PRUNED EXTREME LEARNING MACHINES OPTIMIZATION AND INCREASING PERFORMANCE AND PREDICTIONS ON LARGE REAL WORLD PROBLEMS

LAVNEET SINGH

A Thesis Submitted in Partial Fulfilment of the Requirements of the Degree of Doctor of Philosophy

2015

Faculty of Education Science Technology and Mathematics
ACKNOWLEDGEMENTS

This thesis is the end of my journey in obtaining my Ph.D. I have not traveled in a vacuum in this journey. This thesis has been kept on track and been seen through to completion with the support and encouragement of numerous people including my well-wishers, my friends, colleagues and various institutions. At the end of my thesis, it is a pleasant task to express my thanks to all those who contributed in many ways to the success of this study and made it an unforgettable experience for me.

At this moment of accomplishment, first of all I would like to express my special appreciation and thanks to my advisor Dr. Girija Chetty. This work would not have been possible without her guidance, support and encouragement. Under her guidance I successfully overcame many difficulties and learned a lot. You have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a Data scientist. Your advice on both research as well as on my career have been priceless.

I would also like to thank my rest of my supervisory panel, Prof. Michael Wagner, and Dean of ISE Faculty, Prof. Dharmendra Sharma, and Dean of ESTEM faculty, Prof. Geoffrey Riordan for their continuous support throughout my PhD candidature. I would also like to thank you for making my stay in the faculty an enjoyable moment, and for their timely and brilliant comments and suggestions, from time to time.
Very special thanks to the University Research Office who gave me prestigious Weeden’s scholarship for giving me the opportunity to carry out my doctoral research and for their financial support. It would have been impossible for me to even start my study had they not given me a scholarship.

Of course no acknowledgments would be complete without giving thanks to my parents. I especially thank my mom (Mrs Rajvinder Kaur), dad (Dr. Jaspal Singh). Both have instilled many admirable qualities in me and given me a good foundation with which to meet life. They’ve taught me about hard work and self-respect, about persistence and about how to be independent. Mom, especially, was a great role model of resilience, strength and character. Both have always expressed how proud they are of me and how much they love me. I too am proud of them and love them very much. I am grateful for them both and for the ‘smart genes’ they passed on to me.

My sister Savleen Singh has been my best friend all my life and I love her dearly and thank her for all her advice and support. Special thanks are also due to my bro-in-law Major Sunny Gandhi for his amazing support. It is amazing to have family close by so far away from home.
Many of my friends and colleagues helped me both morally and academically at various stages during the period of my study but the number is too great to name them all. Any omissions in acknowledgement do not decipher as ingratitude.

Lavneet Singh
Abstract

Feed-forward neural networks have been extensively used in many fields due to their ability: (1) to approximate complex nonlinear mappings directly from the input samples; and (2) to provide models for a large class of natural and artificial phenomena that are difficult to handle using classical parametric techniques. On the other hand, there is a lack of faster learning algorithms for neural networks. The traditional learning algorithms are usually far slower than required. It is not surprising to see that it may take several hours, several days, and even more time to train neural networks by using traditional methods.

(Huang et al., 2006) proposed a new novel algorithm known as the Extreme Machine Learning (ELM) for single hidden layer feed forward neural network, which has less computational time and faster speed even on large datasets. The main working core of ELM is random initialization of weights rather than learning through slow process via iteratively gradient based learning, called as the backpropagation. Significant work have been done in past for better generalization, faster learning and rate of convergence for ELMs. In spite of that, unfortunately, ELM suffers with certain limitations, such as outliers, irrelevant variables and number of hidden nodes. To address these limitations of ELM, constructive and heuristic approaches have been proposed in the literature.
The accuracy and performance of machine learning and statistical models are still based on tuning certain parameters and optimization for generating better predictive models of learning that is based on the training data. Larger datasets and samples are also problematic, due to increase in computational times, complexity and bad generalization due to outliers. Using the motivation from extreme learning machine (ELM), we propose RANSAC multi model response regularization for multiple models to prune the large number of hidden nodes to acquire better optimality, generalization and classification accuracy of the network in ELM. Experimental results on different benchmark datasets and real time problems showed that proposed algorithm optimally prunes the hidden nodes, better generalization and higher classification accuracy compared to other algorithms, including SVM, OP-ELM for binary and multi-class classification and regression problems.

In the first part of this thesis, I have done an extensive investigation of various classifiers that have been known to perform very well for different machine learning problems, and proposed an improved and highly accurate hybrid ensemble classifiers based on Support Vector Machines (SVM and Extreme Learning Machines for protein folding recognition. In contrast to protein folds prediction, it’s very hard to classify its various folds with its different amino acids attributes due to the limited training data availability. Thus, our proposed classifier involves dimensionality reduction using PCA and LDA prior to classification.
The second part of this thesis presents a principled approach for investigating brain abnormalities based on wavelet based feature extraction, PCA based feature selection and deep and optimized Pruned extreme machine learning based classification comparative to various other classifiers.

The third part of this thesis presents the proposed architectural design for email personalization using ourdeal database based on grad boost with optimized pruned extreme learning machines as base estimators. In this experimental study, we also conducted a depth dive in data analysis to find each members behaviour and important attributes which plays a significant role in increasing clicks rates in personalized emails.
Table of Contents

Chapter-I INTRODUCTION 1-40
1.1 What is Machine Learning? 1
1.2 Subdivisions of Machine Learning 5
1.3 Machine Learning Background 7
1.3.1 Supervised Learning 7
1.3.2 Feed forward neural network 9
1.3.3 Adaptive Learning Rate Algorithms in an Optimization Context 12
1.4 Hybrid supervised learning 14
1.4.1 Constructive Neural Networks 14
1.4.2 Cascade Correlation Neural Network 17
1.4.3 Extreme Learning Machines 26
1.5 Objectives and Scope of the Thesis 27
1.6 Scientific Contributions of the Thesis 34
1.7 Peer reviewed Publications 38

Chapter-II ARTIFICIAL NEURAL NETWORKS 41-66
2.1 Introduction 41
2.2 The perceptron algorithm 45
2.3 Feed-Froward Network 48
2.4 Back Propagation Network 49
2.5 Back propagation Algorithm 53
2.6 Activation Function 58
2.7 Rate of Learning 62
2.8 Limitations of Feed Froward and Back Propagation Neural Network 64

Chapter-III LITERATURE REVIEW 67-130
3.1 Supervised Learning 68
3.2 Artificial Neural Networks and its variants 69
3.3 Constructive Neural Networks 82
3.3.1 Formulation of Constructive Neural Networks 84
3.4 Cascading Neural Networks 91
3.5 Extreme Machine Learning 98
3.6 Conclusion 127
3.7 Roadmap of scientific contribution of thesis as follows 129

Chapter-IV PRUNED ANNULAR EXTREME LEARNING MACHINES OPTIMIZATION BASED ON RANSAC MULTI MODEL RESPONSE REGULARIZATION 131-164
4.1 Extreme Learning Machine Networks 131
4.2 Regularized Extreme Learning Machines 134
4.3 The L1 penalty as LASSO Regularization 137
4.4 The L2 Penalty as Tikhonov regularization 140
4.5 ANNULAR ELM 142
4.6 RANSAC 144
4.7 RANSAC Multi Model Response Regularization 145
4.7.1 RANSAC Multi Model Response Regularization for Regression problems 145
4.7.2 RANSAC Multi Model Response Regularization for binary and multiclass problems 149
4.8 Experimental Results 151
4.8.1 Classification Results 154
4.8.2 Regression Results 157
4.9 Conclusions 163

Chapter-V PROTEIN FOLDING RECOGNITION USING PRUNED ANNULAR EXTREME LEARNING MACHINES 165-190

| 5.1 Introduction | 165 |
| 5.2 Protein Folding Recognition Problem | 168 |
| 5.3 Protein Sequence, Structure, and Function | 169 |
| 5.4 1-D, 2-D, and 3-D Protein Structure Prediction | 170 |
| 5.5 Protein Database and Its Features | 174 |
| 5.5.1 Protein Database | 174 |
| 5.5.2 Feature Vectors | 175 |
| 5.6 Feature Selection and Classification | 175 |
| 5.6.1 Support Vector Machine Classifiers (SVM) | 175 |
| 5.6.2 Quadratic Discriminant Analysis | 177 |
| 5.6.3 Feature Selection | 179 |
| 5.6.4 Generalized Linear Model | 179 |
| 5.6.5 Sequential Feature Selection Algorithm | 181 |
| 5.6.6 The PCA/LDA-Based Pruned Annular Extreme Learning Machines Algorithm | 182 |
| 5.7 Experiments | 184 |
| 5.7.1 Feature Selection algorithms | 184 |
| 5.7.2 Exemplar Results | 185 |
| 5.8 Conclusions and Future Work | 189 |

Chapter VI USING PRUNED ANNULAR EXTREME LEARNING MACHINES FOR IDENTIFYING THE BRAIN ABNORMALITIES FROM MRI STRUCTURAL IMAGES 191-214

<p>| 6.1 Introduction | 191 |
| 6.1.1 Background | 194 |
| 6.2 Material and Methods | 195 |
| 6.2.1 Datasets | 195 |
| 6.3 Feature Extraction | 197 |
| 6.3.1 Decomposition of images Using Wavelets | 197 |
| 6.3.2 Discrete wavelets transform (DWT) | 198 |
| 6.3.3 Coarse Image Segmentation | 199 |
| 6.4 Image Segmentation | 200 |
| 6.4.1 K-Means clustering based Coarse Image Segmentation | 200 |
| 6.4.2 K-Means clustering based Coarse Image Segmentation Algorithm | 202 |
| 6.5 Trained Classifiers | 203 |
| 6.6 Experiments and Results | 207 |
| 6.6.1 Level of wavelet decomposition | 207 |
| 6.6.2 Attribute Selection and Classification | 208 |
| 6.7 Conclusions | 213 |</p>
<table>
<thead>
<tr>
<th>Chapter VII</th>
<th>EMAIL PERSONALIZATION AND USER PROFILING USING RANSAC MULTI MODEL RESPONSE REGRESSION BASED OPTIMIZED PRUNING EXTREME LEARNING MACHINES AND GRADIENT BOOSTING TREES</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>7.1.1</td>
<td>User Profiling</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Content Management</td>
</tr>
<tr>
<td>7.1.3</td>
<td>Web site publishing</td>
</tr>
<tr>
<td>7.1.4</td>
<td>Information acquisition and searching</td>
</tr>
<tr>
<td>7.2</td>
<td>Email Personalization</td>
</tr>
<tr>
<td>7.3</td>
<td>Gradient Boost Trees</td>
</tr>
<tr>
<td>7.4</td>
<td>Extreme Learning Machines</td>
</tr>
<tr>
<td>7.5</td>
<td>Optimized Extreme Learning Machines as Base Estimators for Gradient Boosting Trees</td>
</tr>
<tr>
<td>7.5.1</td>
<td>RANSAC Multi Model Response Regularization</td>
</tr>
<tr>
<td>7.5.1.1</td>
<td>RANSAC Multi Model Response Regularization for Regression problems</td>
</tr>
<tr>
<td>7.6</td>
<td>Experimental Results</td>
</tr>
<tr>
<td>7.7</td>
<td>Conclusions</td>
</tr>
<tr>
<td>Chapter VIII</td>
<td>Conclusions and Future Work</td>
</tr>
<tr>
<td></td>
<td>Appendices</td>
</tr>
<tr>
<td></td>
<td>References</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Classification datasets attributes and classes</td>
<td>153</td>
</tr>
<tr>
<td>4.2</td>
<td>Regression datasets attributes and classes</td>
<td>154</td>
</tr>
<tr>
<td>4.3</td>
<td>Experimental results in terms of testing accuracy for binary classification using Sigmoid kernel</td>
<td>155</td>
</tr>
<tr>
<td>4.4</td>
<td>Experimental results in terms of testing accuracy for binary classification using RBF kernel</td>
<td>156</td>
</tr>
<tr>
<td>4.5</td>
<td>Experimental results in terms of testing accuracy for multiclass classification</td>
<td>156</td>
</tr>
<tr>
<td>4.6</td>
<td>Experimental results in terms of number of hidden nodes pruned by proposed method using sigmoid and radial basis functions for binary and multi-class classification</td>
<td>157</td>
</tr>
<tr>
<td>4.7</td>
<td>Experimental results in terms of training root mean square accuracy for regression using sigmoid kernel</td>
<td>157</td>
</tr>
<tr>
<td>4.8</td>
<td>Experimental results in terms of testing root mean square accuracy for regression using Sigmoid kernel</td>
<td>158</td>
</tr>
<tr>
<td>4.9</td>
<td>Experimental results in terms of testing root mean square accuracy for regression using RBF kernel</td>
<td>158</td>
</tr>
<tr>
<td>4.10</td>
<td>Experimental results in terms of number of hidden nodes pruned by proposed method using sigmoid and radial basis functions for regression</td>
<td>159</td>
</tr>
<tr>
<td>5.1</td>
<td>Six features extracted from the protein sequence with their dimensions</td>
<td>175</td>
</tr>
<tr>
<td>5.2</td>
<td>Recognition ratio obtained using various feature selection algorithms</td>
<td>187</td>
</tr>
<tr>
<td>5.3</td>
<td>Comparison among different methods</td>
<td>187</td>
</tr>
<tr>
<td>5.4</td>
<td>Comparison among different methods using multiple kernels</td>
<td>187</td>
</tr>
<tr>
<td>5.5</td>
<td>Training and Testing Time computed time spans for PCA/LDA based ELM</td>
<td>188</td>
</tr>
<tr>
<td>6.1</td>
<td>Various Classifiers comparison with respect Average Classification Accuracy(%) and other parameters</td>
<td>210</td>
</tr>
<tr>
<td>6.2</td>
<td>Comparison of pair wise combination of various Attribute Selectors and classifiers with respect to ACC (%)</td>
<td>210</td>
</tr>
<tr>
<td>6.3</td>
<td>Comparison using PCA and other feature attribute evaluators in terms of ACC (%)</td>
<td>211</td>
</tr>
<tr>
<td>6.4</td>
<td>Classification results using Extreme Machine Learning and Deep Machine Learning</td>
<td>213</td>
</tr>
<tr>
<td>7.1</td>
<td>Comparative analysis using grad boost classifier with proposed base estimator and non-base estimator</td>
<td>240</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Automatic classification for handwriting recognition based on supervised artificial neural network learning</td>
<td>8</td>
</tr>
<tr>
<td>1.2</td>
<td>Generic supervised training scheme</td>
<td>10</td>
</tr>
<tr>
<td>1.3</td>
<td>Architecture of a feed forward neural network.</td>
<td>11</td>
</tr>
<tr>
<td>1.4</td>
<td>Development Architectures of Constructive Artificial Neural Networks.</td>
<td>16</td>
</tr>
<tr>
<td>1.5</td>
<td>Cascade Correlation Neural Network Architecture</td>
<td>21</td>
</tr>
<tr>
<td>1.6</td>
<td>Architecture Representation of estimation of weights in different layers of cascade correlation neural network</td>
<td>23</td>
</tr>
<tr>
<td>2.1</td>
<td>Structural representation of Neuron and its components</td>
<td>42</td>
</tr>
<tr>
<td>2.2</td>
<td>D Representation of Neural Networks in Human Brain communication through electric pulses among neurons</td>
<td>43</td>
</tr>
<tr>
<td>2.3</td>
<td>Working Principle of Artificial Neuron</td>
<td>44</td>
</tr>
<tr>
<td>3.1</td>
<td>Development Architectures of Constructive Artificial Neural Networks.</td>
<td>84</td>
</tr>
<tr>
<td>4.1</td>
<td>Lasso plot of lambda on diabetes dataset</td>
<td>138</td>
</tr>
<tr>
<td>4.2</td>
<td>Lasso plot of lambda parameter using regularization approach of L1 penalty on output of Hidden Layer for all predictors as hidden units.</td>
<td>139</td>
</tr>
<tr>
<td>4.3</td>
<td>Architectural Design of Annular ELM based on RANSAC Multi Model Response Regularization</td>
<td>144</td>
</tr>
<tr>
<td>4.4</td>
<td>ROC Curve using Extreme Learning Machine Algorithm of Auto Dataset for Multiclass Classification.</td>
<td>161</td>
</tr>
<tr>
<td>4.5</td>
<td>Confusion Matrix depicting true positive and false positive rate using Extreme Learning Machine Algorithm of Auto Dataset for Multiclass Classification.</td>
<td>161</td>
</tr>
<tr>
<td>4.6</td>
<td>ROC Curve using Proposed Extreme Learning Machine Algorithm of Auto Dataset for Multiclass Classification.</td>
<td>162</td>
</tr>
<tr>
<td>4.7</td>
<td>Confusion Matrix depicting true positive and false positive rate using Extreme Learning Machine Algorithm of Auto Dataset for Multiclass Classification.</td>
<td>162</td>
</tr>
<tr>
<td>5.1</td>
<td>Three-dimensional model of converting primary and secondary structure into the 3-D protein structure for various folds</td>
<td>169</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>A model α-helix shows the hydrogen bonds (dotted lines) between oxygen and hydrogen atoms of the fourth amino acid up the chain.</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>CDF value with respect to p value of protein dataset</td>
<td></td>
</tr>
<tr>
<td>5.5</td>
<td>Classification of 27 fold protein sequences using QDA-ELM Classifiers</td>
<td></td>
</tr>
<tr>
<td>5.6</td>
<td>Classification of 27 fold protein sequences using PCA-SVM.</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>T2, weighted an axial MRI Brain Image; (b) T2, weighted an axial MR brain image as abnormal brain</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>(a) Coarse Segmented MRI Image based on above algorithm (b) ROI segmented image of White Lesions</td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>Architectural design of email personalization</td>
<td></td>
</tr>
<tr>
<td>7.2</td>
<td>Density Distribution of member’s data with respect to subscribers and non-subscribers in 1 year tenure scale</td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>Density Distribution of member’s data with respect to subscribers and non-subscribers in 1-10 location scale</td>
<td></td>
</tr>
<tr>
<td>7.4</td>
<td>Distribution of deviance for training and testing data for 0-300 iterations.</td>
<td></td>
</tr>
<tr>
<td>7.5</td>
<td>Distribution of deviance for training and testing data for 0-300 iterations with no regularization and no base estimators for gradient boosting trees</td>
<td></td>
</tr>
<tr>
<td>7.6</td>
<td>Feature importance of multiple attributes as features for member’s data fed into grad boost classifier</td>
<td></td>
</tr>
</tbody>
</table>
Chapter-I

