FEBRUARY 2020 M.Sc. in Electrical and Electronics Engineering EMRAN ALCHIKH ALNAJAR

REPUBLIC OF TURKEY

GAZİANTEP UNIVERSITY

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES

SPARSE EXTREME LEARNING MACHINE FOR CLASSIFICATION

M.Sc. THESIS

IN

ELECTRICAL AND ELECTRONICS ENGINEERING

BY

EMRAN ALCHIKH ALNAJAR

FEBRUARY 2020

SPARSE EXTREME LEARNING MACHINE FOR CLASSIFICATION

M.Sc. Thesis

in

Electrical and Electronics Engineering

Gaziantep University

Supervisor

Assoc. Prof. Dr. Sema KOÇ KAYHAN

by

Emran ALCHIKH ALNAJAR

February 2020

REPUBLIC OF TURKEY
GAZİANTEP UNIVERSITY
GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES
ELECTRICAL AND ELECTRONICS ENGINEERING

Name of the Thesis : Sparse Extreme Learning Machine for Classification

Name of the Student : Emran ALCHIKH ALNAJAR

Exam Date : 05.02.2020

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. A. Necmeddin YAZICI
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ergun ERÇELEBİ
Head of Department

This is to certify that we have read this thesis and that in our consensus opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Sema KOÇ KAYHAN
Supervisor

Examining Committee Members:	Signature
Prof. Dr. Kemal DELİHACIOĞLU	……..
Assoc. Prof. Dr. Sema KOÇ KAYHAN	……..
Asst. Prof. Dr. Serkan ÖZBAY	……..

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Emran ALCHIKH ALNAJAR

ABSTRACT

SPARSE EXTREME LEARNING MACHINE FOR CLASSIFICATION

ALCHIKH ALNAJAR, Emran
M.Sc. in Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Sema KOÇ KAYHAN
February 2020
45 pages

Extreme Learning Machine (ELM) is a single-hidden-layer feed forward neural network that randomly initiates the weights for the connections between input and hidden layer and the bias of the hidden layer. Two hidden layers and multiple hidden layers ELMs have added some enhancements – when used on the same data set - keeping the randomly initiated weights and biases and extending the number of hidden layers. A greedy algorithm is also proposed within the ELM architecture to penetrate sparse approaches and presented enhancement in compare with standard. This study presents three different standard ELM algorithms and a greedy single hidden layer ELM to detect classification problems on 3 datasets and compares the testing accuracy of these total 4 ELM algorithms. Based on the results of experiments conducted, greedy Extreme Learning Machine algorithm provide a good prediction accuracy with a high prediction rate.

Key Words: Extreme Learning Machine, Sparse ELM, Artificial Neural Network, Classification

ÖZET

SINIFLANDIRMA İÇİN SEYREK AŞIRI ÖĞRENME MAKİNESİ

ALCHIKH ALNAJAR, Emran
Yüksek Lisans Tezi, Elektrik Elektronik Mühendisliği
Danışman: Doç. Dr. Sema KOÇ KAYHAN
Şubat 2020
45 sayfa

Aşırı Öğrenme Makinesi (AÖM), giriş ve gizli katman arasındaki bağlantıların ağırlıklarını ve eşik değerlerini rastgele seçen, tek gizli katmanlı ileri beslemeli bir sinir ağıdır. İki gizli katmanlı ve çoklu gizli katmanlı AÖM, yine rasgele seçilen ağırlık ve eşik değerlerini kullanarak, -aynı veri seti için- gizli katman sayısının artırılmasıyla iyileştirmeler sunmaktadır. Bu çalışmada, AÖM mimarisinde seyrek yaklaşımın avantajlarından faydalanmak için standart AÖM'ye kıyasla daha iyi sonuçlar veren, açgözlü bir AÖM algoritması önerilmiştir. Ayrıca, üç farklı standart AÖM algoritması ve açgözlü tek gizli katmanlı AÖM algoritması, üç ayrı veri kümesindeki sınıflandırma problemleri için uygulanmış ve algoritmaların test doğruluğu karşılaştırılmıştır. Yapılan deneylerin sonuçlarına dayanarak, açgözlü AÖM, diğer AÖM'lere göre hem daha hızlı hem de daha yüksek tahmin oranı ile daha yüksek doğrulukta kestirim yapabilmektedir.

Anahtar Kelimeler: Aşırı Öğrenme Makinesi (AÖM), Seyrek AÖM, Yapay Sinir Ağları.

"Dedicated to my father"

ACKNOWLEDGEMENTS

It has been a great opportunity to seek my master degree in Turkey, Gaziantep where I have living for almost 5 years. It took me 10 years to return to academic field after graduating on 2010 back in Syria. I feel grateful for Gaziantep University academic staff starting from the acceptance on 2017 till defense date in 2020.

I would like to thank my supervisor, Assoc. Prof. Dr. Sema KOÇ KAYHAN for her guidance and support throughout the study. I am thankful for her encouragement and motivation. I have reached a point to give up my research but her commitment and support has recharged me again specially when she highlighted important research areas and directed me toward the correct direction during thesis period.

Finally, a great appreciation to my family specially my father how has passed away on 2018. He would be very proud there in the sky. My daughter Maryana, although she is 5 years old, has been a great motivation for better future and an academic record to the family would hopefully lead to a better future for all of us. Thanks for my lovely wife for her support specially during the stressful times we have been though.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS

ELM	Extreme learning machine
TELM	Two hidden later extreme learning machine
MELM	Multi hidden layers extreme learning machine
AI	Artificial Intelligent
ANN	Artificial Neural Networks
MP	Moore-Penrose inverse
OMP	Orthogonal matching pursuit
RMSE	Root mean square error
MS	Milli second(s)
ELMs.	Extreme learning machine algorithms

CHAPTER I

INTRODUCTION

1.1 Motivation of Study

Artificial Intelligence (AI) fields have been a trend in the recent years because of the development of humanity in multi direction and specially technology. Machine Learning under AI umbrella has grabbed attention to bring machines to act smart and learn. Extreme learning machine has taken our attention since is its release on 2004 (Huang et al. 2004) specially after multi enhancements approaches to extra hidden layers. Furthermore, Sparsity approaches have attracted our attention and integrating its results with extreme machine learning have become our direction of research. Real time learning has been a direction of multiple researches to target fast learning for machine learning algorithms.

CHAPTER 2

LITERATURE REVIEW

In the literature review section, collection of most relevant and significant researches and studies will be presented from various topics related to this thesis topic, Sparse Extreme Learning Marching for Classification. These are related to machine learning algorithms and especially extreme (ELM) and some of its enhancements by various of researches. Focusing on the classification problems and trying to reach highest accuracy, all algorithms and researches tried to put extra features and /or change the structure of the machine learning algorithms.

Literature review section includes general information about machine learning and focuses on 4 types of machine learning algorithms as neural networks.

2.1 Machine Learning - ML

Machine learning is the part of artificial intelligence that is focused on real life problems. It provides chance for the machine to learn based of a training data set and developed an artificial model to predict probabilities, classify outputs based on specific features and solve regression problems. Artificial Neural Networks ANN (Mandal and Sairam, 2011) is a set of algorithms that is used in machine learning to develop models using graphs of neurons which is derived from real human neurons idea of activating 2 neurons using the links between them. Multiple machine learning algorithms were established based on ANNs such as learning rules varies from backpropagation by taking the error (loss) obtained in proceeding iteration and feed it in to the new iteration to get closer to target output. Feed forward neural networks are the type of nets where information is feeding only in one direction, from input nodes toward outputs through hidden nodes in between and no iterations are adopted here.

Machine learning is a kind of artificial intelligence (AI) which allows computers to be trained on solving problems without being directly programmed. ML. ML algorithms are generally classified into three types named supervised learning, unsupervised learning and reinforcement learning and are shown in Figure 2.1 (Praveena and Jaiganesh, 2017). The development of machine learning is comparable to development of data mining. Both data mining and machine learning reflect or discover from end to end data to assume for patterns finding. On the other hand, in cases of extracting data for persons knowledge and same for data mining applications; machine learning extracts opportunities from the data to identify outlines/patterns in data and fine-tune program activities as a result (Praveena and Jaiganesh, 2017).

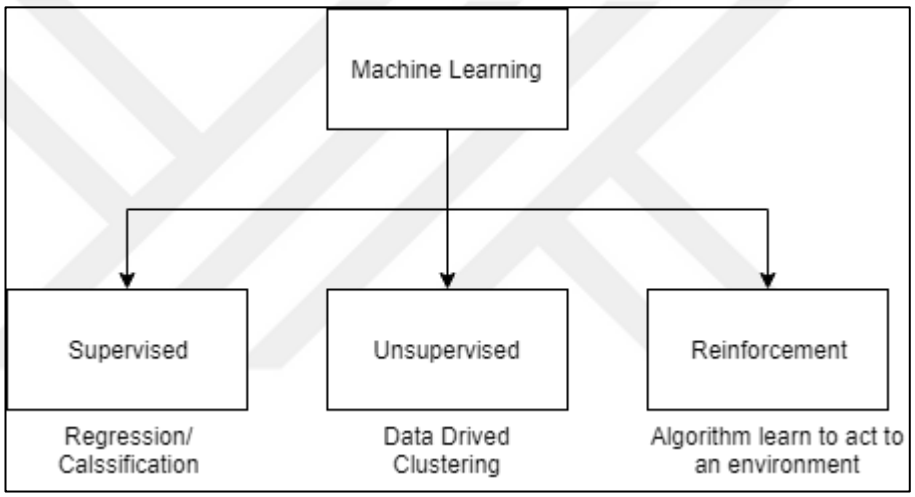

Figure 2.1 Machine Learning types

Supervised machine learning is aims identify meaning from labelled training data which has a set of training examples. These training examples includes the desired results for either regression problems or classification ones. As far as supervised learning is studied, each example is a pillar containing an input features/attributes and an enforced output value.

Initially, supervised learning algorithm does some analysis from the training data and builds a depending function, in order to map new inputs/outputs pairs. A threshold setting might smooth the algorithm to precisely courage the class labels for scanned instances and the same needs the supervised learning algorithm to decrease from the training data to cover states in a balanced manner. The supervised approaches are used

in various application areas that include advertising, economics, engineering, testing, stock market prediction, and so on.

Furthermore, there are couple steps that Supervised Machine Learning Algorithms (Praveena and Jaiganesh, 2017) follow in general:

Step 1. Establish the types of training dataset: the user might need to modify the type(s) of input and/or data that will be used as a training set;
Step 2. Converge a training set: the training set is desired to be delegate of the real-world use of the function. As an effort, a set of input attributes is composed that remains and corresponding outputs are also composed;
Step 3. Resolve the input attribute design of the learned function / learned feature. The accurateness of the learned function is tightly associated with the input attributes representation;
Step 4. Resolve for the construction of the learned function and comparable machine learning algorithm;
Step 5. Integrate the design and perform the learning algorithm on the collected training dataset;
Step 6. Evaluate the accuracy / exactness of the learned function. Then, parameters are observed with and learning may be performed on the resulting function and should also be applied on a test dataset which different from training one.
Step 7. Time performance could be calculated when applying the training and while doing the testing. Training and testing times can be a factor when comparing multiple machine learning algorithms.

There are multiple factors to be considered when machine learning algorithms are studied:

1- Data diverseness: when the features contain various data types which includes separate, discrete ordered, totals and continuous decimal values, some algorithms are simpler to implement than rest of the algorithms. Many such algorithms specifically - linear regression, logistic regression, artificial neural networks, Support Vector Machines and nearest neighbor methods, desire that the input attributes/features be numerical and scaled to similar ranges.

2- Data redundancy: it happens when he input features have extra unwelcomed information. Rarely, learning algorithms may execute badly due to numerical weakness. Such researches issues might overcome by apply some pre-processing actions on the dataset, such as normalization.
3- Non-linearity and interactions: this scenario occur when input attributes make an independent role to the output, then algorithms that are based on linear / distance functions typically fit. On the other hand, when there are complex relations amongst features, then some algorithms give better results in terms of accuracy, as they are typically intended to identify these relations.

One of the famous enhancements to feed forward neural network is called the extreme neural network and it is the main direction of this study.

The following section highlights areas of related work focused on the supervised machine learning.

2.1.1 Related Work on Supervised Machine Learning

Within the supervised machine learning, decision tree and supper vector machine are surveyed.

Firstly, for decision tree wise, Lertworaprachaya et al. suggested a new approach for composing decision trees using interval-valued fuzzy membership values (Lertworaprachayaa et al. 2014). Generally, most fuzzy decision trees do not contemplate the concerned related values with their membership; though, exact values of fuzzy membership values are not always achievable. Thus, the authors characterized fuzzy membership values as distance to model concerned and engage the look-ahead based fuzzy decision tree induction method to build decision trees. The authors also measured the value of different neighborhood values and defined a new parameter severe to precise data sets using fuzzy sets. Some examples are provided to proof the efficiency of their model.

Also, Bahnsen et al. have suggested an example-dependent cost-sensitive decision tree approach, by integrating different example-reliant costs into a proposed cost-based impurity measure and new cost-based pruning methodology (Bahnsen et al. 2015). Subsequently, using three different databases, from three real-world examples namely

credit card scam detection, credit scoring and direct marketing, the authors evaluated their proposed method. Their outcomes displayed that their anticipated algorithm is the best performing proposal for all studied databases. Moreover, when equated with a typical decision tree, their proposal has built significantly reduced trees size in only 20% of the time, while having an outstanding performance weighted by cost reserves, leading to a model that not only has more business-oriented outcomes, but also a model that has simpler designs that are easier to understand.

