Mean of a Multivariate Normal Distribution. Proceeding of the Third Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, Berkeley, 197-206.

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.

Tang, J., Deng, C., Huang, G. B., and Zhao, B., 2015. Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine. IEEE Transactions on Geoscience and Remote Sensing, 53(3), 1174-1185.

Theodoridis, S., 2015. Machine learning: a Bayesian and optimization perspective. Academic Press.

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288.

Torgo, L., 2019. Regression datasets. http://www.dcc.fc.up.pt/˜ltorgo/Regression/DataSets.html. (Accessed on 18 April 2019).

Vapnik, V., 1995. The nature of statistical learning theory. Springer-Verlag Berlin, Heidelberg.

Wang, J. N., Jin, J. L., Geng, Y., Sun, S. L., Xu, H. L., Lu, Y. H., and Su, Z. M., 2013. An accurate and efficient method to predict the electronic excitation energies of BODIPY fluorescent dyes. Journal of computational chemistry, 34(7), 566-575.

Wang, Y., Cao, F., and Yuan, Y., 2011. A study on effectiveness of extreme learning machine. Neurocomputing, 74(16), 2483–2490.

Widrow, B., and Hoff, M. E., 1960. Adaptive switching circuits. In: IRE WESCON convention record, 96-104.

Wieringen, W. N. V., 2015. Lecture notes on ridge regression. arXiv:1509.09169.

Wilamowski, B. M., and Hao Y., 2010. Neural Network Learning Without Back-propagation. IEEE Transactions on Neural Networks, 21(11), 1793–1803.

Wong, W. K., and Guo, Z. X., 2010. A hybrid intelligent model for medium-term sales forecasting in fashion retail supply chains using extreme learning ma-

chine and harmony search algorithm. International Journal of Production Economics, 128(2), 614-624.

Yan, L., 2006. A PCA-based PCM data analyzing method for diagnosing process failures. IEEE transactions on semiconductor manufacturing, 19(4), 404-410.

Yan, Z., and Wang, J., 2014. Robust model predictive control of nonlinear systems with unmodeled dynamics and bounded uncertainties based on neural networks. IEEE transactions on neural networks and learning systems, 25(3), 457-469.

Yıldırım H., and Özkale M. R., 2019. The performance of ELM based ridge regression via the regularization parameters. Expert Systems with Applications 134, 225-233.

Yıldırım H., and Özkale M. R., 2020. An enhanced extreme learning machine based on Liu regression. Neural Processing Letters, Doi: 10.1007/s11063-020-10263-2.

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

Yuan, Y., Wang, Y., and Cao, F., 2011. Optimization approximation solution for regression problem based on extreme learning machine. Neurocomputing, 74(16), 2475–2482.

Zhang, H., Zhang, S., and Yin, Y., 2013. An improved ELM algorithm based on EM-ELM and ridge regression. In: 2013 International Conference on Intelligent Science and Big Data Engineering (pp. 756-763). Springer, Berlin, Germany, pp. 756-763.

Zhang, H., Yin, Y., and Zhang, S., 2016. An improved ELM algorithm for the measurement of hot metal temperature in blast furnace. Neurocomputing, 174, 232–237.

Zhao, J., Wang, Z., and Park, D. S., 2012. Online sequential extreme learning machine with forgetting mechanism. Neurocomputing, 87, 79-89.

Zhao, Y., Yu, S., Chu, B., Zhang, N., and Hu, X., 2013. Classification of Three Wine Varieties Based on ELM and PCA. In J. Yang, F. Fang, & C. Sun (Eds.), Intelligent Science and Intelligent Data Engineering (Vol. 7751, pp. 647–654). Springer Berlin Heidelberg.

Zhao, Z., Li, P., and Xu, X., 2013. Forecasting model of coal mine water inrush based on extreme learning machine. Applied Mathematics & Information Sciences, 7(3), 1243.

Zheng, W., Qian, Y., and Lu, H., 2013. Text categorization based on regularization extreme learning machine. Neural Computing and Applications, 22(3-4), 447-456.

Zou, H., and Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301–320.

CURRICULUM VITAE

Hasan YILDIRIM was born in Gaziantep, in 1988. He completed his primary and secondary education in Gaziantep, high school education in Adana. He received B.E degree from the statistics department of Anadolu University in 2012. In the year of graduating, he was appointed as a research assistant to Karamanoğlu Mehmetbey University by an official academic program (a.k.a ÖYP). After a while, he started his graduate education in the department of statistics of Çukurova University. He received his MSc degree from same department in 2015. He is still a research assistant in Karamanoğlu Mehmetbey University and has been carried out his PhD studies in Çukurova University, the Department of Statistics. His research interests include statistical machine learning, problems in high dimensional data and advanced statistical data analysis. He is married and father of a son. His email address is hasanyildirim@kmu.edu.tr.