Introduction

1.1. What is Machine Learning?

Every computer based system runs on a sequence of instructions known as algorithms, which carry out input output transformations. There may be various algorithms for the same task and each one is classified in term of different parameters – efficiency, low storage, computational `speed and accuracy. For example- an efficient algorithm to classify spam for emails automatically, in which, the email document is considered as a file of characters, as the input and output should be stamped as email or the spam. The problem is, we can compile thousands of email messages as a spam list but unfortunately, spam changes in time and from different individuals. To the spam classification algorithm efficient for emails, we need to learn from data, and build the knowledge base, so as to extract spam automatically.

Fortunately, due to exponential expansion in the computer technologies, the improvement in machine learning and optimizing the parameters of statistical models has enhanced significantly, by availability of large datasets, and abundance of information of a prior knowledge, with large number of variables and samples. However, an increase in number of variables with respect to samples in large datasets creates an ill structured
problem. Larger datasets and samples can be problematic, due to increase in computational times, complexity and bad generalization due to outliers. But, increase in samples of large datasets can be resolved, thanks to better computer hardware processing units these days with capability of high processing and multi-tasking ability.

Machine Learning and data mining are the programming and statistical models used to optimize performance criteria and better generalization capability using acquired data and information. The accuracy and performance of machine learning and statistical models are based on tuning certain parameters and optimize them for generating better predictive models of learning based, on training data. Currently, most of the machine learning models are based on deterministic learning algorithms rather than non-deterministic approach, which limit the learning capability in real time datasets. To learn from data, and to acquire knowledge from different attributes, so as to identify the process completely, a good and useful approximation is required. The approximation or generalization functions are mathematical models which learn from data with different attributes to identify the complete process. Indeed, it is hard to learn complete process similar to human beings, but it is possible for these data driven models to identify certain patterns or regularities from acquired knowledge.

The power of machine learning comes from recognizing patterns or regularities, with better generalization and approximations to make
predictions. That’s how the future works, making computer machines smarter to have better predictions based on past sample data, and some good examples include, the area of risk analysis, and cost and benefit analysis.

This whole paradigm of learning from data, and prediction using various machine learning algorithms are based on historical data drawn from large databases known as data mining. Of late, data mining techniques are applied for several applications, including financial banking, fraud detection and risk analysis of stock market. Machine learning and data mining techniques use the programmatic statistical models used to optimize certain performance criteria and achieve better generalization capability using acquired data and information. The benefits of using machine learning and statistical models for achieving better accuracy and performance, still needs appropriate tuning of certain parameters and optimization for generating better predictive models of learning from training data. Currently, most of the machines learning models are based on deterministic learning algorithms rather than non-deterministic approach which narrow down their learning applications to real time datasets.

Data mining and machine learning models are extensively used in several application areas, including, manufacturing for optimization, control and troubleshooting, medical image segmentation and diagnosis, and early
prediction of diseases. Further, other similar applications involve, improvement of the network optimization is done in tele-communications by defining the patterns detected by computational intelligence algorithms

Another important application area based on implementing machine learning algorithms is learning through various patterns and processes for predictions related to the World Wide Web. The large amount of information and the network in the World Wide Web is constantly growing. The task of searching information on Internet (World Wide Web) is not easy, particularly a manual in the massive database. Hence, we need to create such a system that is intelligent, and has the ability to learn, does automatic search and can adapt to the changes for providing solution to all possible situations. Of late, from simple Internet search to many problems in computer vision and speech recognition, machine learning and data mining algorithms are used extensively.

In computer vision and face recognition, the system analyses the sample face images of a person and captures the patterns specific to that person. Later on, a learning program can recognize that person by matching the patterns of an image with learned pattern. The complete process of finding and matching patterns is known as pattern recognition.

Thus, machine learning is the set of algorithms which can optimize and predict the performance criteria using past historical data or experience or prior knowledge. A machine learning model is defined by a set of
parameters which need to be tuned or optimized using the training data. The model maybe predictive or descriptive depending on the data and the set of parameters. Most of the machine learning algorithms use the statistical mathematical models of making inference from a sample. The implementations of machine learning algorithms are based on two fold process – to generate the model and define the parameters based on large set of training data. During this step, the data is also processed to churn out irrelevant or missed variables. Once model is set up and learned, its representation and algorithm solution of inference is optimized and validated using testing data. During this complete process, the efficiency of learning algorithm is also optimized with respect to space, time complexity and its predictive accuracy.

1.2. Subdivisions of Machine Learning

The area of machine learning is quite wide and different learning algorithms are implemented to various disciplines. Based on this, different subdivisions of machine learning can be categorised as follows:

1. **Statistics:** Statistics is the main division of machine learning and it involves defining the process in terms of the probability distribution to estimate the value of an unknown function at a new point with respect to the value of this function at a set of sample points. Different theories of statistical machine learning can be found in further detail in (Anderson, 1958).
2. **Brain Models:** Non-linear elements with weighted inputs are the simple biological models of neurons in living brain. Based on non-linear model of neurons in brain, several researchers including (McCulloch & Pitts, 1943), (Rosenblatt, 1958), (Nilsson, 1990), (Sejnowski, Koch, & Churchland, 1988) proposed several machine learning techniques known as neural networks. Work inspired by this school is sometimes called the connectionism, the brain-style computation, or the sub-symbolic processing.

3. **Adaptive Control Theory:** Adaptive control theory is a parametric model of machine learning of estimating unknown parameters and changing them during the operations, and controlling the process to track changes. For example – controlling a robot based on sensory inputs.

4. **Psychological Models:** Significant studies have been done to know the learning process and human behaviour by psychologists. Related work led to a number of early decision trees(Hunt, Marin, & Stone, 1966) and semantic networks(Anderson, 1983) and Reinforcement learning (Barto, Sutton, & Watkins, 1989) which is based on reward stimuli of goal seeking behaviour based on learning in animals.

5. **Evolutionary Models:** Evolutionary models are inspired from nature, where not only do individual animals learn to perform better, but species evolve to be a better fit in their individual contexts. These techniques are based on modelling with certain aspects of
biological evolutions which improve the performance of learning methods of computer algorithms. Genetic algorithms (Goldberg & Holland, 1988) and genetic programming (Koza, 1995) are the most prominent computational techniques for evolution.

1.3. Machine Learning Background

1.3.1. Supervised Learning

Supervised learning is a form of learning by formalizing learning from supervised examples. In supervised learning, learning is done with two sets of data with training and testing dataset, where the model is learnt using a set of labelled examples, and then validated on test set to identify unlabelled examples with highest possible accuracy. The goal of supervised learner is to develop a rule that can identify unlabelled elements in the test set. There are many different approaches that try to build the best possible method for classification based on training and testing dataset. In the earlier work, several supervised learning methods have been proposed for classification and regression problems. In this study, the emphasis is on supervised learning using artificial neural networks and its hybrid adaptive variants, as well as strategies for their optimization, for better performance and generalization, for the classification and the regression problems. To appreciate the paradigm of supervised learning based on artificial neural networks, let us consider a classification problem for classifying input patterns of different classes of handwritten
recognition. For input-output transformation, supervised learning based on neural networks is implemented by tuning certain parameters, and then they are optimized to make as few errors as possible.

Figure-1.1: Automatic classification for handwriting recognition based on supervised artificial neural network learning

Supervised learning is based on training, and then optimizing the parameters. Now, an important question arises, on how can we optimize these parameters? By assuring that the data vector \( X \) contains inputs, in several sets - as number of samples, each with labelled outputs. An important question arises how many samples, which attributes of the input data, how important are these attributes in terms of learning the relationship correctly between inputs and outputs. In other words, how can we optimize these parameters to ensure the best learning model? Assuming a data \( X \) which contain number of samples as input with correctly labelled output, the representation for database \( X \) of input-output pairs, \( x^{in}(\mu), t^{out}(\mu) \) can be represented as
\[ X = \{(x_{\text{in}}(\mu), x_{\text{out}}^{\text{err}}(\mu)); 1 \leq \mu \leq \text{P}\} \]  

The output value \( x_{\text{out}}^{\text{err}}(\mu) \) in Equation (1) is called as the output value with database \( X \), to adapt the parameters of the input-output transform. Thus, our study is not limited to by analysing the learning performance of machine learning method via input-output pairs in database but also includes optimizing parameters for input-output mapping to generalize new data.

Supervised learning is based on training algorithm which is used for updating the weights. During the learning process, the desired output for each set of inputs is known and fed into the network. Update and difference of weights is calculated by the difference between the desired and actual output to globally minimize the difference. It’s a difficult task for rapid computation of global minimum where the number of network weights are high and corresponding non convex multimodal error function possess multitudes of local minima with broad or flat regions adjoined with narrow steep ones.

1.3.2. Feed forward neural network

One of the supervised learning methods based on ANNs is feed forward neural network (FNN). In FNNs, neurons are defined as layers, where neurons of each layer are interconnected and the inputs are propagated
through various layers of the network. The loss function of the FNN is defined as

$$\text{net}_j^l = \sum_{i=1}^{n_{l-1}} w_{ji}^{l-1, l} y_{i}^{l-1}, y_j^l = f(\text{net}_j^l)$$ (2)

Where in Equation (2), \( \text{net}_j^l \) is for the \( j^{th} \) neuron in the \( l^{th} \) layer \((j-2,\ldots,n)\), the sum of its weighted inputs. The weights from the \( i^{th} \) node at the \((l-1)\) layer to the \( j^{th} \) neuron at the \( l^{th} \) layer are denoted by \( w_{ji}^{l-1, l} \), \( y_j^l \) is the output of the \( j^{th} \) neuron that belongs to the \( l^{th} \) layer, and \( f(\text{net}_j^l) \) is the \( j^{th} \) node activation function. If there is a fixed, finite set of input-output samples, the squared error over the training dataset, which contains \( P \) representative samples as the neurons of the next layer, the inputs are propagated through the various layers of the network moving from the input layer to the output layer as shown in Figure 1.2.

\[\text{Figure 1.2: Generic supervised training scheme.}\]
The operation of an FNN is usually based on the following equations:

\[ E(w) = \sum_{p=1}^{p_n} \sum_{j=1}^{j_m} (y_{jp}^L - t_{jp})^2 \]  

This Equation (3) gives the general form of the FNN error function to be minimized, in which \( t_{jp} \) specifies the desired response at the \( j \)th output neuron for the stimulus \( p \) and \( y_{jp}^L \) is the output of the \( j \)th neuron at layer \( L \) that depends on the weight of the network, and \( f \) is a nonlinear activation function as depicted in Figure 1.3.