Using fuzzy decision trees for prediction of learning themes in conversational intelligent training systems has been presented by (Crockett et al. 2016). Predicting learning style is accepted by locking independent performance parameter during the teaching reflection with the peak value variable. They have admitted weakness in their approach that it does not consider the connections between behavior parameters and, due to the doubt present in modelling learning techniques, minor changes in behavior can lead to unfitting guesses. Subsequently, the learner is offered with supervision material not appropriate with their learning themes. So, the mentioned challenges, a new approach that uses fuzzy decision trees to create a sequence of fuzzy predictive representations connecting these parameters for all scopes/dimensions of the Felder Silverman Learning Styles model. According to their results and by the use of live data, it is observed that the fuzzy models have developed the anticipate accuracy across four learning themes dimensions and enabled the detection of some thought-provoking connections amongst behavior parameters.

Secondly, for support vector machine, it is noticed that standing organizational classification algorithms do not have a sense of balance for structural information's connections both intra-class and inter-class. Chen et al. have proposed a new structural none-parallel support vector machine namely SNPSVM which is linking the structural information with nonparallel support vector machine (NPSVM) (Chen et al. 2016). Within SNPSVM examine all models and not only the awareness in both categories by the structural information but also the ability to be fixed between categories. Thus, it can completely explorer previous information to recover the algorithms generalization size straightly. Furthermore, they implemented the improved Alternating Direction Designed of Multipliers (ADMM) to SNPSVM. Both their model itself and the resolving algorithm can assure that it probably would manage

large-scale classification difficulties with a big number of occurrence as well as attributes. Their experimental outcomes show that the proposed SNPSVM is larger than the other existing procedures based on time performance and accuracy perspectives.

Additionally, Linear kernel support vector machine (SVM) as a regularized least-squares (RLS) problem, have been proposed by Peng et al (Peng et al., 2016). By an equation, which includes the error vector to the indicator values, have represented the solution for this RLS. By applying segmentation to the training dataset, the SVM weights and direction are stated systematically using the support vectors. Furthermore, they have identified the methodology that their approach logically spreads to calculations with none-linear kernels whilst deflect the necessity to use of Lagrange multipliers and duality philosophy. A fast-constant solution process based on Cholesky breakdown with adjustments of the support vectors is suggested as a solution process. Using a simple example that can be decorated visually, the settings of their proposed SVM design were inspected and correlated with typical SVMs. The precision and behavior of their proposed work have been verified using a set of public benchmarking problems for both linear and nonlinear SVMs.

Another proposal by Utkin and Zhuk has been launched on 2016. It includes a well-known one-class classification support vector machine (OCC SVM) which is interact with interval-valued or set-valued training dataset (Utkin and Zhuk, 2016). They proposed a new representation for all distances within the training dataset. This representation is a determined set of explicit data with rough weights. Their representation is driven by the idea of changing of the interval-valued familiar risk produced by a new interval-valued data with the interval-valued expected risk produced by ambiguous weights or lists of weights. Furthermore, the interval matter is substituted with the ambiguous weight or probabilistic uncertainty. They presented how restrictions for the imprecise weights are combined into dual quadratic programming problems which can be observed as expansion of the known OCC SVM models. The authors designed their model and examine its settings using numerical instances with artificial and real interval-valued training.

2.2 Extreme Learning Machine – ELM

ELM is a single hidden layer feedforward neural network that claims to overcome the slow gradient based learning algorithms problem (Huang et al. 2004). Using the Moore-Penrose (MP) inverse and the smallest norm least-squares solution of a general linear system and take advantage of biological argument saying, "that there must be some parts of the brain where neuron configurations do not depend on the external environment".

Figure 2.2 shows the three layers of a single hidden layer feed forward neural network (SLFNs) ELM. Consider N arbitrary input-target samples $(x_i\ ,\ t_i)\ where\ i = 1,2,3,\dots,N$ where$(x_i\ ,\ t_i)\ where\ i = 1,2,3,\dots,N$ i.e. there is an input matrix $X = [x_1, x_2, x_3, \dots, x_N]^T$ and a desired target matrix $T\ = [t_1, t_2, t_3, \dots, t_N]^T$ where $x_i\ ,\ t_i$ are vectors $x_i\ = [x_{i1}, x_{i2}\ , x_{i3}, \dots, x_{in}]^T \in \mathbb{R}^n$ and $t_i\ = [t_{i1}\ ,\ t_{i2},\ t_{i3}, \dots, t_{im}]^T \in \mathbb{R}^m$, []T denotes to matrix transposition and the parameters n and m are the dimensions of input matrix and output matrix, e.g. n is the number of input attributes and m is the number of target outputs.

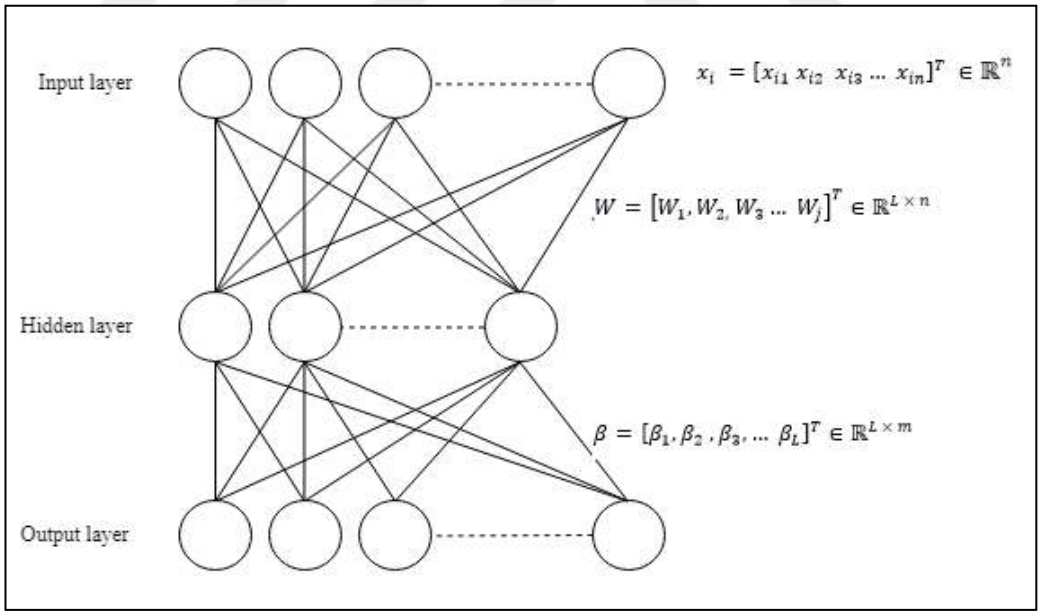

Figure 2.2 Input, hidden, output layers of ELM

ELM assigns random weights & biases between input and hidden layer $W = [W_1, W_2, W_3 \dots W_j]^T \in \mathbb{R}^{L \times n}$, $B = [B_1, B_2\ , B_3, \dots B_L]^T \in \mathbb{R}^{L \times N}$. L is the number of hidden neurons.

ELM assigns random weights and biases for the connection between input and the single hidden layer. These weights and biases do not change during the training, so this allows transforming the nonlinear system to be described by linear one

$$H\beta = T \tag{2.1}$$

Where $\beta = [\beta_1, \beta_2, \beta_3, \dots \beta_L]^T \in \mathbb{R}^{L \times m}$ is the weight matrix between hidden layer and output layer and $\beta_i = [\beta_{i1}, \beta_{i2}, \beta_{i3}, \dots \beta_{im}]^T; i = 1,2,3, \dots, L$. β_i denotes to the connection weights between ith hidden neuron and m output neurons. $H = g(WX + B)\mathbb{R}^{N \times L}$ is the hidden layer output matrix. $g(x)$ is the activation function and the sigmoidal activation function is used in ELM algorithm and the following ones as well (Huang et al. 2004).

The only parameter that needs to be calculated is the outputs weight matrix β and using the method of least-squares β can calculated as:

$$\beta = H^+ T \tag{2.2}$$

Where H^+ is the Moore-Penrose inverse matrix of H matrix. Then the actual output of the training algorithm will be Y.

$$Y = H\beta \tag{2.3}$$

ELM results and performance were compared with back-propagation (BP) and support vector machine (SVM) and it demonstrated high performance and better results using the diabetes data set used in (Huang et al. 2004).

2.3 Two-Hidden Layer ELM - TELM

One of the enhancements for ELM is the Two-Hidden-Layers extreme machine learning TELM (Qu et al. 2015). It has 2 hidden layers with the same number of neurons L. Same as ELM, the connections between input layer and first hidden layer are randomly weighted W, biases are randomly weighted as well B. This initiation is an orthogonal initialization because it was observed that this orthogonalization has produced better performance in the classification problems (Quet al. 2015). When the weight/bias matrix is not squired semi-orthogonal matrix is applied using (Orth)

function. For simplicity W_{IE} is known as $[B\ W]$ and similarly X_E is known as $[1\ X]^T$ where 1 denotes to one vector of size N where its elements are all scalar 1.

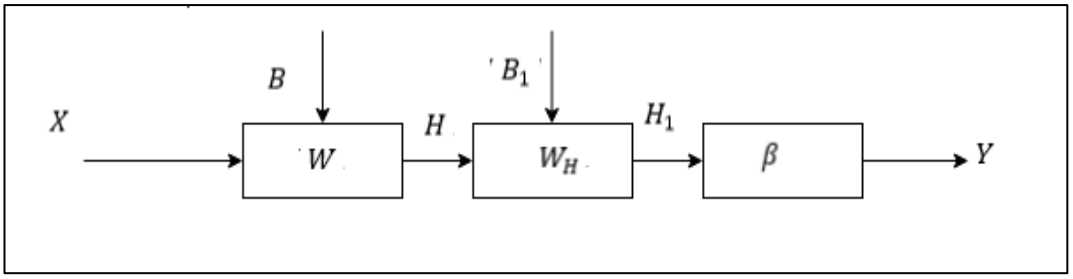

Figure 2.3 The workflow of the TELM

The next step in this algorithm is calculating the output from first hidden layer H and using $g(x)$ as the activation function. Again, sigmodal function is used.

$$H = g(W_{IE}X_E) \tag{2.4}$$

Using Moore-Penrose inverse of output matrix from first hidden layer H and to get the target outputs T with minimum squared error, the weight matrix between the second hidden layer and the output is calculated as

$$\beta = H^+ T \tag{2.5}$$

The workflow of TELM architecture is depicted in Figure 2.3. W_H denotes to the weight matrix between 1st and 2nd hidden layers. H_1, B_1matrices respectively represent the weight and bias of the expected output. According to Figure 2.3.

$$g\ (W_H\ H + B_1) = H_1 \tag{2.6}$$

But the expected output H_1 of the second hidden layer is calculated as

$$H_1 = T\ \beta^+ \tag{2.7}$$

where β^+ is the MP generalized inverse.

Subsequently, W_{HE} consists of two components, a matrix and a vector. The first is the matrix of the connection weight matrix between first and second hidden layers. The

second component is the bias of the second hidden layer, defined as $[B_1\ W_H]$ and calculated as

$$W_{HE} = g^{-1}(H_1)\ {H_E}^{+} \tag{2.8}$$

Where $g^{-1}(x)$ is the inverse function of the activation function and $H_E = [1\ H]^T$ same as we defined X_E before. Then the actual outputs of the second hidden layer can be obtained as

$$H_2 = g(W_{HE}\ H_E) \tag{2.9}$$

The weight matrix between the second hidden layer and the output can be recalculated as

$$\beta_{new} = {H_2}^{+}\ T \tag{2.10}$$

After obtaining the output weights β_{new} by the training, the final output of the TELM is expressed as

$$Y = H_2\ \beta_{new} \tag{2.11}$$

The TELM has introduced a novel method to obtaining parameters of the 2nd hidden layer which is the connection weights and bias between 1st and 2nd hidden layer. Orthogonality of the randomly initiated weights and biases might have played a part of get improved performance and results as experiments showed on regression and some classification problems (Qu et al. 2015)

2.4 Multi Hidden Layer ELM - MELM

Another enhancement for ELM was the multi-hidden layers MELM (Xiao et al. 2017) which is extended from TELM. The 1st hidden layer is still aa first hidden layer, while the 2nd hidden layer looks as a merged from 2 hidden layers considering 3 hidden layers. MELM proposed unlimited number of hidden layers but experiments show that the more hidden layers don't lead to better performance or higher accuracy rates. The

workflow of three hidden layers is illustrated in Figure 2.4. An orthogonal initiation is also adopted for MELM like TELM before.

The initial steps for MELM are the same as TELM using the formulas (2.4) to (2.10) to find the weight matrix β_{new}. So for thee hidden layers, the updated parameters are calculated by recycling the operation repeating formulas (2.7)(2.8)(2.9)(2.10) to obtain the revised weight matrix β_{new} between last hidden layer and output. Resetting $\beta_{new} = \beta$ & $H_E = [1\ H_2]^T$ is required before the recycle. For four hidden layers, two recycle are needed and so on.