A variety of approaches adapted from unconstrained optimization theory have been applied to solve this problem. The Back-Propagation (BP) algorithm (Rumelhart et al., 1986) is widely recognized as a powerful tool for training FNNs using information from the first derivatives of the error function. In an attempt to use not only the gradient of the error function but also the second derivative to accelerate the learning process training algorithms that apply nonlinear conjugate gradient methods, such as the
Fletcher–Reeves or the Polak–Ribiere methods (Moller, 1993; Van der Smagt, 1994), or variable metric methods, such as the Broyden–Fletcher–Goldfarb–Shanno method (Watrous, 1987; Battiti, 1992), or even Newton’s method (Parker, 1987) have been proposed. However, these approaches are computationally intensive for FNNs with several hundred weights: derivative calculations as well as sub-minimization procedures (for the case of nonlinear conjugate gradient methods) and approximations of various matrices (for the case of variable metric and quasi-Newton methods) are required. Moreover, approximations of the Hessian matrix made during training may be close to singular, or badly scaled, and as a consequence they might produce inaccurate results. Furthermore, it is not certain that the extra computational cost speeds up far from the minimizer, as is usually the case with the neural network training problem (Dennis & More, 1977; Nocedal, 1992; Battiti, 1992).

1.3.3. Adaptive Learning Rate Algorithms in an Optimization Context

Adaptive learning rate algorithms are usually based on the following approaches:

i. Start with a small learning rate and increase it exponentially- if successive iterations reduce the error. Or else rapidly decrease it- if a significant error increase occurs (Voglet al., 1988; Battiti, 1989).
ii. Start with a small learning rate and increase it - if successive
iterations keep gradient direction fairly constant, or else rapidly
decrease it - if the direction of the gradient varies greatly at each
iteration (Chan & Fallside, 1987).

iii. For each weight an individual learning rate is given, which increases
if the successive changes in the weights are in the same direction
and decreases otherwise. The well-known delta-bar delta method
(Jacobs, 1988; Minai & Williams, 1990) and Silva and Almeida’s
method (Silva & Almeida, 1990) follow the last approach. Another
method of this group, named quickprop, has been presented in
(Fahlman, 1989). Quickprop is based on independent secant steps in
the direction of each weight (Vrahatis et al., 2000). Riedmiller and
Braun in 1993 proposed the Rprop algorithm (Riedmiller & Braun,
1993). The algorithm updates the weights using the learning rate and
the sign of the partial derivative of the error function with respect to
each weight. This approach accelerates training, mainly, in the flat
regions of the error function (Pfister & Rojas, 1993; Rojas, 1996).

Note that all the above mentioned learning rate adaptation methods employ
heuristic coefficients in an attempt to obtain convergence of the BP
algorithm to a minimizer of $E$ and to avoid oscillations. A different
approach is to exploit the local shape of the error surface as described by
the direction cosines. In this case the learning rate is a weighted average of
the direction cosines of weight changes at the current and several previous
successive iterations, the so-called *search-then-converge* schedules combine the desirable features of the standard Least-Mean-Square and traditional stochastic approximation algorithms to gradually decrease the learning rate.

1.4. Hybrid supervised learning

1.4.1. Constructive Neural Networks

For better function approximation and pattern classification, many types of neural networks model have been proposed since past few years. To improve learning accuracy, generalization capability and training time of multilayered neural networks, various tuning parameters of network architecture (number of hidden nodes and layers, weights, network topologies, activation function) need to be optimized and dynamically adjusted during training of the network. Over fitting and under fitting takes place due to inappropriate neural network architecture. A number of ANN architectures and algorithms have been recently proposed in past few years, but adaptive and constructive Neural Networks (CONN) offer a higher optimization solution for finding the parametric parameters and architecture of neural networks. Constructive algorithms start with small network and dynamically grow by adding and training neurons until optimal solutions is calculated. As in simple single layered feed forward neural networks, learning of network and training is done by modification
of interconnection weights between neurons to minimize the error. But constructive algorithms find an optimal solution by automatically fitting the network size to the data without overspecializing yielding better generalization and classification for pattern recognition problems. The advantages of using constructive algorithms compared to other neural networks are

1) During training, the topology of network can be dynamically determined.
2) On non-contradictory finite datasets, they deliver convergence to zero classification errors.
3) During training process of constructive algorithms, elementary thresholds neurons are implemented.
4) The architectural size of neural network can be restricted for reducing complexity and increasing generalization and classification accuracy.
5) Learning parameters need not be fine-tuned and searching of initial weights (as in constructive algorithms), are randomly assigned.
As can be seen in Figure 1.4, circles are representing units and lines are representing connection weights. Direction of activation flow is represented by thick arrow and trainable connection weights are represented by dashed lines. Figure 1.4 (a) depicts the first output phase of training. Figure 1.4 (b) shows the first input phase in which candidate hidden units are trained so that their outputs co-vary with network error. Figure 1.4 (c) depicts the second output phase, in which the best correlating candidate hidden unit is installed in the network. Finally, in Fig 1.4(d), a second hidden unit is installed downstream of the first hidden unit as third output phase.

The well-known CONN algorithms for two class classifications are the Tower and Pyramid (Gallant, 1986), the Tiling (Mazard and Andal, 1989), the Upstart (Fren, 1990) and Perceptron Cascade algorithm (Burgess,
There are further improvements of CONN algorithms done by minimization of classification of errors Irregular partitioning algorithm (Marchand, 1990), the Carve algorithm (Young and Downs, 1998), the Target and Switch algorithm (Campell and Vicente, 1995), the Oil Spot algorithm (Mascioli and Martinelli, 1995), the Constraint based decomposition algorithm (Draghici, 2001), the Decomposition Algorithm for Synthesis and generalization (Subirats, 2008) and Dynamic node Creation algorithm (Ash, 1989).

1.4.2. Cascade Correlation Neural Network

Generalization is the critical capacity for feed forward neural network influenced by many factors and parameters. In past few years, many researchers have been making efforts to improve the generalization capability of neural networks with lesser computational times and maximum accuracy. Several techniques were proposed for optimizing the network such as regularization and pruning algorithms for automatically tuning the network parameters and size of network. One of such hybrid variants of neural network was proposed by (Fahlman & Lebiere, 1989) as cascade correlation neural network algorithm. The cascade correlation learning algorithm was developed to overcome certain problems and limitation as discussed before on the popular back propagation algorithm. The most important of the back propagation limitations is the slow processing of back propagation learning process from training data. Even
on simple benchmark problems, back propagation networks require thousands of epochs to learn training patterns. These limitations are categorised as step wise and moving target problems. The step size problem occurs in back propagation method when it computes only partial first derivatives of the overall error function with respect to weight in the network. Using these derivatives, it can perform gradient descent in weight space, reducing the error in each step. A local minimum of the error function will be generated if taken infinite steps down the gradient vector running multiple epoch in recomputing the gradient.

In practical learning models, we can’t take large step sizes and to define the step size, we need to know the slope of the error function and also take into consideration, higher order derivatives and its curvature in the vicinity of the current point in weight space. A number of schemes have been proposed for dealing with step size problems. Conjugate gradient methods, momentum, Jacobian and approximation of the second derivatives of the error function. These and certain other methods target several problems, such as speed of the back propagation algorithm with faster learning. (Fahlman & Lebiere, 1989) proposed the Quickprop algorithm as one of the successful algorithms for handling the step size problems in back propagation method. Quickprop computes the first partial; derivatives as standard backdrop and uses second order method as Newton’s method to update the weights. The Quickprop was tested on different benchmark
problems and out performs the other backprop like algorithms with a large difference. Quickprop weight update procedure depends on two approximation as firstly, small changes in one weight which have relatively little effect on the error gradient at other weights and secondly, factor is the error function which is updated according to each locally quadratic weight. For each weight, Quickprop saves first partial derivatives as \( \frac{E \partial y}{\partial x} (t - 1) \) and slope is computed in each cycle. For each weight, two slopes and the step between them are used to define parabola and finding the minimum point of this curve.

A second limitation in back propagation learning is the moving target problem. During learning, each unit in the interior of the network is trying to evolve into a feature detector and becomes complicated with all the units which are changing in the same time. The hidden layer of the network cannot link with another hidden layer and their inputs and error function via network output. The backprop learning slows down exponentially by increasing in the number of hidden layers in the network. This slowdown is due to the dilution of the error signal as it propagates backward through the layers in the network which is known as moving target problem. Units in the interior layers of the network constantly shifts in both forward and backward while units evolve, making it impossible for units to come at optimal good solution.
The better interpretation of moving target problem is so called herd effect. Suppose we have two computational tasks, as A and B that is performed by hidden layers in the network. With a set of hidden units, (since units can’t communicate with one another) each unit must decide independently to solve the problem in maximum optimal way. If task A generates a large error single than task B, there a tendency for all the units to concentrate on A rather than B. Once problem A is solved, units will target task B to reduce the remaining error. However, if they all begin to move toward N at once, problem A appears. In most cases, the herd of units will eventually split up and deal with both tasks A and B at once, creating a long computational time.

To combat the step size and moving target problem, the one strategy is to update the weights or units in the network at once, holding the rest constant. Thus, the cascade correlation algorithm allows only one hidden unit to evolve at any given time. Keeping most of the network units frozen most of the time and eliminating the moving target effect, can result in faster learning and the unit that is frozen can quickly be useful in overall optimal solution.

Cascade correlation is combined with cascade architecture in which hidden units are added to the network on at a time and freeze after they have been added and learning algorithm which creates and install the new hidden units. For each hidden unit, the magnitude of correlation need to be
maximized between the new unit output and the residual error signal which is also need to be minimized. The cascade architecture is better explained in Figure 1.5.

![Cascade Correlation Neural Network Architecture](image)

Figure 1.5: Cascade Correlation Neural Network Architecture
It begins with some inputs and one or more output units with no hidden units. The number of inputs and outputs is defined by the problem with respect to input/output representation. Every input is connected to every output unit by a connection with an adjustable weight with a bias input set to +1. The output units produce a linear sum of their weighted inputs by using nonlinear activation function. The activation function can be used as Gaussian, sigmoid, exponential or radial basis functions depending on the linear or non-linearity of data with respect to classification and regression problems. In cascade architecture network, hidden units to the network add one by one. Each new hidden unit receives a connection from each of the network original inputs from every pre-existing hidden unit. The hidden unit’s input weights are frozen at the time the unit is added to the net and only the output connections are trained iteratively. Each new unit add a new one unit layer to the network, unless some of its incoming weights become zero. This leads to designing of high order feature detectors which may lead to very deep networks and high fan in to hidden units. The learning algorithm of cascade architecture begins with no hidden units. The direct input – output connections are trained over the entire training set using cascade architecture by adding hidden nodes one by one. With no need to back propagate through hidden units, any perceptron algorithm can be used as Quickprop or delta rule can be used to train the output weights with better and faster convergence.
Thus, overall, in cascade correlation architecture, the hidden units are added by one and during learning input weights are frozen and the output weights are trained using learning algorithm until residual error is completely minimized and there are no further changes in error reduction. The neuron creation step is divided into two parts as candidate neuron which is connected to all the input and hidden neurons by trainable input connections but its output is not connected to the network. The weights of the candidate neuron can be trained while the other weights are frozen. As can be seen from Figure 1.6, the dashed connections are trained and freeze during training.

![Architecture Representation of estimation of weights in different layers of cascade correlation neural network](image)

Figure 1.6: Architecture Representation of estimation of weights in different layers of cascade correlation neural network

The empty layer shows the place of the candidate neuron where it is activated to after its inputs weights are learned and become frozen.
Secondly, the candidate is connected to the output neurons and all the output connections during training. The complete process is trained iteratively until desired accuracy is obtained. During candidate neuron training, the goal is to maximize the covariance (C) between the network’s and neurons output with respect to the weights of the candidate neuron. The covariance can be obtained by

\[ C = \sum_{o \in O} \left| \sum_{s \in S} (y_s - \bar{y})(e_{o,s} - \bar{e}_o) \right| \quad (4) \]

Where in Equation (4), O is the set of output neurons of the network, s is the set of training samples, y_s is the candidates output for sample s, e_{o,s} is the error at output neuron o for the sample s and \( \bar{y}, \bar{e}_{o,s} \) are the average values of y_s and e_{o,s} respectively over all the samples s \( \in S \). This maximization can be done by using gradient ascent or Quickprop algorithm which is similar to gradient descent performed by delta rule. This is done by calculating the partial derivative of S for each incoming weight, \( \omega_i \), using Equation (4), where \( \sigma \) is the sign of correlation between the candidate’s output and the network output, \( f_p \) is the derivative of the unit’s activation function and \( i_{i,p} \) is the input received from unit i. Once all the candidates have been trained, the most successful candidate is connected to the output layer and the other candidates are discarded. All the output unit weights are then then trained.

\[ \frac{\partial S}{\partial w_i} = \sum_{o,s} \sigma_o (e_{o,s} - \bar{e}_o) f'_p i_{i,o} \quad (5) \]
Where in Equation (5), $\sigma_O$ is the correlation between the candidate’s value and the output $O$, $\hat{f}_O$ is the derivative for pattern $O$ of the candidate unit activation function with respect to the sum of its inputs and $I_{I,O}$ is the input the candidate unit receive from unit I for pattern $O$.

After computing the $\frac{as}{\partial w_i}$ for each input units, $S$ can be maximized by performing gradient descent or Quickprop as training learning algorithm. When $S$ stop improving with stable error rate, we install a new candidate as a unit in the network, freeze its input weights and continue the cycles as mentioned above. Because of the absolute value in the formula for $S$, a candidate unit care only about the magnitude of its correlation with error at given output, but not the sign of correlation. As a rule, if hidden unit correlates positively, it will develop a negative connection weight to that unit, attempting to cancel some error. If the correlation is negative, the output weight will be positive. During addition of hidden unit in the network, instead of using single hidden unit, cascade architecture uses a pool of candidate units, each with different set of random initial weights. The only good thing in using candidate units is that, they do not interact with each other or cause an impact on the active network during training. Thus all candidate units work in parallel and install the units where correlation score is the best.

The use of this pool of candidates is beneficial in two ways: it greatly reduces the chance that a useless unit will be permanently installed because
an individual candidate got stuck during training, and (on a parallel machine) it can speed up the training because many parts of weight-space can be explored simultaneously. The hidden and candidate units may all be of the same type, for example with a sigmoid activation function. Alternatively, we might create a pool of candidate units with a mixture of nonlinear activation functions—some sigmoid, some Gaussian, some with radial activation functions, and so on—and let them compete to be chosen for addition to the active network. The resulting networks, with a mixture of unit-types adapted specifically to the problem at hand, may lead to more compact and elegant solutions than are possible in homogeneous networks.

1.4.3. Extreme Learning Machines

(Huang, Zhu, & Siew, 2006) proposed a novel algorithm known as Extreme Machine Learning (ELM) for single hidden layer feedforward neural network which has less computational time and has faster speed even on large datasets. The main working principle of ELM is random initialization of weights rather than learning through slow process via iterative gradient based learning, such as back-propagation. In Extreme machine learning, the number of hidden nodes and their weights are randomly assigned, which distinguishes the linear differentiable between the output of hidden layer and output layer. The output weights can be determined by linear least square solution of hidden layer output through activation function and the data samples targets.
For $N$ arbitrary distinct samples $(x_i, t_i)$, where $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbb{R}^n$ and $t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbb{R}^m$, standard SLFNs with $N$ hidden nodes and activation function $g(x)$ are mathematically modelled as

$$
\sum_{i=1}^{\hat{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\hat{N}} \beta_i g(w_i \cdot x_j + b_i) = 0_j,
$$

$$
\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T
$$

Where $w_i = [w_{i1}, w_{i2}, \ldots, w_{im}]^T$ is the weight vector connecting the $i^{th}$ hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the $i^{th}$ hidden node and the output nodes, and $b_i$ is the threshold of the $i^{th}$ hidden node. $w_i \cdot x_j$ denotes the inner product of $w_i$ and $x_j$. The output nodes are chosen as linear nodes here.

That standard SLFNs with $N$ hidden nodes with activation function $g(x)$ can approximate these $N$ samples with zero error means that

$$
\sum_{j=1}^{N} \|o_j - t_j\| = 0,
$$
i.e., there exist $\beta_i$, $w_i$ and $b_i$ such that

$$
\sum_{i=1}^{\hat{N}} \beta_i g_i (w_i \cdot x_j + b_i) = t_j, \quad j = 1, \ldots, N.
$$

The above $N$ equations can be written compactly as

$$
H \beta = T,
$$

Where
\[ H(w_1, \ldots, w_N, b_1, \ldots, b_N, x_1, \ldots, x_N) \] (9)

\[
\begin{bmatrix}
g(w_1 \cdot x_1 + b_1) & \cdots & g(w_N \cdot x_1 + b_N) \\
\vdots & \cdots & \vdots \\
g(w_1 \cdot x_N + b_1) & \cdots & g(w_N \cdot x_N + b_N)
\end{bmatrix}_{N \times N} \] (10)

\[
\beta = \begin{bmatrix}
\beta_1^T \\
\vdots \\
\beta_N^T
\end{bmatrix}_{N \times m} \quad T = \begin{bmatrix}
t_1^T \\
\vdots \\
t_N^T
\end{bmatrix}_{N \times m}
\] (11)

H is called the hidden layer output matrix of the neural network; the \( \text{i}^{\text{th}} \) column of H is the \( \text{i}^{\text{th}} \) hidden node output with respect to inputs \( x_1, x_2, \ldots, x_N \).

1.5. Objectives and Scope of the Thesis

In many fields, feed forward neural networks has been extensively used due to the following factors

1. Has the ability to approximate nonlinear mappings in complex patterns from input samples.

2. Classic parametric models are bit unsuccessful for large class of natural and artificial phenomenon. Whereas FNNs provide fine tuning and faster learning process for large class of models.

Neural networks provide faster learning algorithms compared to traditional learning algorithms. From mathematical point of view, feed forward neural network focused on approximation capabilities on universal
Approximation on compact input sets and on a finite set of training samples. Many researchers have explored universal approximation for multilayer feed forward neural networks in terms of the capability. (Hornik, 1993) proposed that if the activation function is continuous, bounded and non-constant, the approximation of continuous mapping can be measured by neural networks over compact input sets. (Leshno, Lin, Pinkus, & Schocken, 1993) extend the works of (Hornik, 1993) results, proving feed forward network with a non-polynomial activation function to approximate continuous functions. In a finite training set, for function approximation, using nonlinear activation function, a single hidden layer SLFN with N hidden nodes can learn N distinct observations. In previous theoretical research and all practical learning algorithms, input weights (linking input layer with hidden layer) and hidden layer biases also need to be adjusted.

In earlier single feed forward network, the parameter needs to be tuned based on the dependency between different layers of parameters (weights and biases). For training and tuning the parameters, gradient descent based methods were used for learning of feed forward neural networks. Gradient descent based learning methods are slow due to slower learning process and easy convergence to local minima. Thus, due to avoid these learning issues, iterative learning steps are being followed to obtain better learning performance. It has been proposed and proved the SLFNs (with N hidden nodes) with random chosen input weights and hidden layer
biases (as random hidden nodes) to learn exactly $N$ distinct observations. Unlike, the previous manually iterative tuning of parameters of SLFNs, one can get better learning outcomes for the feed forward neural network by randomly putting the input weights and hidden layer biases in real time problems.

The classical ELM strongly proves that input weights and hidden layer biases of SLFN can be randomly assigned with infinitely differentiable activation function of hidden layers. With random choice of input weights and hidden layer biases, SLFNs act as a linear system on the output weights, linking hidden layer to the output layer. These output weights of SLFNs can be determined through simple generalised inverse operation of the hidden layer output matrices.

Based on concept of randomization of hidden weights and biases, a new learning algorithm for SLFNs has been proposed in this thesis, known as Pruned Optimized Annular ELM, and is described in detail in Chapter IV. This algorithm results in a learning speed thousand times faster than the traditional SLFN learning algorithm like back propagation (BP) algorithm and classic ELM to obtain better generalization performance. Compared to the traditional learning algorithm, the proposed pruned annular ELM tends to reach smallest training error with the smallest norms of weights. According to Barlet theory, for better generalization performance of neural network, the smaller the norm of weights with small training error, the
better the generalization performance of network we have. Thus, compared to traditional learning algorithm, proposed Annular ELM tends to have good generalization and faster performance for feed forward neural networks. SLFNs with various types of hidden nodes activation function can form disjoint decision regions with arbitrary samples in multi-dimensional cases. Thus, the problem of slow iterative procedure can be solved by random assignment of hidden node parameters and computing least square solution rather than time consuming optimization.

Although the network training utilizing ELM is faster than other algorithms and with improvement in generalization performance, there are still two major unresolved problems: (1) the only parameter that needs to be determined for ELM is the number of hidden nodes in the hidden layer. In former studies, this parameter is usually obtained by trial and error method that may not be optimal. How to choose the most suitable network structures for different applications is still unknown. (2) ELM sometimes requires a large network structure (large number of hidden nodes in the hidden layer) due to the random process in the initial stage. The issue is whether the network complexity can be further reduced without affecting the generalization performance.

Pruning method is one such useful heuristic approach to address the problem of network architecture design. Some researchers have tried to obtain compact ELM networks based on the pruning method. (Miche et al.,
has proposed a pruned ELM (P-ELM) for pattern classification applications, which starts with a large network and then eliminates the hidden nodes with low relevance to the class label. The statistical criteria, the OP-ELM builds the SLFN by applying original ELM algorithm, and then the hidden nodes are ranked by multi-response sparse regression algorithm (MRSR) (Similä & Tikka, 2005). The final model is selected by the leave-one-out (LOO) cross-validation. Although pruning methods are straightforward and easy to understand, they are rather inefficient since most of the time they are dealing with the network structure larger than necessary. And the situation may become worse for cases with a huge number of training samples. Several researchers managed to handle the architecture design through incremental learning. The incremental extreme learning machine (I-ELM) (Huang & Chen, 2007) adds the randomly generated hidden node one-by-one to the hidden layer until achieving an expected training accuracy or reaching the maximum number of hidden nodes. The output weights for all the hidden nodes existing in the network are fixed, and then the output weight for the newly added hidden node can be analytically determined. It is proved that with randomly generated hidden nodes, the I-ELM network can work as a universal approximation. However, I-ELM network may need even larger number of hidden nodes to achieve similar generalization performance as compared to the preliminary ELM. In order to get a compact ELM network, (Huang and Chen, 2007) improved the I-ELM by adding hidden nodes with the largest
error residues among several randomly generated hidden nodes each time. Different from I-ELM, the error minimized extreme learning machine (EM-ELM) (Feng, et al. 2009) (Feng, Huang, Lin, & Gay, 2009) (Feng, Huang, Lin, & Gay, 2009) (G. Feng, Huang, Lin, & Gay, 2009) (G. Feng, Huang, Lin, & Gay, 2009), involves adding the random hidden node group-by-group and all the output weights are updated recursively when new hidden nodes are added into the network until reaching the expected training accuracy, which increases the convergence rate. Theoretically speaking, the EM-ELM network is equivalent to the preliminary ELM with the same network structure. Therefore, they are sharing the same drawback that requires a large network structure for some applications. Besides, how to determine the appropriate expected training accuracy for both of I-ELM and EM-ELM algorithm is still unknown. However, the idea of incremental learning and recursive updating of the output weights does motivate a forward hidden node selection algorithm.

Like pruning algorithms, we could randomly generate a large number of hidden nodes as the candidate reservoir. Instead of pruning the hidden nodes from a large network, a forward selection method can be designed to select the significant hidden nodes from the candidate reservoir to form the smallest model, which can explain the data. The accuracy and performance of machine learning and statistical models are still based on tuning certain parameters and optimization for generating better predictive models of
learning based on training (historical) data. Larger datasets and samples are also problematic, due to increase in computational times, complexity and bad generalization due to outliers. Using the motivation from extreme learning machine (ELM), in this thesis, a novel RANSAC multi model response regularization for multiple models in Chapter IV is proposed, to prune the large number of hidden nodes for better optimality, improved generalization and classification accuracy of the ELM network. Experimental results on different benchmark datasets showed that proposed algorithm can optimally prune the hidden nodes, provide better generalization and a higher classification accuracy compared to other algorithms, including SVM, OP-ELM for binary and multi-class classification.

1.6. Scientific Contributions of the Thesis

This thesis makes the following contributions:

- A new machine learning model is proposed, called an annular ELM based on RANSAC multi model response regularization, and is described in detail in chapter IV. This algorithm allows optimal pruning of the hidden nodes in a network and leads to better generalization and classification accuracy. Experimental results were conducted using comparative analysis of proposed RANSAC multi model response regularization based on annular ELM network
on different benchmark datasets for binary and multiclass classification and regression problems.

From the experimental evaluation, it can be concluded that that the proposed RANSAC multi model response regularized based annular ELM works significantly well in terms of higher classification accuracy, with optimally pruned hidden units. Further, the proposed algorithm allows faster implementation compared to other algorithms, as it implements the ELM with less pruned hidden units without sacrificing the higher generalization capability of ELM network.

- In the area of bio-informatics, large amount of data is harvested with functional and genetic features of proteins. The structure of protein plays an important role in its biological and genetic functions. The next contribution of the thesis is the proposal for a protein structure prediction scheme based on novel learning algorithms – the annular ELM based on RANSAC multi model response regularization and the Support Vector Machine using multiple kernel learning, and is described in detail in Chapter V. The experimental validation of the proposed approach on a publicly available protein data set shows a significant improvement in performance of the proposed approach in terms of accuracy of classification of protein folds using multiple kernels where multiple heterogeneous feature space data are
available. The proposed method provides the higher recognition ratio as compared to other methods reported in previous studies.

- In Chapter VI, the proposed algorithm, Annular ELM based on RANSAC multi model response regularization was extended for magnetic resonance images (MRI) brain imaging problem. Here, various states of brain images were processed for extracting the most significant features, and were classified into normal and abnormal brain images. The novel method based on deep and annular ELM based on RANSAC multi model response regularization, involves the on wavelet transform to initially decompose the images, and then uses various features selection and search algorithms to extract the most significant features of brain regions from the MRI images. By using a comparative study with different classifiers to detect the abnormality of brain images from publicly available neuro-imaging datasets, it was found that a principled approach involving wavelet based feature extraction, followed by selection of most significant features using PCA technique, and the classification using deep and annular ELM based on RANSAC multi model response regularization based classifiers results in a significant improvement in accuracy and faster training and testing time as compared to previously reported studies.

- In Chapter VII, a new architectural design is proposed for email personalization and user profiling using gradient boost trees and
optimized pruned extreme learning machines as base estimators. Email personalization is the process of customizing the content and structure of email according to member’s specific and individual needs taking advantage of member’s navigational behaviour. Personalization is a refined version of customization, where marketing is done automated on behalf of customers’ user profiles, rather than customer requests on his own behalf. There is very thin line between customization and personalization which is achieved by leveraging customer level information using analytical tools. E-commerce is growing fast, and with this growth, companies are willing to spend more on improving the online experience. Further, an in-depth data analysis was conducted to find each member’s behaviour and important attributes that can play a significant role in increasing click rates in personalized emails. From the experimental validation, it was concluded that the proposed method works much better in predicting customer’s behaviour on deals sent in personalized emails compared to other methods proposed in previous literature on this problem.

1.7. **Peer Reviewed Publications**

Most of the contributions made in this thesis were peer reviewed by the scientific community in terms of conference and journal
publications, and publications arising out of the thesis are listed below in chronological order:


Chapter II

ARTIFICIAL NEURAL NETWORKS

2.1. Introduction

The human brain possesses many advantages over a digital computer for solving and handling many tasks in efficient way. We can quickly recognize different faces with different factors like skin colour, lightening and room full with different objects. We can easily understand speech of an unknown person with noise and classify different objects even in different positions and localization. Our brain has the enormous ability for better synchronization to identify patterns and to learn and train from different patterns in a robust way. Despite years of focused research, it is still hard to replicate the properties and working of brain with a factor of evolution. Self-organizing and autonomous, has made human brain astonishingly fast and robust compared to current super smart computers.