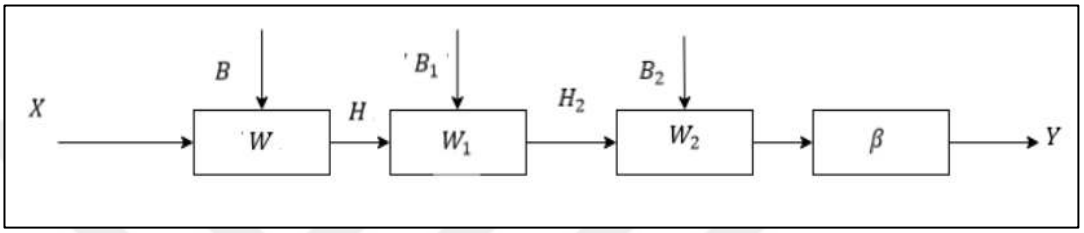

Figure 2.4 The workflow of three hidden layer ELM

Figure 2.5 shows the structure of MELM considering 3 hidden layers for simplicity. The output weights β_{new} of the last hidden layer is used to calculate the final results of MELM

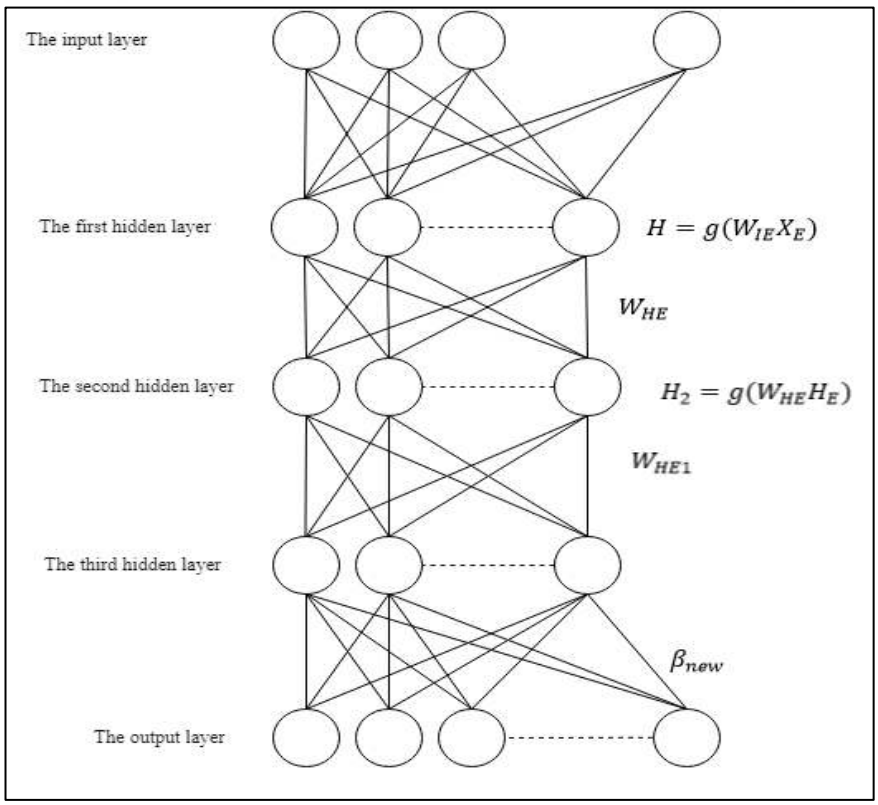

Figure 2.5 The structure of three hidden layer ELM

2.5 Sparse Representations

Sparse representation for a signal or data is produced when only few non-zero coefficients are produced represent a signal based on a dictionary which is normally an overcomplete dictionary.

Sparse coding is in the context of unsupervised learning which is using only the input y_i for learning. The idea is to find a latent representation x_i for each input y_i such that $x_i$is sparse vector. Means vector x_i has many zeros and we can reconstruct the original input y_i as well as possible. Sparse coding algorithms are mentioned in details in the Section 2.5.1.

Formally as an objective function:

$$\min_{D} \frac{1}{T} \sum_{t=1}^{T} \min_{x^{(t)}} \frac{1}{2} \left\| y^{(t)} - D\, x^{(t)} \right\|_2^2 + \lambda \left\| x^{(t)} \right\|_1 \tag{2.12}$$

Subject to $\|D_i\|_2^2 \leq C: given\ constant; i = 1, \dots, N$

$\lambda \left\| x^{(t)} \right\|_1$ is the sparsity penalty, λ is coefficient of sparsity.

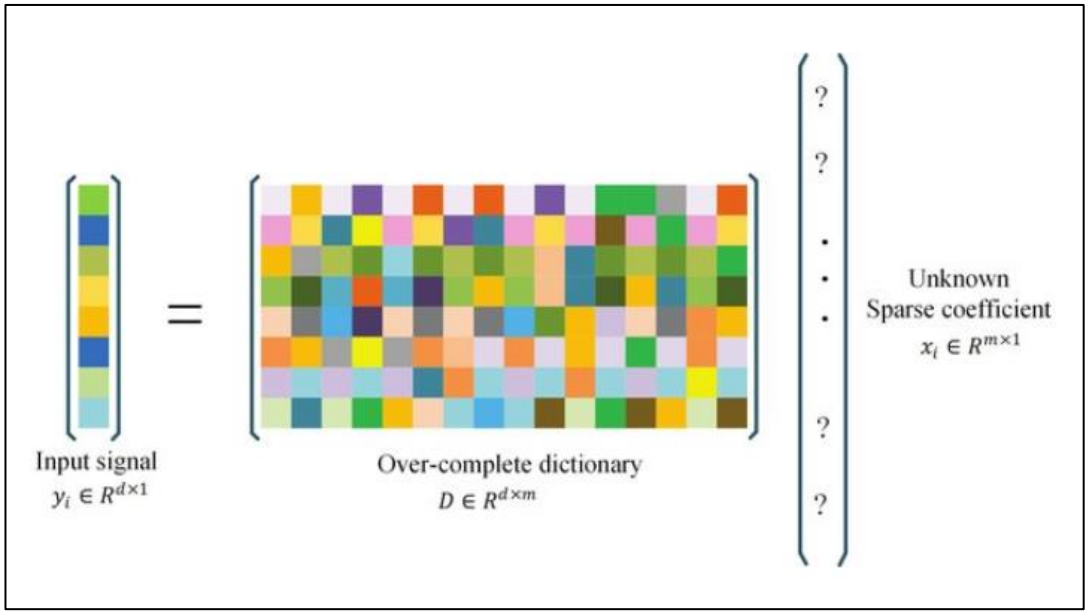

Figure 2.6 Sparse coding of a signal

Since sparse coding problem is NP hard, solution can be by approximation algorithms (Elad, 2018). Multiple solutions are there such as convex relaxation methods and greedy methods, we are focusing on the greedy methods

2.5.1 Sparse Coding Algorithms

Exploring By exploring the interesting domain of simple neurons in the graphic stripe cortex of cats, Hubel and Wiesel suggested that the receptive field of main visual cortex -i.e., V1 neurons- can produce a sparse representation for visual signal (Hubel and Wiesel, 1959). Other electrophysiological researchers have confirmed that the sparse coding principle had been existed in the visual cortex. These findings encouraged engineering researchers to advance sparse coding algorithms for signal processing.

Optimization methods were used in attempts to encode sparse representation of signals. Most of optimization methods pursue the sparse codes with the lowest number of none zero factors for a dictionary by minimizing l_0 norm as illustrated in Figure 2.6.

Couple researchers have been proposed multiple approaches to find suitable substandard solutions to the aforementioned optimization challenge. Such as the basis pursuit, matching pursuit and its modified version the orthogonal matching pursuit.

2.5.2 Basis Pursuit

The solution is simply changing l_0 by l_1 and then the problem become convex and solution is manageable.

When signal's sparsity is approaching the number non-orthogonal atoms, the basis pursuit method can produce a similar solution to l_0 optimization problem (Donoho and Elad, 2002). For noisy signals, an improved algorithm called basis pursuit de-noising (BPDN) (Chen et al. 2001) has proposed a balance achieved between sparsity and rebuilding error. BPDN procedure can obtain the sparsest approximation of l_0 norm minimization specifying rebuilding quality. BPDN has inspired and were used for many enhanced algorithms such as gradient projection (GP) and feature-sign search (FSS). In comparison with greedy strategy-based approaches, BPDN algorithm has

better effectiveness and computationally efficiency. Thus, this algorithm and its enhancements have been broadly used in application.

2.5.3 Matching Pursuit - MP

Matching Pursuit (MP) is a greedy algorithm which basically swap among the atoms and check how to bring the two sides of the equation closer together. So, it finds the correlation between what we can call residual and a candidate atom of feature to be added to the next iteration.

In MP, the best column is selected which reflect the least error, then it is necessary to update the weight vector and residual. At each step of the iteration, updating is fed back until the constraint of the MP is matched. Means if the sparsity needed level is k, the algorithm stops after k integration.

2.5.4 Orthogonal Machine Pursuit - OMP

The OMP algorithm is a modification of MP in a logic of it can ensure that the identified atoms and the recent residual are orthogonal at each loop cycle. Thus, the OMP algorithm converges much faster than the MP one.

In the orthogonal matching pursuit (OMP), an extra step is placed. All vectors that are utilized and new selected atom/vector/feature and semantically update the values of the coefficient vector.

2.6 Sparse And ELM

Multiple researches have tried to implement sparse representation with extreme learning marching algorithms. Majority were focused on the sparsity of the input features. Sun and Yu have proposed sparse coding ELM which maps the input feature vector into a sparse representation (Sun and Yu, 2017). In sparse coding ELM, no changes to the standard ELM supervised learning for the output weights and gradient projection based was used for the sparse coding problem. Sparse Representation Coding (SRC) was also used for face recognition (Wright et al. 2009) by Wright et al. SRC have anticipated dictionary initiation using the training data set with no further training for the dictionary atoms. SRC could achieve fast training time and high recognition accuracy. It is noticed that classification problems were the main focus

when sparse and extreme learning machine algorithms are studied (Sun and Yu, 2017), (Ray and Raj, 2018). Different neural networks and classifiers are employed to classify the extracted sparse features from ECG signals (Ray and Raj, 2018). The new sparse features were decomposed such as time delays, frequencies, the width parameters of ECG and the square of expansion coefficient for each of the dictionary atoms.

Convex relaxation and greedy algorithms are used to provide approximation solutions for sparse coding problem. The two widely using greedy approaches are the matching pursuit and orthogonal matching pursuit (Ray and Raj, 2018).

Alcin et al. have proposed sparse approximation for single hidden layer ELM called GA-SELM (Ince et al. 2014). They have focused on some factors such as 1) the challenge to choose the number of hidden neurons, 2) the irrelevant variables of the input training dataset and 3) the singularity problem that might appear. Greedy algorithms were used to suggest an arbitrary sparse vector output weight and then calculate an estimated output using these output weights and hidden layer weights and biases. Next step was using greedy algorithms to find a specific k-sparse new output weights based on the estimated output and calculating the root mean squared error (RMSE). By iterating k-sparse value to reach the needed number of hidden neurons, different estimated outputs/ RMSEs could be calculated and the minimum RMSE refers to the best results and corresponding k-sparse would represent the number of needed hidden neurons.

CHAPTER 3

PROPOSED SCHEME

As shown in Figure 3.1, Greedy extreme learning machine interferes in the single hidden layer feed forward extreme learning machine in the output weights calculation part. Rather than using the Moore-Penrose invers to calculate the output weights β as in equation (2.2), the greedy ELM follows the following steps to propose output weights:

1- An interim arbitrary vector β_{arb} is randomly initiated where it is k-sparse, means k none-zero values are initiated, then an interim output Y_1 is calculated. Same techniques as the used in equation (2.3) of the SLFNs.

$$Y_1 = H\ \beta_{arb} \tag{3.1}$$

2- The interim output Y_1 is used with the H to calculate the k-sparse output weight vector β_{temp} using the ortogonal matching pursuit algorithms as a greedy approximation.

$$\beta_{temp}\ = OMP(Y_1, H, k) \tag{3.2}$$

3- An estimated output $Y^{\wedge}$ is considered based on H and the k-sparse output weight vector

$$Y^{\wedge} = H\ \beta_{temp} \tag{3.3}$$

4- The root mean squared error performance function donated RMSE is used between the estimated output $Y^{\wedge}$ and the target values of the training dataset. RMSE values are observed and compared with a stored RMSE array.

$$error = \sqrt{mean(T - Y^{\wedge})^2} \tag{3.4}$$

Greedy ELM compares all RMSE and increase the k values and steps number one is repeated again until k reaches the suggested number of hidden neurons of the ELM.

The minimum RMSE corresponds to the best output that is closest to the target values and output weight β_{temp} that is associated with minimum RMSE is considered as the final output weights that is used on the testing phase of the extreme learning machine. Testing accuracy percentage and k-sparsity level are extracted as last step for the each minimum RMSE.