The human brain computation of brain is done by a highly interconnected network of neurons consisting of axons, synapses and dendrites and communication is done by sending electric pulses using neural wiring. During the evolution, the cerebrum is one of the areas of the brain that changed most. Cerebral cortex which is approximately 2-4 cm thick is divided into different cortical fields, and has a large number of neurons.
Primary cortical fields are responsible for qualitative management of different perceptions and association cortical fields perform more association and memory thinking process (Kriesel, 2007).

![Fig.2.1: Structural representation of Neuron and its components](image)

Fig.2.1: Structural representation of Neuron and its components
Figure-2.2: Representation of Neural Networks in Human Brain

communication through electric pulses among neurons

As can be seen in Figure 2.1 and 2.2, dendrites are branch like trees from the cell nucleus of the neuron to receive electric pulses from many different sources which is then transferred into the nucleus of the cell. The cell nucleus fired the electric pulses to the other neurons after exceeding the threshold value of activating and inhibiting electric signals by synapses or dendrites. The pulses are transferred to other neurons via axons which are isolated to achieve better conduction of electric signal.

To exhibit the similar characteristics as in biological neurons, (W. McCulloch & W. Pitts, 1943) proposed an mathematical model known as Artificial Neural Network (ANN) which tries to simulate the structure and functionalities of biological neural networks. (F. Rosenblatt,
1958) improved the ANN by using algorithm called “Perceptron” as a binary classifier for multi-layer networks. Artificial neuron, is a mathematical model function, consisting of set of rules as multiplication, summation and activation are the basic building blocks of ANNs. At the input stage of artificial neuron, the input nodes are weighted by multiplying input value with individual weight. Before the output of neuron, there is summation function which is the sum of weighted inputs and bias which is passed through activation function (also known as transfer function) to release the output.

![Image of artificial neuron](image)

**Figure-2.3: Working Principle of Artificial Neuron**

The complexity of real neurons is highly abstracted when modelling artificial Neurons as shown in Figure 2.3. These basically consist of inputs (like synapses), which are multiplied by weights (strength of the respective signals), and then computed by a mathematical function which determines the activation of the neuron. Another function (which may be the identity) computes the output of the artificial neuron (sometimes in dependence of a
certain threshold). ANNs combine artificial neurons in order to process information.

The higher a weight of an artificial neuron is, the stronger the input which is multiplied by it will be. Weights can also be negative, so we can say that the signal is inhibited by the negative weight. Depending on the weights, the computation of the neuron will be different. By adjusting the weights of an artificial neuron we can obtain the output we want for specific inputs. But when we have an ANN of hundreds or thousands of neurons, it would be quite complicated to find by hand all the necessary weights. But we can find algorithms which can adjust the weights of the ANN in order to obtain the desired output from the network. This process of adjusting the weights is called learning or training.

2.2. The perceptron algorithm

The perceptron algorithm, a linear discriminant model, is proposed by Rosenblatt (1962), which is employed in majority of pattern recognition algorithms. It corresponds to a two-class model in which the input vector $x$ is first transformed using a fixed nonlinear transformation to give a feature vector $\phi(x)$, and this is then used to construct a generalized linear model of the form

$$y(x) = f(w^T \phi(x))$$  \hspace{1cm} (1)
Where, the nonlinear activation function $f()$ is given by a step function of the form

$$f(a) = \begin{cases} \begin{array}{ll} +1, & a \geq 0 \\ -1, & a < 0 \end{array} \end{cases} \quad (2)$$

The vector $\phi(x)$ will typically include a bias component $\pi = 1$.

For two-class classification problems, target coding scheme is focussed in which $t \in \{\text{Abid, Fnaiech, & Najim, 2001}\}$, is appropriate in the context of probabilistic models. For the perceptron, to use target values $t = +1$ for class $C_1$ and $t = -1$ for class $C_2$, which matches the output of activation function.

Error function minimization algorithm is used to determine the parameters of the perceptron. However, error function minimization algorithm does not lead to a simple learning algorithm because the error is a piecewise constant function of $w$, with discontinuities wherever a change in because the decision boundary to move across one of the data points. Using the gradient of the error function methods can be applied based on changing $w$, but cannot then be applied if the gradient is zero almost everywhere.

An alternative error function known as the perceptron criterion is based on seeking a weight vector $w$ such that patterns $x_n$ in class $C_1$ will have $w^T \phi(x_n) > 0$, whereas patterns $x_n$ in class $C_2$ have $w^T \phi(x_n) < 0$. Using the $t \in \{-1, +1\}$ target values, all patterns satisfy the $w^T \phi(x_n) t_n > 0$. The perceptron criterion associates zero error with any pattern that is correctly classified,
whereas for a misclassified pattern $x_n$, it tries to minimize the quantity $-w^T \phi(x_n) t_n$. The perceptron function is defined by

$$E_p(w) = \sum_{n \in M} W^T \phi t_n$$ (3)

Where, $M$ denotes the set of all misclassified patterns with a linear error function.

The change in the weight vector is then given by

$$w^{(r+1)} = w^{(r)} - \pi \nabla E_p(w) = w^{(r)} + \pi \phi t_n$$ (4)

Where, $\pi$ is the learning rate parameter and $\tau$ is an integer that indexes the steps of the algorithm. As the weight vector evolves during training, the set of patterns that are misclassified will change.

Considering the effect of a single update in the perceptron learning algorithm, the contribution to the error from a misclassified pattern will be reduced by

$$-w^{(r+1)} T \phi t_n = -w^{(r)} T \phi t_n - (\phi t_n)^T \phi t_n < -w^{(r)} T \phi t_n$$ (5)

Where, set $\eta=1$, and made use of $\|\phi t_n\|>0$. Furthermore, the change in weight vector may have caused some previously correctly classified patterns to become misclassified. Thus the perceptron learning rule is not guaranteed to reduce the total error function at each stage.
However, the perceptron convergence theorem states that if there exists an exact solution (in other words, if the training data set is linearly separable), then the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps. Even when the data set is linearly separable, there may be many solutions, and which one is found will depend on the initialization of the parameters and on the order of presentation of the data points. Furthermore, for data sets that are not linearly separable, the perceptron learning algorithm will never converge.

Aside from difficulties with the learning algorithm, the perceptron does not provide probabilistic outputs, nor does it generalize readily to \( K > 2 \) classes. The most important limitation, however, arises from the fact that (in common with all of the models discussed in this chapter and the previous one) it is based on linear combinations of fixed basis functions.

### 2.3. Feed-Forward Network

Multi-Layer Feed Forward (MLF) neural networks and single layer forward (SLF) feed forward neural networks, trained with a back-propagation learning algorithm, are the most simple implementations of neural networks. A MLF neural network consists of hidden neurons that are ordered into layers (Fig. 2.1). The first layer is called the input layer, the last layer is called the output layer, and the layers between are hidden layers. The mapping function \( T \), that assigns for each neuron \( i \) a subset
where \( T(i) \subseteq V \) which consists of all previous output of the given neuron. Each neuron in a particular layer is connected with all neurons in the next layer. The connection between the \( i^{th} \) and \( j^{th} \) neuron is characterised by the weight coefficient \( w_{ij} \) and the \( i^{th} \) neuron by the threshold coefficient \( \varphi_i \) (Fig.2.2).

The weight coefficient reflects the degree of importance of the given connection in the neural network. The output value (activity) of the \( i^{th} \) neuron \( x_i \) is determined by

\[
x_i = f(\xi_i) \quad \text{(6)}
\]

\[
x_i = \varphi_i + \sum_{j \in T_i} w_{ij} x_j \quad \text{(7)}
\]

Where, \( \xi_i \) is the value of \( i^{th} \) neuron and function \( f(\xi) \) is called transfer function and \( \varphi_i \) is the bias.

The supervised adaptation process varies the threshold coefficients \( \varphi_i \) and weight coefficients \( w_{ij} \) to minimise the sum of the squared differences between the computed and required output values. This is accomplished by minimisation of the objective function \( E \):

\[
\min_{\hat{W}} \sum_{i=1}^{n} (y_i - x_i \hat{w})^2 \quad \text{(8)}
\]

The number of layers and the number of hidden neurons in each hidden layer are user design parameters. The general rule is to choose these design parameters so that the best possible model with as few parameters as
possible is obtained. In general, we have to experiment with different designs and compare the results, to find the most suitable neural network model for the problem at hand. The output neurons in the FF networks in Figures 4 are linear; that is, they do not have any nonlinear activation function after the weighted sum.

2.4. Back Propagation Network

Multilayer perceptron have been applied successfully to solve some difficult and diverse problems by training them in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm. This algorithm is based on the error-correction learning rule. As such, it may be viewed as a generalization of an equally popular adaptive filtering algorithm: the ubiquitous least-mean-square (LMS) algorithm described in Chapter 3 for the special case of a single linear neuron.

Basically, error back-propagation learning consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass, the synaptic weights of the networks are all fixed. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with an error-correction
rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an error signal. This error signal is then propagated backward through the network, against the direction of synaptic connections—hence the name "error back-propagation." The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense. The error back-propagation algorithm is also referred to in the literature as the back-propagation algorithm, or simply back-prop. Henceforth we will refer to it as the back-propagation algorithm. The learning process performed with the algorithm is called back-propagation learning.

A multilayer perceptron has three distinctive characteristics:

1. The model of each neuron in the network includes a nonlinear activation function. The important point to emphasize here is that the nonlinearity is smooth (i.e., differentiable everywhere), as opposed to the hard-limiting used in Rosenblatt's perceptron. A commonly used form of nonlinearity that satisfies this requirement is a sigmoidal nonlinearity defined by the logistic junction:

\[ y_i = \frac{1}{1 + \exp(-v_j)} \]  

Where, \( v_j \) is the induced local field (i.e., the weighted sum of all synaptic inputs plus the bias) of neuron \( j \), and \( y_i \) is the output of the neuron. The presence of non-linearity is important because otherwise the input-output
relation of the network could be reduced to that of a single-layer perceptron. Moreover, the use of the logistic function is biologically motivated, since it attempts to account for the refractory phase of real neurons.

2. The network contains one or more layers of hidden neurons that are not part of the input or output of the network. These hidden neurons enable the network to learn complex tasks by extracting progressively more meaningful features from the input patterns (vectors).

3. The network exhibits a high degree of connectivity, determined by the synapses of the network. A change in the connectivity of the network requires a change in the population of synaptic connections or their weights.

It is through the combination of these characteristics together with the ability to learn from experience through training that the multilayer perceptron derives its computing power. These same characteristics, however, are also responsible for the deficiencies in our present state of knowledge on the behavior of the network. First, the presence of a distributed form of nonlinearity and the high connectivity of the network make the theoretical analysis of a multilayer perceptron difficult to undertake. Second, the use of hidden neurons makes the learning process harder to visualize. In an implicit sense, the learning process must decide which features of the input pattern should be represented by the hidden
neurons. The learning process is therefore made more difficult because the search has to be conducted in a much larger space of possible functions, and a choice has to be made between alternative representations of the input pattern (Hinton, 1989).

The development of the back-propagation algorithm represents a landmark in neural networks in that it provides a computationally efficient method for the training of multilayer perceptron. Although we cannot claim that the back-propagation algorithm provides an optimal solution for all solvable problems, it has put to rest the pessimism about learning in multilayer machines that may have been inferred from the book by Minsky and Papert (1969).

### 2.5. Back Propagation Algorithm

The error signal at the output of neuron; at iteration n (i.e., presentation of the nth training example) is defined by

$$ e_j(n) = d_j(n) - y_j(n), $$  

(10)

Where, neuron j is an output node in equation 10. We define the instantaneous value of the error energy for neuron j as

$$ \frac{1}{2} e_j^2(n) $$

Correspondingly, the instantaneous value $\zeta(n)$ of the total error energy is obtained by summing $\frac{1}{2} e_j^2(n)$ over all neurons in the output layer; these
are the only "visible" neurons for which error signals can be calculated directly. We may thus write

\[ \zeta(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \]

...(11)

Where, the set C in Equation(11) includes all the neurons in the output layer of the network. Let N denote the total number of patterns (examples) contained in the training set. The average squared error energy is obtained by summing \( \zeta(n) \) over all n and then normalizing with respect to the set size N, as shown by

The instantaneous error energy \( \zeta(n) \), and therefore the average error energy \( \zeta_{av} \), is a function of all the free parameters (i.e., synaptic weights and bias levels) of the network. For a given training set, \( \zeta_{av} \) represents the cost function as a measure of learning performance. The objective of the learning process is to adjust the free parameters of the network to minimize \( \zeta_{av} \). Specifically, we consider a simple method of training in which the weights are updated on a pattern-by-pattern basis until one epoch, and that is, one complete presentation of the entire training set has been dealt with. The adjustments to the weights are made in accordance with the respective errors computed for each pattern presented to the network.

The arithmetic average of these individual weight changes over the training set is therefore an estimate of the true change that would result
from modifying the weights based on minimizing the cost function $\zeta_{av}$ over the entire training set. We will address the quality of the estimate later in this section.

Cost function which depicts neuron $j$ being fed by a set of function signals produced by a layer of neurons to its left. The induced local field $v_j(n)$ produced at the input of the activation function associated with neuron $j$ is therefore

$$ u_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) $$

(12)

Where, $m$ is the total number of inputs (excluding the bias) applied to neuron $j$. The synaptic weight $w_{jo}$ (corresponding to the fixed input $y_0= +1$) equals the bias $b_j$ applied to neuron $j$. Hence, the function signal $y_j(n)$ appearing at the output of neuron; at iteration $n$ is

$$ y_j(n) = \phi_j(u_j(n)) $$

(13)

In a manner similar to the LMS algorithm, the back-propagation algorithm applies a correction $\Delta w_{ji}(n)$ to the synaptic weight $w_{ji}(n)$, which is proportional to the partial derivative $\partial \zeta(n) / \partial w_{ji}(n)$. According to the chain rule of calculus, we may express this gradient as:

$$ \frac{\partial \zeta(n)}{\partial w_{ji}(n)} = \frac{\partial \zeta(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial u_j(n)} \frac{\partial u_j(n)}{\partial w_{ji}(n)} $$

(14)
The partial derivative \( \frac{\partial \xi(n)}{\partial w_{ji}(n)} \) represents a sensitivity factor, determining the direction of search in weight space for the synaptic weight \( w_{ji} \).

\[
\frac{\partial \xi(n)}{\partial e_j(n)} = e_j(n)
\]

(15)

Differentiating both sides of Eq. (14) with respect to \( e_j(n) \), we get

\[
\frac{\partial e_j(n)}{\partial y_j(n)} = -1
\]

(16)

Next, differentiating Eq. (16) with respect to \( v_j(n) \), we get

\[
\frac{\partial y_j(n)}{\partial u_j(n)} = \varphi_j'(u_j(n))
\]

(17)

Where, the use of prime (on the right-hand side) signifies differentiation with respect to the argument. Finally, differentiating Eq. (17) with respect to \( w_{ji}(n) \) yields

\[
\frac{\partial u_j(n)}{\partial w_{ji}(n)} = y_i(n)
\]

(18)

The use of Eqs. (16) to (17) in (18) yields
The correction \( \Delta w_{ji}(n) \) applied to \( w_{ji}(n) \) is defined by the delta rule:

\[
\Delta w_{ji}(n) = -\eta \frac{\partial \xi(n)}{\partial w_{ji}(n)}
\]  

(20)

Where, \( \eta \) is the learning-rate parameter of the back-propagation algorithm. The use of the minus sign in Eq. (20) accounts for gradient descent in weight space (i.e., seeking a direction for weight change that reduces the value of \( \xi(n) \)). Accordingly, the use of Eq. (19) in (20) yields

\[
\Delta w_{ji}(n) = \eta \delta_j(n) y_j(n)
\]  

(21)

Where, the local gradient \( \delta_j(n) \) is defined by

The local gradient points to required changes in synaptic weights. According to Eq. (4.14), the local gradient \( \delta_j(n) \) for output neuron \( j \) is equal to the product of the corresponding error signal \( e_j(n) \) for that neuron and the derivative \( \varphi_j'(u_j(n)) \) of the associated activation function.

We now summarize the relations that we have derived for the back-propagation algorithm. First, the correction \( \Delta w_{ji}(n) \) applied to the synaptic weight connecting neuron \( i \) to neuron \( j \) is defined by the delta rule:
\[
\begin{pmatrix}
\text{Weight} \\
\text{correction}
\end{pmatrix}
= 
\begin{pmatrix}
\text{learning} \\
\text{rate parameter}
\end{pmatrix}
\cdot 
\begin{pmatrix}
\text{local} \\
\text{gradient}
\end{pmatrix}
\cdot 
\begin{pmatrix}
\text{input signal} \\
\text{of neuron } j
\end{pmatrix}
\]

Second, the local gradient \( \delta_j(n) \) depends on whether neuron \( j \) is an output node or a hidden node:

1. If neuron \( j \) is an output node, \( \delta_j(n) \) equals the product of the derivative \( \phi_j'(u_j(n)) \) and the error signal \( e_j(n) \) both of which are associated with neuron \( j \).

2. If neuron \( i \) is a hidden node, \( \delta_j(n) \) equals the product of the associated derivative. \( \phi_j'(u_j(n)) \) and the weighted sum of the \( \delta_s \) computed for the neurons in the next hidden or output layer that are connected to neuron \( i \).

### 2.6. Activation Function

The computation of each neuron of the multilayer perceptron requires knowledge of the derivative of the activation function \( \phi(\cdot) \) associated with that neuron. For this derivative to exist, we require the function \( \phi(\cdot) \) to be continuous. In basic terms, differentiability is the only requirement that an activation function has to satisfy. An example of a continuously differentiable nonlinear activation function commonly used in multilayer perceptron is sigmoidal nonlinearity; two forms are described:
1. Logistic Function. This form of sigmoidal nonlinearity in its general form is defined by

$$\varphi_j(u_j(n)) = \frac{1}{1 + \exp(-au_j(n))} \quad a > 0 \text{ and } -\infty < u_j(n) < \infty$$  \hspace{1cm} (23)

Where, \(v_j(n)\) is the induced local field of neuron \(j\). According to this nonlinearity, the amplitude of the output lies inside the range \(0 \leq y_j \leq 1\).

Differentiating Eq. (4.30) with respect to \(v_j(n)\), we get

$$\varphi_j'(u_j(n)) = \frac{a \exp(-au_j(n))}{\left[1 + \exp(-au_j(n))\right]^2}$$  \hspace{1cm} (24)

With \(y_j(n) = \varphi_j(u_j(n))\), we may eliminate the exponential term \(\exp(-a \cdot n)\) from Eq. (4.31), and so express the derivative \(\varphi_j'(u_j(n))\) as

$$\varphi_j'(u_j(n)) = ay_j(n) \left[1 - y_j(n)\right]$$  \hspace{1cm} (25)

For a neuron \(j\) located in the output layer, \(y_j(n) = o_j(n)\). Hence, we may express the local gradient for neuron \(j\) as

$$\delta_j(n) = e_j(n) \varphi_j'(u_j(n))$$  \hspace{1cm} (26)

$$= a \left[d_j(n) - o_j(n)\right] o_j(n) \left[1 - o_j(n)\right].$$  \hspace{1cm} Neuron / is an output node
Where, $o_j(n)$ is the function signal at the output of neuron $j$, and $d_j(n)$ is the desired response for it. On the other hand, for an arbitrary hidden neuron $j$, we may express the local gradient as

$$
\delta_j(n) = \phi_j(u_j(n)) \sum_k \delta_k(n)w_{jk}(n)
$$

(27)

$$
= ay_j(n) [1 - y_j(n)] \sum_k \delta_k(n)w_{jk}(n),
$$

Neuron $j$ is hidden

Note from Eq. (4.32) that the derivative $\phi_j'(u_j(n))$ attains its maximum value at $y_j(n)= 0.5$, and its minimum value (zero) at $y_j(n)= 0$, or $y_j(n)= 1.0$, Since the amount of change in a synaptic weight of the network is proportional to the derivative $\phi_j'(u_j(n))$, it follows that for a sigmoid activation function the synaptic weights are changed the most for those neurons in the network where the function signals are in their midrange. According to Rumelhart et al. (1986a), it is this feature of back-propagation learning that contributes to its stability as a learning algorithm.

2. Hyperbolic tangent function. Another commonly used form of sigmoidal non-linearity is the hyperbolic tangent function, which in its most general form is defined by

$$
\varphi_j(u_j(n)) = a \tanh (bu_j(n)), \quad (a,b) > 0
$$

(28)
Where, \(a\) and \(b\) are constants. In reality, the hyperbolic tangent function is just the logistic function rescaled and biased. Its derivative with respect to \(u_j(n)\) is given by

\[
\phi_j'(u_j(n)) = ab \sec^2(bu_j(n)) = ab \left(1 - \tanh^2(bu_j(n))\right)
\]

\[
= \frac{b}{a} \left[a - y_j(n)\right] \left[a + y_j(n)\right]
\]

For a neuron \(j\) located in the output layer, the local gradient is

\[
\delta_j(n) = e_j(n) \phi_j'(u_j(n))
\]

\[
= \frac{b}{a} \left[d_j(n) - o_j(n)\right] \left[a - o_j(n)\right] \left[a + o_j(n)\right]
\]

(30)

For a neuron \(j\) in a hidden layer, we have

\[
\delta_j(n) = \phi_j'(u_j(n)) \sum_k \delta_k(n) w_{kj}(n)
\]

\[
= \frac{b}{a} \left[a - y_j(n)\right] \left[a + y_j(n)\right] \sum_k \delta_k(n) w_{kj}(n),
\]

(31)

By using Eqs. (30) and (31) for the logistic function and Eqs. (28) and (29) for the hyperbolic tangent function, we may calculate the local gradient \(\delta_j\) without requiring explicit knowledge of the activation function.
2.7. Rate of Learning

The back-propagation algorithm provides an "approximation" to the trajectory in weight space computed by the method of steepest descent. The smaller we make the learning-rate parameter $\eta$, the smaller the changes to the synaptic weights in the network will be from one iteration to the next, and the smoother will be the trajectory in weight space. This improvement, however, is attained at the cost of a slower rate of learning. If, on the other hand, we make the learning-rate parameter $\eta$ too large in order to speed up the rate of learning, the resulting large changes in the synaptic weights assume such a form that the network may become unstable (i.e., oscillatory). A simple method of increasing the rate of learning yet avoiding the danger of instability is to modify the delta rule of Eq.(31) by including a momentum term, as shown by (Rumelhart et al, 1986a)

$$\Delta w_{ji} (n) = a\Delta w_{ji} (n-1) + \eta \delta_j (n) y_i (n)$$  \hspace{1cm} (32)

Where, $a$ is usually a positive number called the momentum constant. It controls the feedback loop acting around $\Delta w_{ji} (n)$, where $z^{-1}$ is the unit-delay operator. Equation (32) is called the generalized delta rule; it includes" the delta rule of Eq. (32) as a special case (i.e., $\alpha = 0$).

1. The current adjustment $\Delta w_{ji} (n)$ represents the sum of an exponentially weighted time series. For the time series to be convergent,
the momentum constant must be restricted to the range $0 \leq |\alpha| < 1$. When $\alpha$ is zero, the back-propagation algorithm operates without momentum. Also the momentum constant $\alpha$ can be positive or negative, although it is unlikely that a negative $\alpha$ would be used in practice.

2. When the partial derivative $\frac{\partial \xi(t)}{\partial w_{ji}(t)}$ has the same algebraic sign on consecutive iterations, the exponentially weighted sum $\Delta w_{ji}(n)$ grows in magnitude, and so the weight $w_{ji}(n)$ is adjusted by a large amount. The inclusion of momentum in the back-propagation algorithm tends to accelerate descent in steady downhill directions.

3. When the partial derivative $\frac{\partial \xi(t)}{\partial w_{ji}(t)}$ has opposite signs on consecutive iterations, the exponentially weighted sum $\Delta w_{ji}(n)$ shrinks in magnitude, so the weight $w_{ji}(n)$ is adjusted by a small amount. The inclusion of momentum in the back-propagation algorithm has a stabilizing effect in directions that oscillate in sign.

The incorporation of momentum in the back-propagation algorithm represents a minor modification to the weight update, yet it may have some beneficial effects on the learning behavior of the algorithm. The momentum term may also have the benefit of preventing the learning process from terminating in a shallow local minimum on the error surface.

In deriving the back-propagation algorithm, it was assumed that the learning-rate parameter is a constant denoted by $\eta$. In reality, however, it
should be defined as $\eta_{ji}$; that is, the learning-rate parameter should be connection-dependent. Indeed, many interesting things can be done by making the learning-rate parameter different for different parts of the network. We provide more detail on this issue in subsequent sections.

It is also noteworthy that in the application of the back-propagation algorithm we may choose all the synaptic weights in the network to be adjustable, or we may constrain any number of weights in the network to remain fixed during the adaptation process. In the latter case, the error signals are back-propagated through the network in the usual manner; however, the fixed synaptic weights are left unaltered. This can be done simply by making the learning-rate parameter $\eta_{ji}$, for synaptic weight $w_{ji}$ equal to zero.

2.8. Limitations of Feed Forward and Back propagation Neural Network

1. Back propagation neural networks (and many other types of networks) are in a sense the ultimate 'black boxes'. Apart from defining the general architecture of a network and perhaps initially seeding it with a random numbers, the user has no other role than to feed the input and watch it train and await the output. In fact, it has been said that with back propagation, "you almost don't know what you're doing". Some freely available software packages (NevProp, bp, Mactivation) do allow the user
to sample the network’s progress at regular time intervals, but the learning itself progresses on its own. The final product of this activity is a trained network that provides no equations or coefficients defining a relationship (as in regression) beyond its own internal mathematics. The network 'IS' the final equation of the relationship.

2. Back propagation networks also tend to be slower to train than other types of networks and sometimes require thousands of epochs. If run on a truly parallel computer system this issue is not really a problem, but if the BPNN is being simulated on a standard serial machine (i.e. a single SPARC, Mac or PC) training can take some time. This is because the machines CPU must compute the function of each node and connection separately, which can be problematic in very large networks with a large amount of data. However, the speed of most current machines is such that this is typically not much of an issue.
Chapter III

LITERATURE REVIEW

This chapter presents the past work done in area of machine learning that is directly relevant to the scope of this thesis. Machine learning studies involve automatic techniques for learning to make accurate predictions based on past observations. There are several applications for Machine Learning (ML), the most significant of which is data mining. People are often prone to making mistakes during analyses or, possibly, when trying to establish relationships between multiple features. This makes it difficult for them to find solutions to certain problems. Machine learning can often be successfully applied to these problems, improving the efficiency of systems and the designs of machines. Machine-learning techniques are being applied to new kinds of problem, including knowledge discovery in data-bases, language processing, robot control, and combinatorial optimization, as well as to more traditional problems such as speech recognition, face recognition, handwriting recognition, medical data analysis, and game playing.

In this chapter, various machine learning algorithms are investigated which are used to find optimal solution for various applications. From past 40 years, significant work has been done in the area of machine learning to classify different application problems, and to achieve higher accuracy and less computational time. Various machine learning methods like
supervised and unsupervised algorithms are used in different applications to classify the extracted features into defined labels. The main purpose of machine learning is to classify set of features represented as instances of any dataset. The features may be continuous, categorical or binary. Supervised machine learning learn through known labels of instances of dataset or their corresponding outputs, in contrast to the unsupervised learning, where instances are unlabelled. (Jain et. al 1999).

3.1. Supervised Learning

As discussed above, significant work has been done in supervised learning on multiple application like image analysis, object recognition etc. An exhaustive review was undertaken by (King, R.D. 1995) for comparison of different classification algorithms on large real world problems known as STATLOG. They used multiple supervised learning algorithms such as Naive Bayes, k-nearest neighbour, neural networks etc. on different datasets from medical image analysis to engineering and finance. They developed a set of dataset descriptors to help decide which algorithms are suited to particular datasets. The objective measures of performance are used as time (training and testing data) and accuracy in terms of cost function. They concluded that the accuracy and performance of supervised learning algorithms crucially depend on the features of particular dataset. Supervised learning algorithms are divided into following categories:

1. Statistical classifiers: k-nearest neighbour, Linear and Quadratic Discriminant Analysis
2. Symbolic or Decision learning classifiers: CART(Classification and Regression Tree) (Breiman et al.,1984), NEWID (Clark and Boswell,1991) and Random Forests.