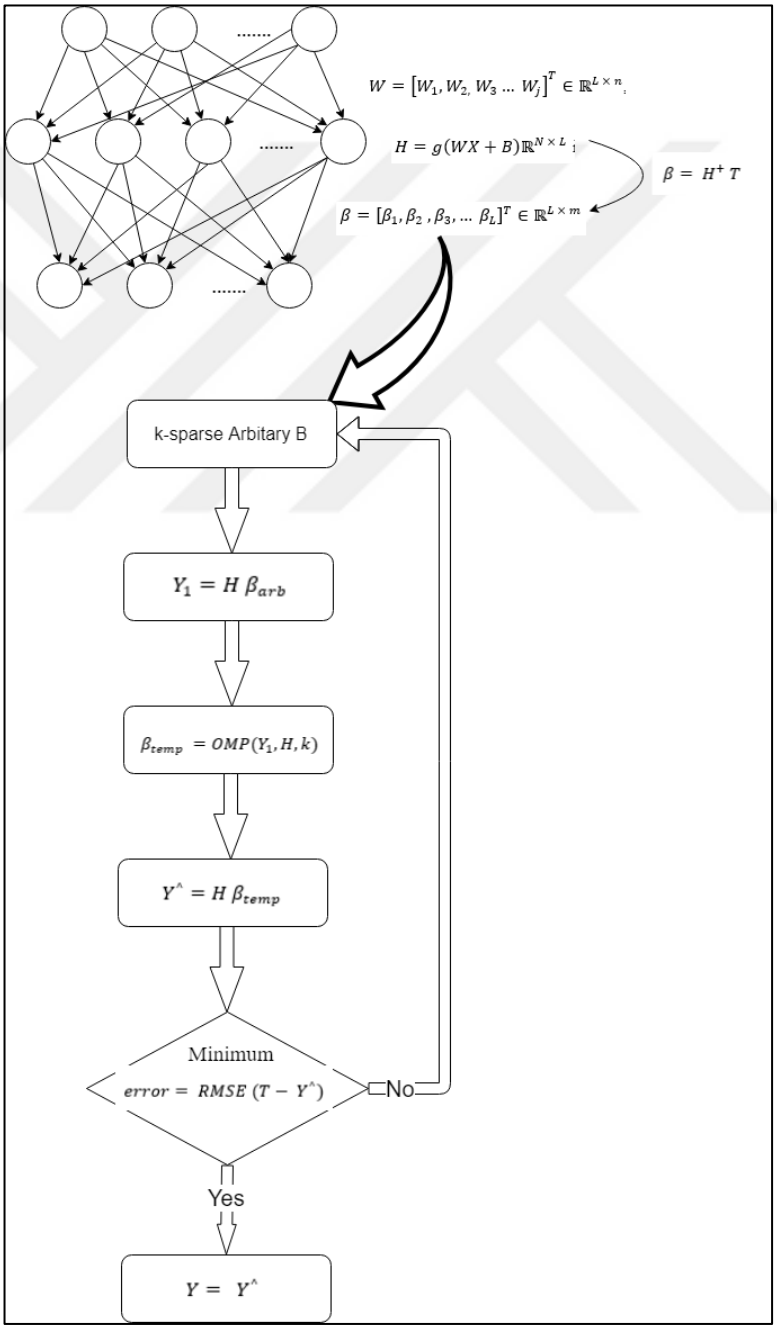

Figure 3.1 Greedy ELM

It is required for the greedy ELM to normalize all features of each dataset to zero mean and unit variance.

For $H = g(WX + B)$, the sigmodal activation function is used. To determine the k sparsity level, k is iterated from 1 to N the number of hidden neurons. Thus, k sparsity level shows the necessary number of hidden neurons.

3.1 Comparison With Other ELMs

This section declares the adopted comparison methodology. The comparison factor is testing accuracy which is the percentage of the correctly classified inputs. There are always a training data set and testing data set for each used dataset. Datasets are divided into training and testing parts when one big dataset used.

Other comparison factors could be used as well such as training time/accuracy to compare performance of the application algorithms.

Three algorithms results are taken into consideration in evaluation, 1) the Greedy single hidden layer feed forward ELM, 2) the standard ELM, 3) two hidden layer ELM and 4) multi hidden layers ELM.

CHAPTER 4

EXPERIMENT AND RESULTS

In practical experiment, three forwarded mentioned datasets are used to compare the four algorithms in respect to the testing accuracy. The experiment has been conducted on a Core i7-4500U Intel CPU @ 1.80 GHz and 8 GB DDR3 RAM hardware. MATLAB R2016b is used to run the algorithms and extract results. The experiment is focused on the classification problems, although some more enhancement and adjustments for the written algorithms would make them suitable for regression problems. The standard ELM algorithm can solve regression problems as it is grabbed from Nanyang Technological University website (Extreme Learning Machines, 2013). The inner fragments of this section are the following, a brief of coding efforts to implement the four experiment algorithms, a detailed description for the three considered datasets and detailed results and discussion with highlights on the testing accuracy changes.

4.1 Coding Efforts

The standard ELM MATLAB code was developed by Nanyang Technological University (Extreme Learning Machines, 2013). ELM code accepts the training and testing datasets, number of hidden neurons, a flag to specify if the problem is classification or regression and the activation function. The outputs are the training time and accuracy and testing time and accuracy. The ELM is iterated 20 times and the maximum testing accuracy is considered as best result.

The TELM algorithm is developed and written using the base ELM code and extended to align with TELM algorithm (Qu et al. 2015). The input parameters for TELM are the same as ELM one nut the output focused on the testing

accuracy as it is the comparison factor in our research. Orthogonal random weights and biases are initiated as suggested in (Qu et al. 2015)

Similarly, TELM is iterated 20 times as well and highest testing accuracy is considered.

The MELM algorithm is also developed based on the ELM code and focused on the second hidden layer of the two hidden layers ELM with a loop mechanism to extended to three hidden layers or more. Therefore, MELM is considered with 3 hidden layers or more. MELM input parameters are same as before with extra parameters to determine the number of hidden layers. Thus, this input parameter is 3 or more. Similarly, the output is focused on testing accuracy and iterated 20 times with highest accuracy consideration.

Lastly, the greedy ELM algorithm coding efforts have been applied on the basic ELM by initiating arbitrary sparse output weights and using them to calculate the classification results. Then using these results and the OMP algorithm to produce a sparse output weight vector which grades a new result to be compared with the target result and store the root mean square error. Further details about the greedy single hidden layer feedforward algorithm and detailed steps of greedy approach are mentioned in (Ince et al. 2014). The greedy ELM is iterated 20 times only and highest accuracy in considered. Additionally, the sparsity and RMSE values are extracted and fed into the results.

4.2 Datasets Used in Experiment

In experiment, three datasets are used to compare the 4 different algorithms. Diabetes dataset which was used earlier in (Alchikh Alnajar and Kayhan, 2018). Additionally, two datasets from the machine learning repository and center for machine learning and intelligent systems are used. The abalone and EEG eye state classification datasets. In the following sub-section, a detailed description for each data set and its attributes and outputs are discussed.

For the 3 used datasets, it is shuffled randomly $\frac{2}{3}$ of the total data records were used for the training and $\frac{1}{3}$ is used as testing data. After applying this approach, same training & testing data were used as input to the four algorithms.

4.2.1 Diabetes Dataset

The diabetes data set was obtained from the Nanyang Technological University website, which is the same data set used to test the basic single hidden layer ELM. It consists of nine columns, the first column represents the target class of classification problem, and other eight columns represent the input attributes of the diabetes data set. The input attributes were normalized to 0, 1 scale and the target classed were either logical 0 or 1. The logical 1 could refer to high possibility of having diabetes by aging. The input attributes could be values of features extracted from the patient samples; where each row reflect the attributes and target class of that patient. For instance, the first attribute could represent the ability to have diabetes because of family (one or both parents have diabetes). Table 4.1 gives a sample of the diabetes data set target class and attributes.

The data set package consists of 2 parts, the first is the training data set and the second is the testing data sets. Training data set is applied to each algorithm, then the testing accuracy is calculated using the testing data set. The diabetes training data set has 576 records and testing data set has 192 records.

Table 4.1 Diabetes data Sample

	Data set target class and attributes								
	Target class	**Attr 1**	**Attr 2**	**Attr 3**	**Attr 4**	**Attr 5**	**Attr 6**	**Attr 7**	**Attr 8**
	0	0.118	0.435	0.000	0.230	0.000	0.431	0.297	0.067
	1	0.647	0.675	0.000	0.000	0.000	0.779	0.213	0.317

4.2.2 Abalone Dataset

The abalone dataset from machine learning repository (Waugh, 1995). It consists of 4177 instances and divided in to a training part of 2784 instance and a testing part of 1393 instances. The aim is to predict the abalone age based on its physical measurements. Normally, abalone age is determined by opening its shell and counting the number of rings though a microscope which is a boring and time-consuming task. Eight attributes can be used to determine the abalone age and these are:

1) Sex, which is nominal value of being a male/female/infant. This attribute is normalized to integer of +1/-1/0 values
2) Length which is a continues real number in millimeters that represents the longest shell measurement
3) Diameter which is a continues real number in millimeters that represents the perpendicular to length
4) Height which is a continues real number in millimeters that represents its height with meat in the shell
5) Whole weight which is a real number in grams for total abalone weight
6) Shucked weight which is a real number in grams for total weight of the meat only
7) Viscera weight which is a real number in grams for total weight of the gut weight
8) Shell weight which is a real number in grams for total weight of the shell after being dried

The target class is the abalone age which ranged within the data set between 1 and 29 years old. For simplicity purposes, this age range was normalized to check if an abalone is over 15 years old or not. Thus, the target classes are only two now, 1 represent that abalone is over 15 years old and 0 represent abalone less than 15 years old.

Table 4.2 shows a sample data of 3 instances for the abalone dataset with its target class. The first instance is for a male abalone, with length 0.455 and shell weight 0.15.

Table 4.2 Abalone data sample

Data set target class and attributes								
Target class	**Attr 1**	**Attr 2**	**Attr 3**	**Attr 4**	**Attr 5**	**Attr 6**	**Attr 7**	**Attr 8**
1	1	0.455	0.365	0.095	0.514	0.2245	0.101	0.15
0	1	0.35	0.265	0.09	0.2255	0.0995	0.0485	0.07
0	-1	0.53	0.42	0.135	0.677	0.2565	0.1415	0.21

4.2.3 EEG Eye State Dataset

Similarly, EEG eye state dataset is grabbed from machine learning repository (Roesler, 2013). It consists of 14980 instances and divided in to a training part of 8997 instance and a testing part of 5983 instances. The aim of this dataset is to classify the eye state

if it is open or closed based EEG measurement using Emotiv EEG Neuroheadset. Each measurement duration was 117 seconds. All values are in chronological order with the first measured value at the top of the data. It is declared by the dataset donors that there are no missing values.

Attributes are all integer and real values and dataset is associated with a classification task of 2 target classes. In other words, 14 attributes of EEG values indicate the eye state.

Table 4.3 displays two instances of the EEG eye state data. The 14 values of EEG are listed to be used to extract the target class of 0 when eye closed and 1 when eye open.

Table 4.3 EEG Eye State data sample

Target Class	**Att1**	**Att2**	**Att3**	**Att4**	**Att5**	**Att6**	**Att7**
	Att8	**Att9**	**Att10**	**Att11**	**Att12**	**Att13**	**Att14**
0	4329.2	4009.2	4289.2	4148.2	4350.3	4586.1	4096.9
	4641	4222.1	4238.5	4211.3	4280.5	4635.9	4393.9
1	4335.9	4058.5	4279.5	4125.6	4350.3	4632.3	4103.1
	4634.9	4242.6	4260	4244.1	4307.7	4650.3	4397.9

4.3 Results

Results presenting methodology is included in this section and comparison in accuracy and time performance between developed algorithms. Detailed results for each algorithm using each data set are presented, then accuracy comparison to highlight the improvements and changes in various approaches. Lastly the time performance results are listed and discussed.

4.3.1 Diabetes Dataset Results

Considering 8 attributes in diabetes dataset, the number neurons in hidden layer is diverse starting at 4 neurons to 8, then 10 and 20 are used to track changes for high number of hidden neurons.

Results are shown in Table 4.4, it is noticed that the 3 hidden layers ELM has given the best results when the number of hidden neurons approaches the input attributes number and exceeds. While the greedy ELM with single hidden layer accuracy did reach very high testing accuracy and ranged around 68% and went down with higher number of hidden neurons.

Table 4.4 Diabetes dataset results

Algorithm	**Number of neurons in a hidden layer**						
	4	**5**	**6**	**7**	**8**	**10**	**20**
ELM	74.61%	76.17%	76.56%	76.95%	77.15%	76.76%	81.06%
TELM	77.34%	80.08%	75.39%	78.13%	74.22%	73.05%	63.67%
MELM (3)	79.30%	78.91%	78.13%	78.13%	78.52%	81.64%	80.86%
MELM (4)	72.66%	71.09%	76.56%	72.27%	68.36%	68.36%	67.19%
MELM (5)	68.36%	70.70%	71.48%	68.75%	68.75%	68.36%	64.84%
Greedy ELM	68.75%	68.75%	68.75%	68.75%	68.75%	68.75%	46.48%

Figure 4.1 show the changes of testing accuracy for all ELMs in respect to 4-12 number neurons in the hidden layer(s). It is perceived that the change in hidden neurons doesn't have major on testing accuracy in the greedy ELM, while it plays some role for other ELMs specially the multi hidden layers ELM.