3. Neural Networks: Back Propagation, Feed-forward network, Extreme Machine learning and Radial Basis Functions (RBF)

4. Ensemble classifiers: Boots-trapping, Boosting

5. Support Vector Machines (SVM)

3.2. Artificial Neural Networks and its variants

The adaptive structural neural networks architecture adapts the network during the training of a given problem. The adaption of the network structure could be at the level of network architecture adaptation, functional adaptation or training parameters adaptation. The adaptation of feed-forward neural networks can also be done by wither evolutionary or non-evolutionary algorithms. In past few years, many researchers use evolutionary algorithms for global optimization of neural networks and their training parameters. The evolutionary algorithms used for adaptive feed-forward neural networks are:-

1. Genetic Algorithms and Programming

2. Ant Colony Optimization

3. Particle Swarm Optimization

Apart from evolutionary algorithms used for improved optimization and generalization in neural networks, non- evolutionary algorithms are also used for better optimized network and tuning the training parameters
of network. One non-evolutionary way for adaptive neural networks is pruning algorithm strategy to trim unnecessary connections or units of the network to reduce a large network to an appropriate size. Pruning is done by estimating the sensitivity of the total error of each weight in the network. Those weights or neurons, which are insensitive to error changes, will be eliminated at each step of training. Thus, pruning algorithms are useful in removing the redundant connections making the network smaller in size and better performance.

Researchers have suggested many pruning algorithms for optimizing the architecture of neural networks (Augasta & Kathirvalavakumar, 2013). Penalty or weight decay method adds a penalty term to minimize the objective function so smaller weights can be forced to zero. (Tin-Yau & Dit-Yan, 1997) proposed two strategies to tune the hidden layers, one is by adding penalty terms on the error function, in this way when one weight is updated, the updating rules update the other weights also. The second algorithm he proposed is Gauss Schmidt algorithm to determine the principal nodes in one epoch, and the weight connected to principal nodes are updated and other weights remain unchanged in same epoch.

Engelbrecht & Cloete (1996) proposed the variance nullity pruning (VNP) algorithm based on new measure called variance nullity based on average sensitivity of a network parameter over all the patterns.
Castellano, Fanelli, & Pelillo (1997) proposed an iterative pruning algorithm to find the most appropriate network by least squares identification algorithm. The output matrix may have deficient rank with infinite solutions with high norms in large network size.

Ponnapalli, Ho, & Thomson (1999) proposed pruning algorithm based on sensitivities of weights as Local Relative Sensitivity Index (LRSI) as the ratio of the sensitivity of a particular weight and their sum are connected to the same node from previous layer.

Abid, et al. (2001) proposed the modified version of Variance Nullity Pruning algorithm by pruning the nodes layer by layer with the use of a pruning decision based on local parameter variance nullity coefficient (LPW). They proposed the global SA method, as the Extended Fourier Amplitude Sensitivity Test (EFSAT) method, quantify the hidden units based on their ranking leads to suggestion of most favourable units to eliminate.

Hagiwara (2002) proposed three effective strategies as Goodness factor, Consuming energy and Weights power for detecting redundant weights and neurons. The proposed method is based on magnitude based pruning (MBP) methods assuming small weights are irrelevant.

Hong, Jigen, Zong-Ben, & Bo (2003) proposed a node pruning algorithm based on neural complexity of neural network. Neural complexity is estimated by acquiring standard covariance matrix of the neural network connection matrix during training. The algorithm doesn’t
need to train the cost function to local minimum leading to pruning of nodes before updating weights of neural network architecture.

*Huynh & Setiono (2005)* proposed the cross validation method in which pruning criteria is still based on previous pruning strategy but network is cross validated to test the pruned network.

*Lauret, Fock, & Mara, (2006)* proposed a new technique on global Sensitivity Analysis Model Output (SAMO) to obtain the number of hidden units of a single layer fully connected network.

*Zeng & Yeung (2006)* proposed a method to prune the hidden neurons of multilayer perceptron network using a quantified sensitivity measure. The method defines the sensitivity of an input neuron as the expectation of its output deviation with respect to all inputs and estimates the relevance of a neuron by summation of values of outgoing weights. Then, the algorithm prunes the neurons which are less relevant.

*Nicoletti and Bertini Jr.(2007)* proposed the penalty function which eliminates the unnecessary functions and prevents the weights of the connection from taking larger values. But this approach may eliminate weights which are useful for network architecture creating local minima on the error surface of training.

*Sabo & Xiao-Hua(2008)* proposed a new pruning algorithm which optimize the network by sensitivity analysis, variance sensitivity analysis and cross fold validation method.
Hong-Jie & Bao-Gang (2009) suggested two phase construction approach for pruning both input and hidden units of multi layered perceptron’s based on mutual information. All salient input units are determined according to their rankings and lower ranking units will be eliminated.

Paliwal (2009) conducted a comprehensive review of articles that involve a comparative study of feed forward neural networks and statistical techniques used for prediction and classification problems in various areas of speech applications.

Baopu Li. (2009) used the back propagation neural network to classify and detect the bleeding and ulcer in wireless capsule endoscopy images by chromaticity moments. Using the extracted features from wireless capsule endoscopy and trained multilayer perceptron (MLP), they computed the automatic computer aided disease detection in wireless capsule endoscopy images.

Kiranyaz, et al. (2009) proposed the evolutionary artificial neural networks by multi-dimensional particle swarm optimization. They used the multidimensional Particle Swarm optimization (PSO) technique, to seek the optimal position and dimension in multidimensional space without setting a fixed dimension as priori. The optimal dimension from the PSO process can be converged to ANN configuration to automatically tune the network parameter at optimal dimension. The experiments results showed that multi-dimensional particle swarm optimization (PSO) in ANN
configuration has better generalization capability compared to other techniques and able to rank network configurations from best to worst.

*Lai & Chang (2009)* proposed the clustering based approach using hierarchical evolutionary algorithm (HEA) for segmenting medical images. They proposed the hierarchical evolutionary algorithm based on conventional genetic algorithm by initializing population and evaluating the fitness function using the genetic operators to classify them into different levels of regions as segmented images. The experiments was conducted on skull based CT, abdominal MRI, brain MRI, knee MRI and a computer generated phantom image. The proposed method was compared with dynamic thresholding, k-means, fuzzy c-means and CHNN. Experimental results were evaluated using evaluation metrics like misclassified rate, segmented pixels, sensitivity and specificity.

Zhou, et al. (2009) used the artificial neural networks and genetic algorithms approach for multi objective optimization of material selection for sustainable products. Materials properties and selection is an important factor in product design and multi objective need to be optimized to develop better sustainable products.

Cheng, H.D. (2010) reported a survey of using on automatic breast cancer detection and classification using ultrasound images by ANNS and Support Vector Machines (SVM). They concluded that supervised learning provide higher classification accuracy compared to other learning methods.
using different features extracted using feature extraction and segmentation of ultrasound images.

Ahmadlou & Adeli (2010) proposed the enhanced probabilistic neural network with local decision circles as a robust classifier compared to the traditional back propagation and hybrid neural network. Probabilistic neural network does not require a separate training phase for tuning network parameters, like weights and hidden neurons based on machine learning rules. Rather, probabilistic neural network (PNN) assigns the rest of the data to a class with maximum likelihood compared to other classes. Likelihood function is computed using kernel density estimation as probability distribution function to find the maximum spread of probability density of particular class with respect to test data. PNN are enhanced by using local decision circles in the proposed method to incorporate local information and in homogeneity existing in the training population. The local decision is limited to decision circle radius to find the spread parameter in local rule of training data of each class to increase performance and accuracy in classification rate.


Ebrahimzadeh & Ranaee (2010) proposed an optimized neural network and efficient features extraction for pattern recognitions in control charts. They designed an efficient system for feature extraction and
classify using optimized neural network based on particle swarm optimization to improve the generalization.

*Khashman (2010)* conducted an investigation study of using back propagation neural network for credit risk evaluation system. A comparative study was done on credit application cases from the German credit approval datasets using neural networks and other learning schemes for better optimal performance in automatic processing of credit applications.

*Kenji Suzuki (2010)* used the massive training ANN (MTANN) as a filter with laplacianeigen function base dimensionality reduction for computed aided detection of polyps in CT colonography. The MTANN is trained with massive number of sub volumes extracted from input volumes together with teaching volumes containing the distribution for the “likelihood of being a polyp”. In order to make MTANN work more efficiently, they proposed a dimension reduction method for an MTANN by using Laplacian eigenfunctions (LAPs), denoted as LAP-MTANN.

*Korürek & Doğan (2010)* classify the electrocardiogram (ECG) beat based on particle swarm optimization (PSO) and radial basis function neural network (RBFNN). Four morphological features are extracted from each beat and classified by RBFNN structure evolved by Particle Swarm Optimization (PSO).

*Fangju (2011)* proposed the improved iterative Pruning algorithm (IIP) based on iteratively pruning the nodes of hidden layer in Feed-
Forward Neural Networks and adjusting the remaining weights using a Conjugate Gradient Precondition Normal Equation (CGPCNE) algorithm.

Feng, et al. (2011) proposed the ozone concentration forecast method based on genetic algorithm using genetic algorithm optimized back propagation neural network and support vector machines. To overcome the over fitting problem due to large datasets, they introduced a parallel computing structure to divide the huge data into different categories and build ANNs for each category. The clustered data categories were classified using support vector machines (SVM) and fed into optimized genetic algorithms neural network for further classification of each category of dataset. The predictions of SVM- GABPNN performed much better than back propagation neural network and genetic algorithms based ANNs.

Fernandez, et al. (2011) conducted a comprehensive review using optimized genetic algorithm based Bayesian neural networks and support vector machines in drug design. They concluded that genetic algorithm optimized parameters were more accurate and robust than previous published models on the same datasets and explained more than 65% of data variances in validation experiments. (G, G, D, & M, 2009) developed the hybrid model and optimized the surface roughness in electric discharge machining using artificial neural networks and genetic algorithms. As done in previous studies, features are extracted and classified using multiple
layer neural networks and genetic algorithm is used to optimize the weighting factors of network.

Aydin, et al. (2011) proposed a multi objective artificial immune algorithm for parameter optimization of Support Vector Machine (SVM). In the proposed model, the SVM parameters are optimized by implementing clonal selection algorithm and classify using these optimal values. The proposed algorithm is implemented for fault diagnosis of induction motors and detection of anomaly with higher optimization and accuracy.

Karaboga & Ozturk (2011) proposed Artificial Bee Colony (ABC) algorithm as a novel clustering approach and optimization algorithm based on the intelligent foraging behaviour of a honey bee swarm. They concluded that simulation results of ABC algorithm can be efficiently used for multivariate data clustering.

Li & Liu (2011) conducted the melt index prediction using MPSO-SA-RNN (modified PSO-SA algorithm and RBF neural network) prediction model. Melt Index is an important quality variable which determines the product specifications, and quality control of practical polypropylene polymerization process.

Min, et al. (2011) proposed a dynamic feed forward network for predictive control and tuning parameters are adjusted by Gaussian Particle Swarm Optimization (GPSO) in the training process.
Oh et al. (2011) proposed a polynomial based radial basis function neural networks (P-RBF NNs) neural network architecture based on fuzzy inference mechanism. The tuning parameters of neural network model (learning rate, momentum coefficient) are optimized by using particle swarm optimization. In this proposed model, the weights between the hidden layer and the output are polynomials weights which are useful in capturing the nonlinear nature of data encountered in regression or classification problems. The proposed classifier is applied to synthetic and machine learning datasets, and results are compared with reported previous studies.

Shen et al. (2011) used the radial basis function neural network (RBF) to train data and forecast the stock indices of Shanghai Stock Exchange. To optimize the tuning parameters of network, they introduce the artificial fish swarm algorithm (AFSA) and k means clustering optimized by AFSA in the learning process of RBF to increase forecast efficiency. This model results are compared with forecasting results of RBF optimized by genetic algorithms, particle swarm optimization (PSO), ARIMA and Support Vector Machines (SVM).

Turaga et al. (2011) presented a machine learning approach to computing an affinity graph using a convolutional network (CN) trained using ground truth provided by human experts. They used the back-propagation procedure to train the CN affinity graph with iterative
optimization procedure of stochastic gradient descent with diagonal scaling.

Qasem & Shamsuddin (2011) proposed the radial basis function network based on time variant multi objective particle swarm optimization for medical diseases diagnosis. They proposed an adaptive evolutionary radial basis function (RBF) network algorithm to evolve weights and connections of RBF networks simultaneously. The generalization and accuracy of the proposed model is validated using different benchmark medical diseases diagnosis data from UCI machine learning repository and compared with the previous literature.

Khashei, et al.(2012) proposed a novel hybrid classification model of artificial neural networks using multiple linear regression models for better generalization and classification accuracy. The proposed model is empirically compared with traditional classification model like Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), K-nearest neighbour(KNN) and Support Vector Machines (SVM) using benchmark and real world data sets for two class and multi class classification.

Leung, Sum, & Liu (2012) proposed an open node fault regularizer (ONFR) approach to train radial basis function (RBF) networks and to improve its generalization capability. They proposed a formula which is able to predict the generalization capability of faulty RBF networks without
using a test set or generating a large number of potential; faulty networks and to optimize the regularization parameter and RBF width.

Augasta & Kumar (2013) proposed the method which prunes the hidden nodes according to their significance and update all the weights of its outgoing connections known as Neural Network Pruning by Significance (N2PS).

Chen, et al. (2013) proposed an evolutionary design of constructive multilayer feed forward neural network constructed through several neural layers. The proposed model is based on using genetic algorithms to automatically fine tune the learning parameters like hidden layer, number of neurons and weights for better optimization in terms of performance and accuracy. Genetic algorithms are used to optimizer the configuration of the neural network with a set of parameters encoded using genetic coding. The transfer function of single neuron is adjustable rather than classic fixed transfer function which is highly suitable for modelling nonlinear system.

Han, et al. (2013) proposed the efficient self-organizing multilayer neural network for nonlinear system modelling. For making the ANNs self-organizing architecture, they proposed the axon neural network (AANN) to automatically change the number of hidden neurons and topologies of the neural network during the training process. To design the AANN architecture, they used the dynamic tuning strategy as adaptive connecting and pruning algorithm (ACP) to determine the number of
hidden neurons and layers. The experimental results showed that AANN have better performance than existing neural networks.