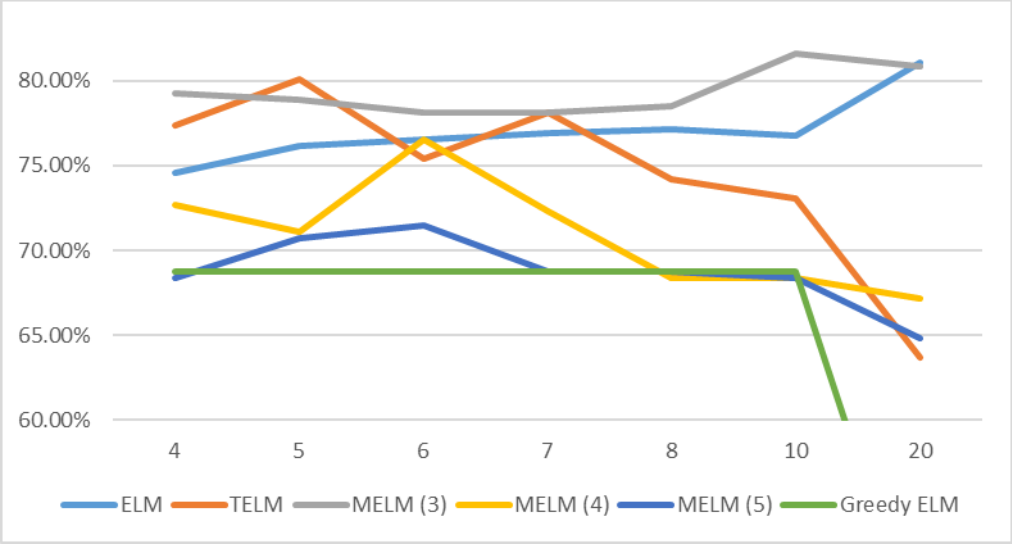

Figure 4.1 Diabetes results chart

Since the greedy algorithm has much more detailed outputs, these outputs are presented in details. As mentioned before the greedy algorithm is iterated 20 times and the maximum accuracy is considered. The following detailed outputs are a sample of the greedy algorithm output when 4 neurons in hidden layer are used.

The accuracy values for the 20 iterations are the following, it is noticed that the 1st iteration has given the best testing accuracy.

Table 4.5 Diabetes testing accuracies

Iteration#	1	2	3	4	5	6	7	8	9	10
Testing Accuracy %	68.75%	31.25%	41.80%	31.25%	31.25%	31.25%	31.25%	44.92%	68.75%	31.25%
Iteration#	11	12	13	14	15	16	17	18	19	20
Testing Accuracy %	31.25%	31.25%	31.25%	31.25%	31.25%	32.81%	31.25%	68.75%	31.25%	36.33%

The sparsity level values for all iterations are the following, the 1st iteration's sparsity level is 1 as well which refers to 1 none-zero values in the output weight vector.

Table 4.6 Diabetes sparsity levels

Iteration#	1	2	3	4	5	6	7	8	9	10
Sparsity	1	4	2	3	3	1	1	3	1	1
Iteration#	11	12	13	14	15	16	17	18	19	20
Sparsity	3	1	1	1	4	2	2	4	1	4

The RMSE values are the following for all iterations. The 1st iteration and sparsity level 1 cell shows the root mean square error values which has given the 68.75% accuracy.

Table 4.7 Diabetes RMSE values for different sparcisty level

Iteration#	**RMSE values**			
	Sparsity 1	**Sparsity 2**	**Sparsity 3**	**Sparsity 4**
1	1.0891	1.3204	1.1561	1.1787
2	1.6314	1.3298	2.0743	0.9583
3	0.9946	0.9811	0.999	1.0531
4	1.2857	1.1605	0.9655	1.0284
5	1.0819	1.2532	1.0705	2.9167
6	0.9872	1.3191	1.0378	1.0133
7	0.9564	1.7941	1.4105	1.3237
8	1.0149	1.22	0.9931	1.1683
9	1.0343	1.3976	1.337	2.9797
10	1.0313	1.863	1.1098	1.609
11	1.1206	1.3667	0.9812	2.8595
12	1.1336	1.4367	2.331	1.3903
13	0.9678	1.0383	1.5513	1.433
14	0.972	1.2814	1.1791	1.4794
15	1.228	1.1697	1.2591	0.9911
16	1.0859	0.9827	1.11	0.9864
17	1.1482	0.9406	1.1717	0.9549
18	1.4299	2.0687	2.0283	1.2932
19	0.9608	1.0076	1.2088	1.6115
20	1.2557	1.5063	1.0084	0.974

The output weights vector values for all iterations are the following, 1st iteration has 1 sparsity level, where third value is none-zeros as in **Table 4.8**

Table 4.8 Diabetes output weights for all iterations

Iteration#	Output weights			
1	0	0	0.4676	0
2	-0.0376	0.2596	-0.7373	-0.3535
3	0	1.133	0	-0.5251
4	-0.6642	0	-0.8623	0.7338
5	0	0.0377	0.0937	-1.214
6	-0.8851	0	0	0
7	0	-0.2932	0	0
8	0	-0.0821	1.7006	-1.3749
9	0.1349	0	0	0
10	0	0	0	-1.2774
11	0.506	0.0807	-1.245	0
12	-1.1453	0	0	0
13	0	0	-0.2881	0
14	-1.2326	0	0	0
15	-0.4953	-0.2611	0.4787	-0.236
16	-0.848	0	0.7438	0
17	0	-0.4568	0	-0.224
18	0.5662	0.4227	1.0255	-1.2261
19	-0.5874	0	0	0
20	0.0118	0.5663	-0.5324	-0.2089

4.3.2 Abalone Dataset Results

Considering 8 attributes in abalone dataset as well, this time the number neurons in hidden layer is diverse starting at 4 neurons to 12 to check different scenarios rather than what was used in the diabetes dataset.

Results are shown in Table 4.9, it is noticed that the better results than what diabetes results were. Among the none sparse ELMs, the single hidden layers ELM gave best testing accuracy with value reached 91.99%. The testing accuracy when the number of hidden neurons approaches the number of input attributes is also high with less than 0.05% difference from the highest accuracy. Additionally, all standard ELMs have given a close testing accuracy.

While, the greedy ELM has reached maximum 91.31% for various number of hidden neurons and preserved very low accuracy for other number of hidden neurons, which raises a flag about its performance.

Table 4.9 Abalone dataset results

Algorithm	**Number of neurons in a hidden layer**								
	4	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
ELM	91.70%	91.81%	91.99%	91.92%	91.92%	91.95%	92.03%	91.99%	91.99%
TELM	91.31%	91.31%	91.31%	91.31%	91.31%	91.39%	91.31%	91.39%	91.31%
MELM (3)	91.39%	91.67%	91.82%	91.74%	91.03%	69.49%	67.62%	67.19%	70.28%
MELM (4)	91.31%	91.24%	91.24%	91.24%	91.24%	91.24%	91.24%	91.17%	91.24%
MELM (5)	91.24%	91.31%	91.31%	91.24%	91.24%	91.24%	91.17%	91.17%	91.24%
Greedy ELM	91.31%	91.31%	35.75%	33.67%	63.68%	25.84%	35.82%	13.28%	45.94%

Figure 4.2 chart shows the changes in testing accuracy in respect to the number of neurons in the hidden layers for all studied algorithms.

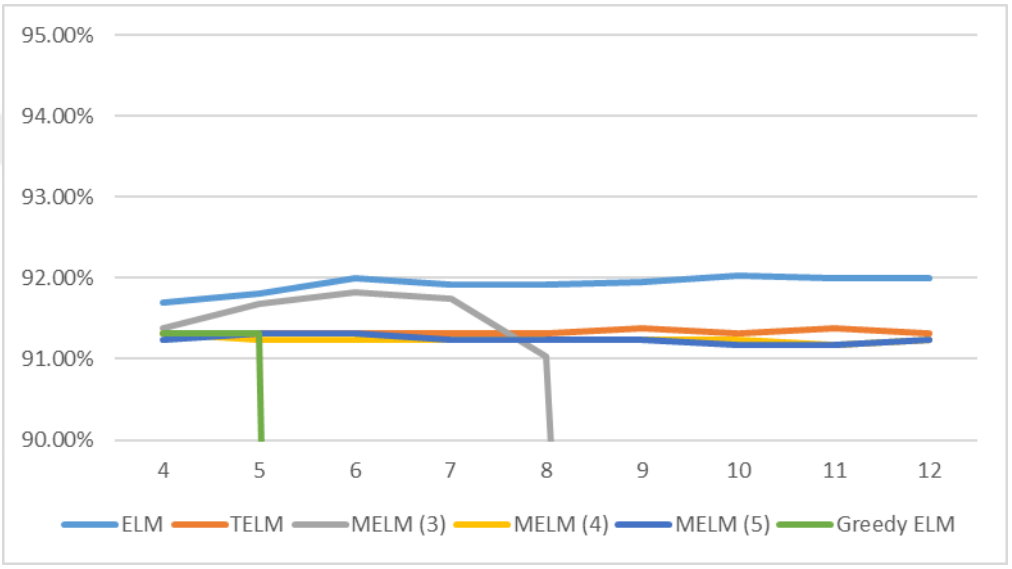

Figure 4.2 Abalone results chart

For the abalone dataset, the greedy ELM is also iterated 20 times and highest testing accuracy is considered. The following outputs are taken when 5 hidden neurons are used.

The 9th iteration has given 91.31% while the rest 19 iterations have given very low accuracy, although the 9th iteration one is considered high and close to other standard ELMs.

Table 4.10 Abalone testing accuracies

Iteration#	1	2	3	4	5	6	7	8	9	10
Testing Accuracy %	8.69%	8.69%	8.69%	28.36%	16.51%	8.69%	8.69%	8.69%	91.31%	8.69%
Iteration#	11	12	13	14	15	16	17	18	19	20
Testing Accuracy %	8.83%	8.69%	8.69%	20.67%	10.62%	8.69%	8.69%	8.69%	8.69%	8.76%

The sparsity level in the 9th iteration is a as the following table shows. Thus, 1 none-zero values are expected in the output weights vector.

Table 4.11 Abalone sparsity levels

Iteration#	1	2	3	4	5	6	7	8	9	10
Sparsity	4	3	3	3	3	4	2	3	1	5
Iteration#	11	12	13	14	15	16	17	18	19	20
Sparsity	4	1	1	2	5	3	3	1	5	5

Since the focus in these results correspond to 5 neurons in hidden layers, the maximum sparsity level is 5. It is noticed that the 9th iteration has given highest accuracy with sparsity level 1, the following table show that the RMSE values corresponding to all iterations and 9th one is highlighted and 1 sparsity level is the minimum value among all RMES values within the same iteration. Referencing the greedy ELM, 1 hidden neuron is the necessary out of 5.

Table 4.12 Abalone RMSE values for different sparsity levels

	RMSE values				
Iteration#	**Sparsity 1**	**Sparsity 2**	**Sparsity 3**	**Sparsity 4**	**Sparsity 5**
1	1.1757	1.1988	2.0139	0.6371	1.3491
2	0.9964	0.9936	0.6414	0.673	0.9262
3	1.274	1.3435	0.7025	1.5227	1.6897
4	2.1358	1.5061	1.0469	1.2064	1.9754
5	0.9234	1.8766	0.7147	1.5109	2.0388
6	1.4213	1.6623	1.4068	0.642	2.3426
7	1.6691	1.073	1.5104	2.3891	1.4996
8	1.502	0.9737	0.6533	1.6704	2.0462
9	1.167	2.7153	1.2199	2.464	1.3468
10	2.0357	1.5047	2.3719	2.2491	0.7415
11	1.5347	0.6426	2.1509	0.6118	1.8867
12	0.6293	1.4698	2.5959	0.8808	1.0195
13	0.737	1.1119	2.7038	1.5598	3.7593
14	1.1237	0.9495	1.7468	1.5951	1.5966
15	0.8006	0.9119	0.9319	0.7921	0.7765
16	0.9922	1.5416	0.6701	1.7628	0.8797
17	0.8395	0.8914	0.7374	2.1521	1.8405
18	0.7016	1.0875	2.261	1.471	2.4068
19	1.4746	0.9778	0.9888	2.9366	0.7864
20	0.8662	1.1677	1.5648	3.27	0.8079

A cross checking is applied by presenting the output weights vectors for all iterations and confirm that the used output weights vector has 1 none-zero values as in below Table 4.13

Table 4.13 Abalone output weights for all iterations

Iteration#	Output weights				
1	-0.609	-0.6105	0.7569	-0.4113	0
2	0	0	0.5131	-1.5972	-0.153
3	0	-0.1106	0	-0.0676	-0.8473
4	0	1.5878	-0.3063	-1.5656	0
5	-0.2538	0	0	0.7611	-0.8637
6	0.1099	-0.2305	-0.2738	0	-0.9119
7	0	0	-1.3989	0	-0.7729
8	0	0.3388	-0.7455	-0.6512	0
9	0	0	0	0.5488	0
10	-1.025	0.5197	-0.3382	0.0539	-0.9924
11	-1.3895	-1.1564	0	0.0468	0.8475
12	0	0	0	-1.2207	0
13	0	-0.6338	0	0	0
14	-0.3077	0.1827	0	0	0
15	-0.9655	0.567	0.0377	0.1353	-2.0912
16	-1.1398	0.681	0	-1.0536	0
17	0	-0.7763	0.2785	0	-0.6735
18	0	0	-1.4286	0	0
19	-0.0477	-0.144	0.2977	-1.2777	-0.3006
20	-0.4961	-0.7094	0.3129	0.746	-0.1997

4.3.3 EEG Eye State Dataset Results

Considering 14 attributes in EEG Eye State dataset (denoted as EEG dataset), this time the number neurons in hidden layer is diverse starting at 8 neurons to 16 to check different scenarios such as previous datasets.

Table 4.14 demonstrates the application if four algorithms using the EEG dataset, no major changes in compare with the results of previous datasets. It is only noticed that the single hidden layer ELM has gave highest accuracy among none-sparse ELMs and the greedy preserved the highest records by reaching 62.82% testing accuracy. Surprisingly, the standard ELMs have given very similar results and the greedy algorithm highest testing accuracy against standard ELMs for the same number of hidden neurons which is 14.

Table 4.14 EEG dataset results

	Number of neurons in a hidden layer									
	8	9	10	11	12	13	14	15	16	20
ELM	55.25%	55.24%	55.24%	56.12%	55.77%	57.62%	55.25%	55.51%	61.82%	55.22%
TELM	54.91%	54.93%	54.91%	55.93%	54.89%	54.91%	56.29%	54.89%	54.91%	54.95%
MELM (3)	55.21%	54.91%	54.91%	55.07%	54.91%	54.91%	54.91%	54.91%	55.11%	57.33%
MELM (4)	54.91%	59.59%	54.91%	54.91%	54.95%	54.97%	57.21%	54.91%	54.91%	56.21%
MELM (5)	54.91%	54.91%	55.03%	55.05%	54.91%	54.91%	54.99%	55.07%	56.15%	56.15%
Greedy ELM	54.89%	54.89%	54.89%	58.57%	55.19%	54.87%	57.01%	54.61%	54.89%	55.35%

Figure 4.3 observes the differences between accuracy values for the studied algorithms in respect to number of hidden neurons. No major effect between the number of input attributes and the number of hidden neurons on the testing accuracy.

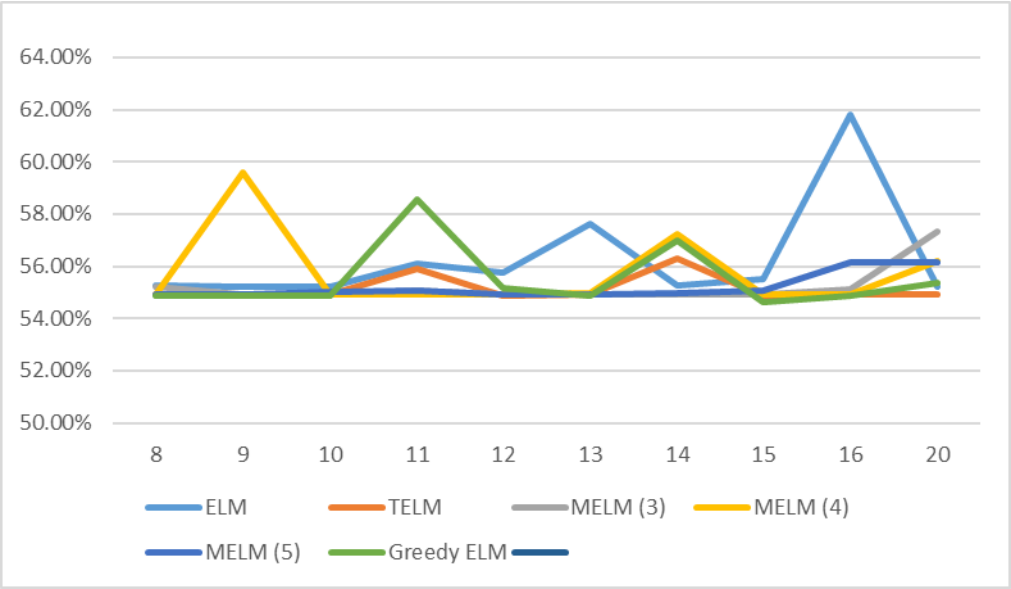

Figure 4.3 EEG results chart

Focusing on the greedy ELM when 14 neurons in the hidden layer(s) is used and for 20 times iteration, the following testing accuracies were produced:

Table 4.15 EEG testing accuracies

Iteration#	1	2	3	4	5	6	7	8	9	10
Testing Accuracy %	45.13%	45.11%	47.92%	46.42%	45.11%	57.01%	50.30%	43.57%	44.59%	45.35%
Iteration#	11	12	13	14	15	16	17	18	19	20
Testing Accuracy %	52.02%	45.11%	45.11%	42.85%	44.37%	46.72%	45.41%	45.15%	48.96%	55.97%

The 6th iteration has given the highest accuracy of 57.01% with sparsity level of 13 as shown below. Thus, the number of necessary coefficients in the output weights vector is 13.

Table 4.16 EEG sparsity levels

Iteration#	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Sparsity	4	1	4	4	3	13	8	5	5	5
Iteration#	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Sparsity	6	1	2	10	14	8	8	8	13	11

Similarly, the maximum sparsity level is 14 since the results are shown for 14 used hidden neurons. Thus, for the 6th iteration and sparsity level 13 the minimum RMES value (within iteration #6) is highlighted in the following table. According to greedy ELM, 13 necessary hidden neurons are needed out of 14.

Table 4.17 EEG RMSE values for different sparsity levels

	RMSE values													
	Sparsity													
Iteration#	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	1.50	1.17	1.44	1.00	1.17	1.19	1.09	1.00	1.32	1.26	1.44	1.47	3.09	1.22
2	1.09	2.46	1.31	4.30	1.53	1.72	1.28	5.13	1.12	1.14	1.41	1.71	2.36	1.12
3	2.42	1.08	1.09	1.00	1.12	1.37	3.24	2.34	1.92	2.61	1.05	4.86	1.15	4.19
4	1.21	2.21	1.48	1.00	2.67	1.14	2.64	1.43	2.42	4.10	1.46	2.69	1.25	2.47
5	1.25	1.64	1.01	1.27	1.98	1.84	2.20	4.59	1.28	4.26	1.67	1.29	1.60	3.09
6	1.13	1.45	1.36	1.37	2.44	1.31	1.73	4.20	1.08	1.07	3.14	1.97	1.03	1.21
7	1.06	2.85	1.27	1.01	1.57	1.01	2.40	1.01	1.07	1.59	1.16	1.37	2.57	3.70
8	1.04	1.06	1.75	1.00	0.99	1.35	1.04	1.97	1.71	1.00	1.49	1.21	1.24	2.84
9	1.17	1.69	1.28	2.12	0.99	1.62	1.84	1.15	4.45	1.23	1.15	1.25	2.02	1.03
10	1.11	1.10	2.11	1.40	1.03	1.59	1.25	1.33	1.75	3.82	1.51	1.09	1.54	1.19
11	1.33	1.03	1.11	1.57	1.67	1.00	1.58	1.06	1.08	1.07	1.04	2.08	2.69	2.06
12	1.00	1.27	1.15	1.31	1.20	1.07	1.02	1.01	1.24	1.01	1.47	1.83	1.02	3.40
13	1.02	1.02	1.95	2.27	1.84	2.41	2.16	4.79	1.25	1.15	1.12	4.32	1.11	1.03
14	1.22	1.01	1.02	1.70	2.52	1.70	1.07	1.18	1.01	1.00	1.76	3.57	1.64	1.71
15	1.00	1.02	1.12	0.99	1.15	1.84	2.19	1.04	1.67	1.30	1.06	1.24	3.10	0.99
16	1.22	1.22	1.72	1.74	1.27	1.59	2.96	0.99	1.00	1.46	1.85	1.39	1.81	2.66
17	1.19	1.19	2.13	2.65	1.09	2.84	2.53	1.02	2.18	2.46	5.64	3.42	2.10	2.41
18	1.49	1.03	1.19	1.49	1.01	5.69	2.01	1.00	2.74	2.01	3.12	1.05	3.20	5.14
19	1.14	1.30	1.31	3.43	2.90	1.11	1.13	2.28	2.33	1.92	3.19	2.37	1.00	1.55
20	2.26	1.08	3.17	1.49	2.34	1.13	1.36	1.81	2.04	2.96	1.01	4.00	5.12	2.06

It is directly concluded that the sparse output weights vector would have 13 none-zero values in the 6th iteration.

Table 4.18 EEG output weights for all iterations

Iteration#	Output weights												
1	0.00	0.00	0.00	0.78	0.00	0.00	-0.26	0.00	0.21	0.00	0.00	0.00	0.12
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.59	-0.24	0.12	0.00	0.00	0.00	0.00	0.00	-2.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.52	-0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.26	-0.68
5	0.00	0.00	0.00	0.00	0.00	0.00	-1.69	0.00	0.33	0.00	0.00	0.00	-0.10
6	2.36	-0.37	-0.25	3.20	0.48	0.00	0.77	-1.91	-0.40	0.54	-2.01	-1.36	-0.84
7	0.00	0.55	0.00	1.01	0.09	0.00	0.00	0.00	0.85	-0.86	-1.81	-0.14	0.17
8	0.00	-2.34	0.00	0.00	0.00	0.00	0.00	1.08	-0.05	0.00	0.63	0.00	0.00
9	0.11	0.00	2.09	0.27	-0.40	0.00	-2.21	0.00	0.00	0.00	0.00	0.00	0.00
10	0.00	1.53	0.00	0.00	0.00	0.00	0.00	-1.68	0.00	-0.90	-0.18	0.07	0.00
11	0.00	1.27	0.00	0.19	-0.60	0.00	0.00	0.00	0.00	0.00	-1.30	0.59	0.00
12	0.00	0.00	0.00	-1.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
13	0.00	0.00	-0.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.37	0.00
14	0.16	0.00	-0.55	-0.46	1.59	0.32	-0.64	0.00	-0.03	-0.31	0.00	-1.44	0.00
15	0.24	1.07	-1.03	-1.34	0.40	0.40	-0.30	-0.78	-0.69	-0.12	-1.11	-0.45	-0.62
16	0.00	-0.08	0.88	-1.63	1.01	0.00	0.00	0.38	1.23	-0.66	0.00	0.00	0.00
17	0.00	0.45	-0.26	0.00	1.21	0.00	0.75	-0.17	-0.07	0.00	0.00	-0.75	0.00
18	0.44	0.00	-0.25	0.00	0.29	0.00	-0.24	0.00	0.00	-1.88	0.27	1.48	0.00
19	0.94	0.08	0.00	-1.92	1.19	0.63	-0.78	1.74	1.28	-0.72	-0.14	0.14	-2.17
20	-0.81	-2.10	-1.63	1.24	0.00	-0.28	1.74	-0.25	-0.75	-1.00	0.00	-0.86	0.92

4.3.4 Results Comparison

Among Table 4.4, Table 4.9 and Table 4.14 it has been generally observed the multi hidden layers Elm giving best accuracy when compared to single, two hidden layer ELMs or greedy algorithms. But when it comes to greedy ELM algorithm it is not always (in respect to the 3 studied datasets) reaching good accuracy. The only case when Greedy ELM has passed the standard ELMs is using the EEG dataset and when the number of hidden neurons is same as number of input features. The testing accuracy values had ranged between a minimum value of ~55% to ~91% with and without sparse methods interfering in the extreme learning machine basic mathematical approach. Injecting sparsity approach with single hidden layer ELM has produced new interesting results but no notable improvements to the testing accuracy factor.

Time efficiency would be another comparison factor which is noticed during the experiment specially when large number of input attributes in the experiment datasets, such as the case of EEG dataset and these are presented in following sections.

The results from any of the RMSE tables presented in each dataset results application shows the lowest RMSE value has given the highest testing accuracy. Although multiple RMSE values could lead to same testing accuracy. For instance, in Table 4.14 there are multiple high testing accuracy values (in compare with other standard ELM accuracies) for 11, 14 & 20 hidden neurons.

Table 4.19 represents the RMSE values correspond to highest accuracies (58.57%, 57.01% & 55.35%) which correspond to 3,11 & 16 sparsity levels in order and number of hidden neurons in greedy ELM for EEG dataset, where the first columns refer to the number of hidden neurons applied the dataset, and S column is the sparsity level that correspond to the highest testing accuracy value. Then a scale of sparsity is listed where maximum sparsity doesn't exceed the number of hidden neurons.
All highlighted RMSE values in Table 4.19 have led to the highest testing accuracy in application of greedy ELM using the EEG dataset. It is noticed that these RMSE values are almost the same.

Table 4.19 RMSE values for EEG dataset

#	S	Sparsity Scale/ RMSE																			
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
11	3	1.50	1.54	1.02	1.37	1.35	1.41	1.96	1.19	1.82	2.04	1.29									
14	11	2.26	1.08	3.17	1.49	2.34	1.13	1.36	1.81	2.04	2.96	1.01	4.00	5.12	2.06						
20	16	1.43	1.16	2.10	2.70	1.76	1.79	3.55	1.23	1.48	1.46	2.09	6.12	4.33	1.89	4.71	1.03	3.44	1.44	3.61	3.96

In the application of greedy ELM on the Abalone dataset, using 4 & 5 hidden neurons have given the highest accuracy of 91.31%. In Table 4.20, RMSE values correspond to highest accuracy and number of hidden neurons in greedy ELM for Abalone dataset and first and second column refer as previous table and the highlighted RMSE values are 1.1 to 1.17 have given same testing accuracy. It is noticed that the sparsity level/number of necessary hidden neurons correspond to the lowest RMSE.

Table 4.20 RMSE values for Abalone dataset

#	S	Sparsity Scale/ RMSE											
		1	2	3	4	5	6	7	8	9	10	11	12
4	1	1.10	1.36	1.69	1.10								
5	1	1.17	2.72	1.22	2.46	1.35							

4.3.5 Accuracy, Sparsity & RMSE Relation Results

Greedy ELM results have emphasized the relation between testing accuracy, sparsity level and RMSEs values, the following results highlight this relation. Considering the EEG dataset (which has 14 input attributes), 16 hidden neurons and 20 iterations for the greedy algorithm.

It is noticed that the highest accuracy with these 20 iteration is hit twice in the 2^{nd} and 17^{th} iterations as shown in below table.

Table 4.21 EEG testing accuracies for 16 hidden neurons

Iteration#	1	2	3	4	5	6	7	8	9	10
Testing Accuracy %	45.11%	54.89%	49.24%	45.11%	53.44%	53.02%	45.11%	44.79%	45.15%	45.11%
Iteration#	11	12	13	14	15	16	17	18	19	20
Testing Accuracy %	45.11%	45.11%	45.35%	45.17%	45.11%	45.11%	54.89%	42.37%	53.26%	45.13%

The corresponding sparsity values for the 2^{nd} and 17^{th} iterations are 3 and 1 in order.

Table 4.22 EEG sparsity levels for 16 hidden neurons

Iteration#	1	2	3	4	5	6	7	8	9	10
Sparsity	5	3	9	1	11	7	1	14	13	4
Iteration#	11	12	13	14	15	16	17	18	19	20
Sparsity	1	1	4	6	1	1	1	5	4	13

The RMSE values (in below table) which gives highest testing accuracy are very close to each other with a difference between 0.04 which is a very small range that in respect to other RMSE values' variations.

The highlighted RMSE values have given testing accuracy 91.31%.

Table 4.23 EEG RMSE values for 16 hidden neurons

RMSE values																
Sparsity																
#	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	1.08	1.36	1.00	1.03	0.99	1.46	1.55	1.57	2.53	1.47	1.02	2.17	1.97	1.06	1.46	1.19
2	1.24	1.06	1.02	2.05	1.41	1.11	1.50	1.31	1.21	2.05	2.66	1.86	1.21	1.06	3.90	2.42
3	1.08	1.54	1.68	1.28	1.31	2.60	2.56	2.69	1.02	1.38	2.66	1.54	3.10	1.72	2.70	2.85
4	1.00	1.03	1.48	1.12	1.14	2.27	1.27	1.87	1.46	1.38	1.88	1.89	1.87	5.66	3.21	3.12
5	1.15	1.19	1.57	1.95	1.23	1.08	1.78	1.37	1.39	1.92	1.02	2.76	1.61	2.42	1.78	1.43
6	1.06	1.02	1.25	1.23	1.02	2.62	1.00	1.20	1.54	3.70	1.25	2.09	1.25	1.36	1.70	1.20
7	0.99	1.06	1.68	1.92	1.42	4.52	2.23	1.69	1.81	1.48	1.01	1.63	1.62	1.04	3.65	1.79
8	1.04	1.02	1.84	1.01	2.61	1.12	3.31	1.02	4.87	1.64	2.01	2.06	1.13	1.00	1.35	3.43
9	1.61	2.78	1.99	1.87	1.93	1.90	1.07	3.50	1.94	2.51	3.25	2.13	1.05	2.09	1.23	1.09
10	1.93	1.45	1.97	0.99	2.30	2.79	1.01	1.38	2.66	3.24	1.40	1.04	1.06	3.40	3.58	2.87
11	1.00	1.59	1.01	1.24	1.23	1.42	1.06	2.25	2.03	1.94	2.91	2.20	3.10	2.72	1.12	1.07
12	1.00	1.65	1.02	1.69	1.01	2.20	1.10	2.22	1.04	1.05	1.12	2.66	1.94	1.19	2.34	1.49
13	0.99	1.01	1.26	0.99	1.28	3.65	1.11	1.36	1.00	2.84	4.45	1.15	2.19	1.57	1.98	3.14
14	1.02	1.13	1.12	1.06	1.12	1.01	1.02	1.13	1.02	1.17	1.34	2.05	4.65	2.97	1.04	2.60
15	1.00	1.40	2.43	1.50	1.86	1.29	1.75	3.48	2.56	2.87	1.34	1.14	1.46	1.24	2.46	2.59
16	1.00	1.52	1.18	1.16	1.10	1.02	1.62	1.48	3.35	4.23	1.36	2.42	1.29	3.00	3.63	1.28
17	1.00	1.17	1.67	1.64	1.14	1.74	1.09	1.76	3.20	1.77	1.00	1.71	5.32	1.70	1.23	1.53
18	1.64	1.01	1.27	1.13	0.99	1.02	2.75	1.09	1.31	1.08	1.08	1.39	2.91	0.99	2.66	1.29
19	1.07	1.06	1.09	1.05	1.44	1.45	1.78	1.26	4.62	2.01	1.94	3.95	4.04	4.02	3.64	7.00
20	1.77	1.04	1.66	1.80	2.84	2.96	1.31	1.62	2.52	1.97	3.75	3.19	1.03	2.21	3.76	4.09

4.3.6 Time Performance Results

All four ELMs algorithms have been applied on the 3 studied datasets and time calculation is measured for only the time needed to compute final output weight β and output of training phase of the algorithms. The normalization of input attributes, testing outputs and accuracy computation are not included in time performance calculation. All values of time in this section are measured in milli seconds using the tic/toc MATALB functions.

For diabetes dataset, time performance is computed for the 3 standard ELMs and greedy ELM using 4 – 8 hidden neurons and then 10 and 20 ones. Table 4.24 presents the time needed to get the highest accuracy values. It is seen that whenever the number of hidden neurons increases the more time it takes to calculate results. Additionally, comparing between different algorithms the greedy algorithms has taken the lead of being the slowest one with notable performance difference specially against the single hidden layer ELM.

Table 4.24 Time performance comparison for diabetes dataset

Algorithms	number of hidden neurons						
	4	5	6	7	8	10	20
ELM	0.57	0.63	0.77	1.13	0.78	0.93	1.32
TELM	2.15	2.75	1.92	1.99	2.21	2.70	9.43
MELM (3)	3.66	2.67	2.67	3.36	3.39	4.09	7.63
MELM (4)	6.83	4.37	4.34	4.78	5.25	6.30	16.03
MELM (5)	4.65	4.68	5.50	6.54	9.41	7.94	14.58
Greedy ELM	24.90	24.60	28.90	28.10	33.20	39.50	134.30

Illustrating the results from Table 4.24 by a line chart, Figure 4.4 points out the variation in time performance differences where the 3 standard ELMs have reached less than 20 MS for the highest studied number of hidden neurons (20) while the greedy ELM has exceeded this time with ~25 MS even using 4 hidden neurons.

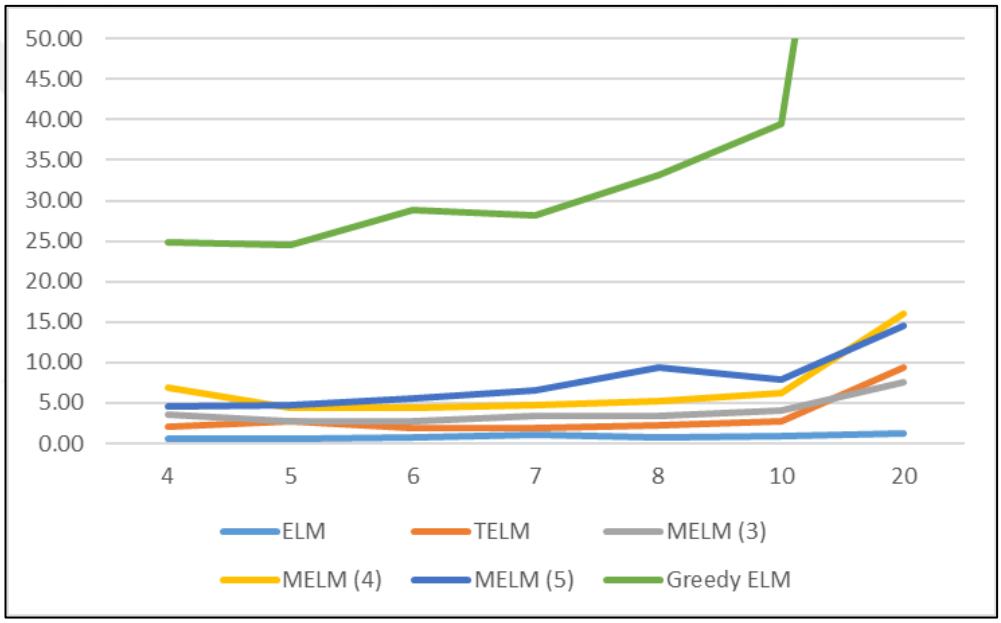

Figure 4.4 Time performance chart - diabetes dataset

Application of the algorithms on the Abalone dataset has also taken a place in the experiment. Similarly, like the application on Abalone dataset to measure the testing accuracy, 4 to 12 hidden neurons have been used to compute the time performance. Table 4.25 shows time performance which correspond to the highest testing accuracy

On the other hand, **Figure 4.5** emphasizes the differences between the 3 standard ELMs and the greedy ELM where the standards ones are very close to each other and the greedy ELM starts with more than 25 MS with the lowest number of hidden neurons and reaches 65% approximately for 12 hidden neurons. The multi hidden layers ELM with 5 hidden layers has preserved the lowest performance among standards ELMs.

Table 4.25 Time performance comparison for Abalone dataset

Algorithms	Number of hidden neurons								
	4	5	6	7	8	9	10	11	12
ELM	0.98	1.62	1.11	1.06	1.30	1.51	1.34	1.84	1.54
TELM	8.75	3.96	4.38	4.58	5.54	5.74	6.35	7.12	7.46
MELM (3)	17.40	6.30	7.21	7.76	8.99	10.77	11.70	12.31	12.36
MELM (4)	8.63	9.94	10.89	13.62	16.15	15.85	24.06	21.86	21.23
MELM (5)	11.08	13.30	14.58	18.33	19.84	21.65	24.08	32.50	32.69
Greedy ELM	31.90	24.50	26.20	43.00	38.80	46.10	47.70	63.00	64.90

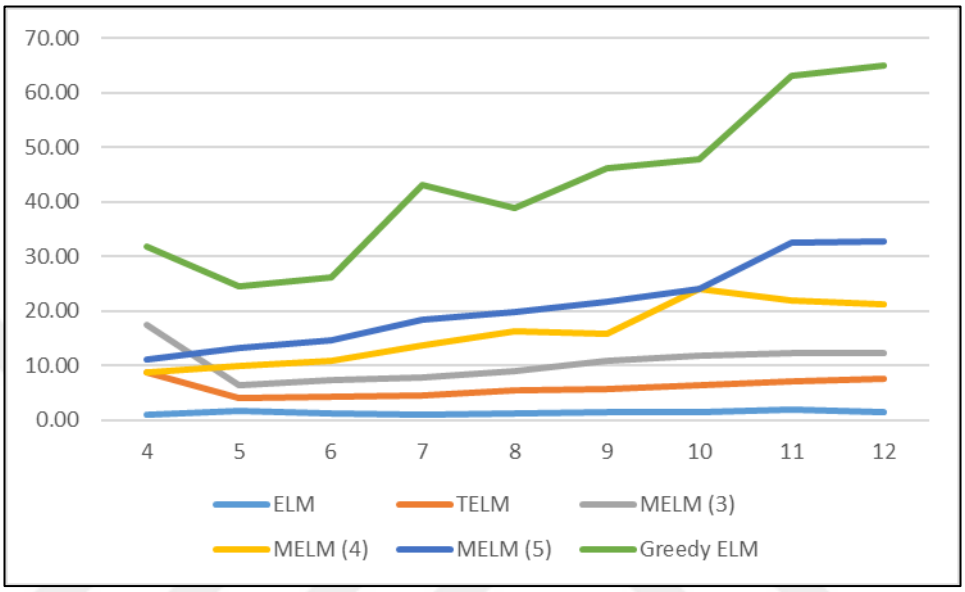

Figure 4.5 Time performance chart - Abalone dataset

Lastly, time performance results of the EEG eye state dataset application using the four algorithms have been aligned with previous dataset's results. Using 8 to 16 hidden neurons for the 14 inputs attribute EEG dataset, time performance has been computed and included in Table 4.26. The MELM with 5 hidden layers has preserved longer time performance comparing to standard ELMs. Likewise, greedy ELM time performance is much lower and longer to train the algorithm.

Table 4.26 Time performance comparison for EEG dataset

Algorithm	Number of hidden neurons								
	8	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
ELM	9.03	11.88	6.51	3.99	3.65	4.33	7.20	7.66	8.46
TELM	22.05	22.56	26.72	19.52	21.23	41.11	43.21	46.98	33.63
MELM (3)	42.10	43.31	47.84	53.13	59.77	70.61	77.84	92.55	95.05
MELM (4)	59.44	52.57	69.06	79.82	92.28	91.68	122.54	133.25	155.40
MELM (5)	78.09	85.04	100.99	95.76	118.90	144.25	167.55	183.30	236.92
Greedy ELM	90.50	100.00	96.50	86.30	144.50	228.70	236.60	256.00	612.50

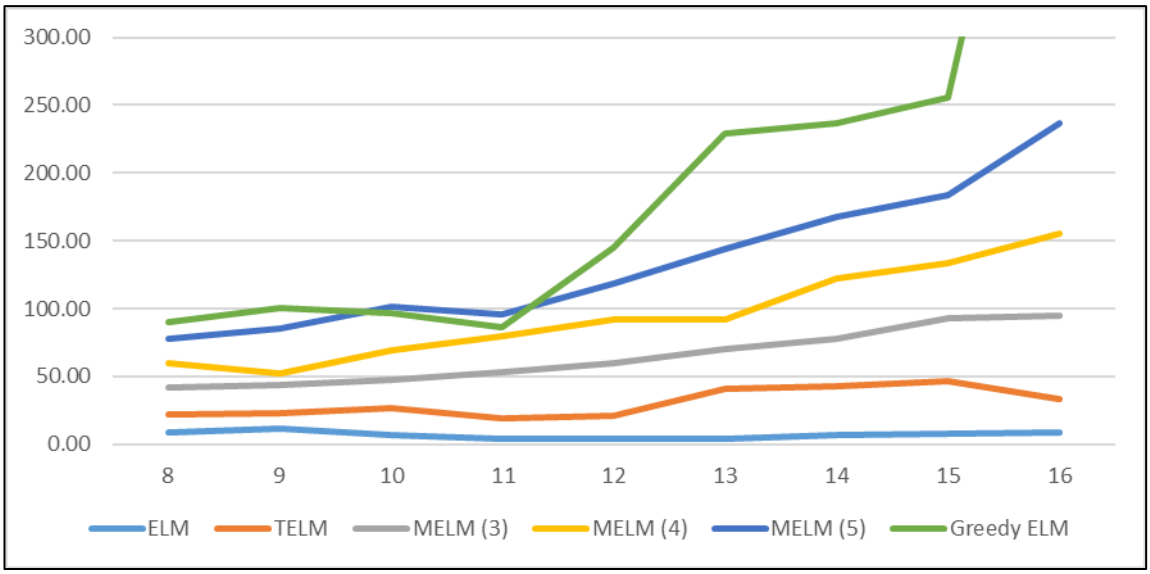

Figure 4.6 Time performance chart - EEG dataset

Figure 4.6 illustrates the results presented in Table 4.26 and shows the time performance differences for greedy ELM against other standards ELMs. The MELM time needed to produce the output weights -with 5 hidden layers performance with 16 hidden neurons in each hidden layer- is almost approaching the lowest time needed for the greedy ELM with 8 hidden neurons for the greedy ELM.

CHAPTER 5

CONCLUSION AND FUTURE RESEARCH

Results show generally better accuracy performance for the multi hidden layer ELM in compare with the other standard ELMs algorithms and greedy one. It can be concluded that the testing accuracy linked to sparse approaches doesn't always give a stronger relation rather than the Moore-Penrose (MP) inverse approaches applied by the single, two and multi hidden layers ELMs.

Since the maximum testing accuracy is considered in each iteration, it is not necessary that the maximum testing accuracy to be corresponded to a unique RMSE but it corresponds to a range of RMES. The relations between accuracy, sparsity and RMSE values – presented in section 4.3.5 – shows that there is arrange of RMSE values which has given same or very close accuracy for different sparsity levels.

The nature of the used datasets might have added influence on the experimental results since they are binary classification and sparsity approach didn't produce massive improvements for testing accuracy for the studied datasets which might refute the theory of a correlation between one or more input features and the targeted class. Sparsity level in greedy algorithm refers to the necessary number of hidden neurons within an ELM hidden layer.

5.1 Future Research Recommendations

Future research opportunities exist is various areas in this study. One of these is to penetrate sparse approaches in the two and multi hidden layers ELMs. In particular, sparse approach can used on the first hidden layer of TELM and/or MELM and then iteration one time for TELM and multiple iterations can be applied to achieve the output weights. One of the challenges that might appear during this process is that the weight between second hidden layer and the ones after should be a matrix, while the sparse approach using greedy algorithms returns a sparse vector.

Training time has been studied as well to focus on the time performance effectiveness on the ELM after penetrating sparse approaches. Results have pointed that the time performance for sparse greedy ELMs would be weaker than standard ELMs specially when large number of input attributes is considered. The reason behind this weaker performance might be associated with the iteration applied for greedy algorithms.

Figure 5.1 shows the potential flow of how sparse approaches, specifically the greedy algorithms, could be integrated with multi hidden layers ELM for the outputs of the weights between first hidden layer and second hidden layer, then MELM algorithms can iterate for the number of hidden layers for various number of hidden neurons. The target output of this process would be the output weights which corresponding to the lowest root mean square error.

After calculating the output weights, it would be used to estimate the output of the training. Training accuracy then can be measured and it is recommended to evaluate the time needed for the algorithm training as it can be used for further assessment of the algorithm.

Lastly, the output weights should have been saved a side during the training process and it can be used directly to compute the estimated outputs. Testing accuracy would be driven from the actual target outputs and the estimated outputs. A percentage of the correctly estimated output from the total target ones can be calculated easily. The formed figures of time performance during the training and the testing accuracy would be used to compere results with standard ELMs and the greedy single hidden layer feed forward ELM.

Other potential factors can be extracted for the developed algorithm such as training accuracy, training time, testing accuracy, testing time and the root mean squared error values.

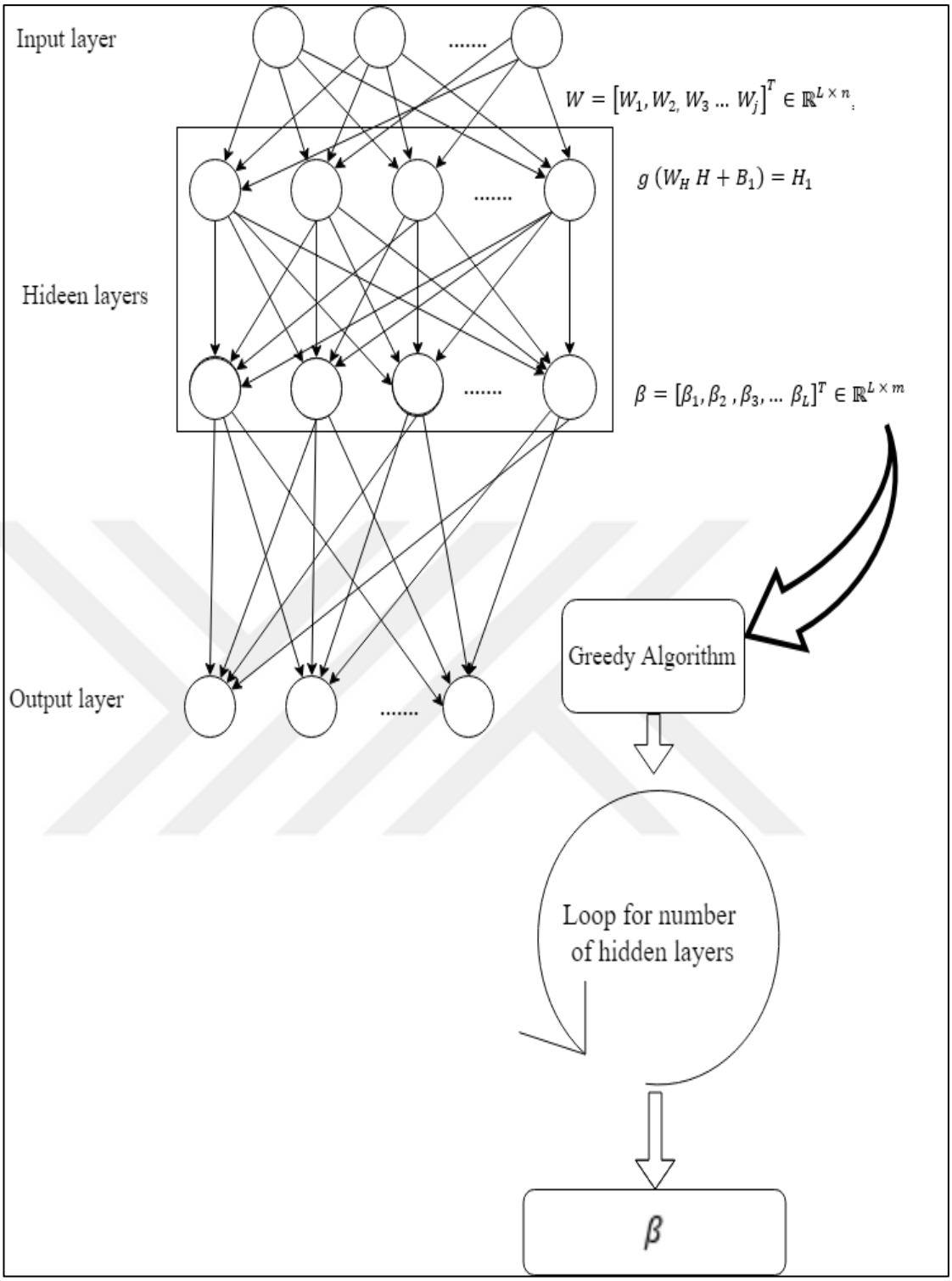

Figure 5.1 Potential sparse MELM

REFERENCES

[1] Alchikh Alnajar, E., Kayhan, S. (2018). Classification Comparison For Different Extreme Learning Machine Algorithms. *Ejons V – International Conference On Mathematics – Engineering – Natural Medical Sciences*.

[2] Bahnsen, A. C., Aouada, D., Ottersten, B. (2015). Example-dependent Cost-sensitive Decision Trees. *Expert Systems with Applications*.

[3] Chen, D., Tian, Y., Liu, X. (2016). Structural Nonparallel Support Vector Machine For Pattern Recognition. *Pattern Recognition*.

[4] Chen, S., Donoho, D., Saunders, M. (2001). Atomic Decomposition By Basis Pursuit. *Society for Industrial and Applied Mathematics*.

[5] Crockett, K., Latham, A., Whitton, N. (2016). On Predicting Learning Styles In Conversational Intelligent Tutoring Systems Using Fuzzy Decision Trees. *Journal of Human Computer Studies*.

[6] Donoho, D., Elad, M. (2002). Optimally Sparse Representation In General (Nonorthogonal) Dictionaries Via L1 Minimization. *Proceedings of the National Academy of Sciences*.

[7] Elad, M. (2018). Sparse Modeling in Image Processing and Deep Learning. *IEEE SigPort*. Retrieved 1 1, 2020, from http://www.ipam.ucla.edu/abstract/?tid=14750pcode=DLT2018

[8] *Extreme Learning Machines*. (2013). (Nanyang Technological University) Retrieved June 11, 2018, from http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html

[9] Huang, G. B., Zhu, Q. Y., Siew, C. K. (2004). Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks. *IEEE*.

[10] Huang, G., Zhu, Q., Siew, C. (2006). Extreme Learning Machine: Theory And Applications. *Neurocomputing*, 489–501.

[11] Hubel, D., Wiesel, T. (1959). Receptive Fields Of Single Neurones In The Cat's Striate Cortex. *The Journal of Physiology*.

[12] Ince, M., Ghofrani, S., Sengur, A., Alcin, O. F. (2014). Ga-selm: Greedy Algorithms For Sparse Extreme Learning. *Measurement*.

[13] Lertworaprachayaa, Y., Yang, Y., John, R. (2014). Interval-valued Fuzzy Decision Trees With Optimal Neighbourhood Perimeter. *Applied Soft Computing*.

[14] Li, M. B., Huang, G., Saratchandran, P., Saratchandran, N. (2005). Fully Complex Extreme Learning Machine. *Neurocomputing,* **68**, 306-314.

[15] Li, Q., Yang, X., Wu, W., Liu, K., Jeon, G. (2018). Multi-Focus Image Fusion Method for Vision Sensor Systems via Dictionary Learning with Guided Filter. *Sensors*.

[16] Liang, N., Huang, G.-B., Saratchandran, P., Sundararajan, N. (2006). A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks. *IEEE, 17*, 1411-1423.

[17] Mandal, I., Sairam, N. (2011). *Enhanced Classification Performance Using Computational Intelligence*. Springer.

[18] Mathworks. (n.d.). *Mean Squared Normalized Error Performance Function*. Retrieved from https://www.mathworks.com/: https://www.mathworks.com/help/deeplearning/ref/mse.html

[19] Peng, J.-X., Rafferty, K., Ferguson, S. (2016). Building Support Vector Machines In The Context Of Regularized Least Squares. *Neurocomputing*.

[20] Praveena, M., Jaiganesh,V. (2017). A Literature Review on Supervised Machine Learning Algorithms and Boosting Process. *International Journal of Computer Applications*.

[21] Qu, B., Lang, B., Liang, J., Qin, A., Crisalle, O. (2015). Two-hidden-layer Extreme Learning Machine For Regression And Classification. *Neurocomputing*.

[22] Ray, K., Raj, S. (2018). Sparse Representation Of Ecg Signals For Automated Recognition Of Cardiac Arrhythmias. *Expert Systems with Applications*, 49-64.

[23] Roesler, O. (2013). *EEG Eye State Data Set*. (Baden-Wuerttemberg Cooperative State University (DHBW)) Retrieved 1 4, 2020, from https://archive.ics.uci.edu/ml/datasets/EEG+Eye+State#

[24] Sun, Z., Yu, Y. (2017). Sparse coding extreme learning machine for classification. *Neurocomputing*, 50-56.

[25] Tang, J., Deng, C., Huang, G.-B. (2015). Extreme Learning Machine for Multilayer Perceptron. *IEEE*.

[26] Tropp, J. (2004). Greed is Good: Algorithmic Results for Sparse Approximation. *IEEE*.

[27] Utkin, L., Zhuk, Y. (2016). An One-class Classification Support Vector Machine Model By Interval-valued Training Data. *Knowledge-Based Systems*.

[28] Waugh, S. (1995, 12 1). *Abalone Data Set*. (Department of Computer Science, University of Tasmania) Retrieved 1 4, 2020, from https://archive.ics.uci.edu/ml/datasets/Abalone

[29] Wright, J., Yang, A. Y., Ganesh, A., Sastry, S., Ma, Y. (2009). Robust Face Recognition via Sparse Representation. *IEEE*.

[30] Xiao, D., Li, B., Mao, Y. (2017). A Multiple Hidden Layers Extreme Learning Machine Method and Its Application. *Hindawi*.

[31] Zong, W., Huang, G.-B., Chen, Y. (2013). Weighted Extreme Learning Machine For Imbalance Learning. *Neurocomputing,* **101**, 229-242.