Ozyildirim & Avci (2013) proposed a new radial basis function neural network as generalized classifier neural network based on five layers. In this model, they added one layer of gradient descent based optimization of smoothing parameter approach and added a diverge effect for term calculation improvements. Diverge effect term is an improvement on summation layer calculation to supply additional separation ability and flexibility. Their experimental results depicted better classification performance of the proposed neural network compared to other neural networks.

3.3. Constructive Neural Networks

For better function approximation and pattern classification, many types of neural networks model have been proposed in past few years. To improve learning accuracy, generalization capability and training time of multilayered neural networks, the various tuning parameters of network architecture (number of hidden nodes and layers, weights, network topologies, activation function) need to be optimized and dynamically adjusted during training of network. Over fitting and under fitting takes place due to inappropriate neural network architecture. A number of ANN architectures and algorithms have been recently proposed in past few years, but adaptive and constructive Neural Networks (CONN) offer a higher optimization solution for finding the parametric parameters and
architecture of neural networks. Constructive algorithms start with small network and dynamically grow by adding and training neurons until optimal solutions is calculated. As in simple single layered feed forward neural networks, learning of network and training is done by modification of interconnection weights between neurons to minimize the error. But constructive algorithms find an optimal solution by automatically fitting the network size to the data without overspecializing yielding better generalization and classification for pattern recognition problems. The advantages of using constructive algorithms compared to other neural networks are

1. During training, the topology of network can be dynamically determined
2. On non-contradictory finite datasets, they deliver convergence to zero classification errors.
3. During training process of constructive algorithms, elementary thresholds neurons are implemented.
4. The architectural size of neural network can be restricted for reducing complexity and increasing generalization and classification accuracy.
5. Learning parameters need not to fine-tune and searching of initial weights as in constructive algorithms, they are randomly assigned.
Figure 3.1: Development Architectures of Constructive Neural Networks.

As can be seen in figure 3.1, Circles are representing units and lines are representing connection weights. Direction of activation flow is represented by thick arrow and trainable connection weights are represented by dashed lines. Fig 3.1(a) depicts the first output phase of training. Fig3.1 (b) shows the first input phase in which candidate hidden units are trained so that their outputs co-vary with network error. Fig3.1 (c) depicts the second output phase, in which the best correlating candidate hidden unit is installed in the network. Finally in Fig3.1 (d), a second hidden unit is installed downstream of the first hidden unit as third output phase.

3.3.1. Formulation of Constructive Neural Networks

Constructive neural network is based on feed forward neural network to approximate a regression function whose input vector is indicated by multi-dimensional vector X and its output is expressed by
scalar Y, without loss of any generality. A regression surface function is represented as relationship between X and Y. A FNN is trained to realize the relationship with input samples denoted by \((x_1, x_2, x_3, \ldots, x_p)\) and output samples of each layer L as \((y^1_1, y^1_2, y^1_3, \ldots, y^1_{L-1}, y^1_L)\) where \(j = (1, \ldots, P)\) and corresponding target samples are denoted by \((d_1, d_2, \ldots, d_p)\) which is the output data contaminated by a additive white noise vector \(\mu = (\varepsilon^1, \varepsilon^2, \varepsilon^3, \ldots, \varepsilon^p)\), where \(L - 1\) is the number of hidden layers, \(L\) is the number of output layer, and \(p\) is the number of patterns in the data set. The network training is formulated using unconstrained least square (LS) nonlinear optimization technique

\[
\min_{L,n,f,w} \sum_{j=1}^{p} (d^j - y^j_L)^2
\]  

subject to

\[
y^j_1 = f_1(w^j_1 x^j), \quad w^j_1 \in \mathbb{R}^{n^1 \times M}, y^j_1 \in \mathbb{R}^{n^1}, x^j \in \mathbb{R}^M \tag{2}
\]

\[
y^j_2 = f_2(w^j_2 x^j), \quad w^j_2 \in \mathbb{R}^{n^2 \times M}, y^j_2 \in \mathbb{R}^{n^2} \tag{3}
\]

\[
y^j_L = f_L(w^j_L x^j), \quad w^j_L \in \mathbb{R}^{n^1 \times n_{L-1}}, y^j_L \in \mathbb{R}^1 \tag{4}
\]

Where \(n = (n_1, n_2, \ldots, n_{L-1})\) is a vector denoting the number of units in each hidden layers as hidden nodes, \(f = (f_1, f_2, \ldots, f_L)\) denoted the activation function of each layer and \(w = (w_1, w_2, \ldots, w_L)\) denotes the weight matrix of each layer. For tuning certain variables, least square optimization is used to tune the weights, hidden nodes and layers in constructive neural network for better optimized solution. For weight selection during training, various recursive methods are used as Quasi
Newton or Back Propagation methods to find the local minima and solution of the problem. But changes in error surface function each time with modified activation functions, resulting different local minima in each time.

The well-known CONN algorithms for two class classifications are the Tower and Pyramid (Gallant, 1986), the Tiling (Mazard and Andal, 1989), the Upstart (Fren, 1990) and Perceptron Cascade algorithm (Burgess, 1994). There are further improvements of CONN algorithms done by minimization of classification of errors (Irregular partitioning algorithm (Marchand et al, 1990), the Carve algorithm (Young and Downs, 1998), the Target and Switch algorithm (Campell and Vicente, 1995), the Oil Spot algorithm (Mascioli and Martinelli, 1995), the Constraint based decomposition algorithm (Draghici, 2001), the Decomposition Algorithm for Synthesis and generalization (Subiratset al, 2008) and Dynamic node Creation algorithm (Ash, 1989).

*Kwok & Yeung (1997)* conducted a comprehensive study on using constructive algorithms in feed forward network for regression problems. Constructive algorithms follow the approach to search for a good network in the other direction by starting with a small network and then add additional hidden units and weights until a satisfactory solution is found.

*Parekh et al. (2000)* proposed two constructive learning algorithms as M-Pyramid-real and M-Tiling-real that extend classic pyramid and tiling algorithms, for learning real to M-ary mappings. The proposed method was
proved by different pattern classification problems for better generalization and accuracy.

Ma & Khorasani (2004a) proposed a new technique for facial expression recognition which uses the two dimensional discrete cosine transform (DCT) over the entire face as feature detector and constructive feed forward neural network as facial expression classifier. Confusion matrices are calculated in both network training and generalization with best recognition rates of 100% and 93.75% for training and generalized images. For better generalization and classification accuracy, the input side weights of the constructed networks are pruned by 30% using pruning method.

Constructive and Pruning Neural network provides better generalization and convergence, but they have set of limitations too. It is still a hurdle to stop constructive neural network when to stop the addition of hidden neurons. Secondly, constructive neural networks are sensitive to initial conditions and get trapped in local minima. On the other side, pruning neural networks suffered from high computational time as majority of the training time is spend on large architectural neural networks. In pruning network, it is also difficult to find fitness threshold to stop pruning with smaller optimize size of network. Thus, to amend their limitations, several researchers proposed the hybrid approach of adaptive constructive and pruning neural networks. It executes the constructive algorithms by
setting up the large network and pruning after the implementation reduce to optimized smaller size.

Ma & Khorasani (2004b) proposed the new training strategies for constructive neural network applying to regression problems. They proposed the constructive feed forward neural network, in which input and output side training will be separated in order to reduce the training time. Error signal is scaled and input side reluctant connections are removed by pruning methods to acquire better generalization performance and training efficiency.

Hsu, (2008) proposed an adaptive growing-and-pruning neural network control (AGPNNC) system for a linear piezoelectric ceramic motor. The neural controller uses a self-constructing neural network (SCNN) to mimic an ideal computation controller, and the robust controller is designed to achieve L2 tracking performance with desired attenuation level. If the approximation performance of the SCNN is inadequate, the SCNN can create new hidden neurons to increase learning ability. If the hidden neuron of the SCNN is insignificant, it should be removed to reduce computation loading; otherwise, if the hidden neuron of the SCNN is significant, it should be retained.

Narasimha et al. (2008) conducted a comparative study on integrated growing pruning method for feed forward neural network training. They concluded that networks designed using the growing
pruning method have less training and validation error than growing and pruning alone using different datasets and benchmarks.

*Islam, et al. (2009)* presented a new constructive algorithm (NCA) to automatically determine ANN architectures by architecture adaptation and function adaptation. Constructive approach are used to determine number of hidden neurons and layers and trained using different training datasets to optimize the whole algorithm. They concluded that NCA perform better generalization capability compared to existing constructive and non-constructive algorithms.

*Islam, et al. (2009)* proposed a new algorithm as adaptive merging and growing algorithm (AMGA), similar to NCA described before by reducing the amount of retraining after modifying ANN architecture by pruning and merging correlated hidden neurons.

*Islam et al., (2009)* proposed a new algorithm, called adaptive merging and growing algorithm (AMGA), for artificial neural networks (ANNs) architecture. This algorithm merges and adds hidden neurons during the training process of ANNs. The merge operation introduced in AMGA is a kind of a mixed mode operation, which is equivalent to pruning two neurons and adding one neuron. The adaptive strategy merges or adds hidden neurons based on the learning ability of hidden neurons or the training progress of ANNs. In order to reduce the amount of retraining after modifying ANN architectures, AMGA prunes hidden neurons by merging
correlated hidden neurons and adds hidden neurons by splitting existing hidden neurons.

Xinjian, et al.(2010) conducted an overview survey of Growing Neural Network (GNN) as constructive neural network to optimize the number of hidden neurons and layers with network topologies. (Puma-Villanueva, dos Santos, & Von Zuben, 2012) presented a constructive algorithm based on constructive and pruning mechanism and adjusting the synaptic weights by quasi newton method to form a hybrid model for better generalization for linear/nonlinear classification problems.

Zhen, et al.(2010) conducted a brief survey on some classical constructive neural network development by reviewing their topological structure and learning features. The survey also discusses about new development in constructive neural networks for betting generalization and learning accuracy.

Subirats, et al. (2010) proposed the new C-Mantec (Competitive majority Network Trained by Error Correction) algorithm for compact neural network architectures incorporating new features like competition between neurons and built in filtering stage of noisy examples. For multi Category pattern classification, they follow three approaches as One against All (OAA), One against One(OAO) and P against Q(PAQ). The algorithm was compared with standard classification algorithm using different size benchmarks.
Huynh & Reggia (2011) proposed the modified error back propagation training process, which learns a different hidden layer representation of input patterns than would normally occur. They concluded that modifying error back propagation technique effectively separates learned pattern encodings in the hidden layer to improve contemporary methods.

Subirats, et al. (2012) proposed a novel constructive neural network that combines competition between neurons with a stable modified perceptron learning rule. The new C-Mantec algorithm make use of thermal perceptron learning rule that grow using constructive neural network mechanism, while neurons follow the completion strategy by learning through incoming information to their stored knowledge with newly added neurons. The C-Mantec algorithm is implemented to wide range of classification problems, using stable learning and competitions between neurons to prevent over fitting with compact architectures and good generalization capabilities.

3.4. Cascading Neural Networks

Bodyanskiy, at al. (1993) also proposed a new hybrid cascade wavelet neural network and its learning algorithm in batch and online. They replaced the wavelet neurons with Rosenblatt perceptron in output neurons of cascade correlation neural networks.

Prechelt (1997) conducted a investigation study of Cascade Correlation neural network (CCNN) and its various learning algorithms.
The comparative study involved experiments using various CCN on several datasets to figure out the algorithms with better generalization and convergence maximization.

*Thivierge, et al. (2003)* introduced an algorithm for performing simultaneous growing and pruning of cascade correlation (CC) networks which add hidden units and removes unimportant connections by using Optimal Brain Damage (OBD) algorithm in both input and output phases of Cascade Correlation neural network. OBD was adapted to prune weights using two separate objective functions to train hidden units and cascade correlation neural network.

*García & Romo (2009)* conducted a comparative study for rockfill strength evaluation using cascade correlation networks. Proper design of rockfill embankments is not straightforward because the strength and compressibility of the rockfill materials are hard to determine. In this study, they used the cascade correlation neural networks to evaluate the principal effective stress failure from known parametric values.

*Gao, et al. (2009)* proposed the fused clonal selection algorithm (CSA) and differential evolution (DE) method for training cascade correlation neural network. They demonstrate that using this effective hybrid optimization approach, parameters can be fine-tuned with better convergence and computational complexity.

*Li-ying & Wei-guo (2009)* implemented the cascade correlation neural network algorithm for finding the energy characteristics of hydraulic
turbine of Waniazhai station. They used the cascade correlation to establish the model of energy characteristics, and the relationship curve between hidden neurons and output neurons with better efficiency and performance and compared with previous neural networks.

Zhuoyong & Harrington (2009) proposed the temperature constrained cascade correlation networks (TCCCNs) to identify powdered rhubarbs based on their near infrared spectra. With optimized cascade correlation neural network training parameters, rhubarb powdered samples were classified based on their spectra with 100% accuracy.

Evans, et al. (2010) proposed a sibling descendant cascade correlation neural network model for toddlers transitions on nonverbal false belief tasks involving a novel location. Their proposed model reproduced the transition that has been observed in older children and generalized its learning to a novel location.

Zhao & He, (2010) conducted a study of using cascade correlation neural network to monitor faulty quality categories of the products in manufacturing process. They concluded that cascade correlation neural network outperforms the classic back propagation neural network in better performance and accuracy.

Riley, et al. (2010) proposed a novel method based on subdivision called as patch working to improve the performance of cascade correlation neural networks on multimodal functions. To improve the fitness of cascade neural network, they used three methods as early stopping,
ensemble averaging and patch working to reduce the error and better generalization.

*Singh & Karnan, (2010)* proposed an intelligent approach by taking the User Profile History (UPH) to reduce the location update cost by using Cascade Correlation Neural Networks. Their results depict that the efficiency of UPH significantly reduced the cost of locations updates and call delivery procedures compared to various other strategies.

*Sharma & Chandra (2010)* proposed an adaptive slope sigmoidal function for cascade correlation constructive neural networks. To achieve an adaptive slope sigmoidal function during learning, the slope is adapted to define optimal value of slope parameter of sigmoidal function for nonlinear nodes.

*Riley et al., (2010)* proposed the improved performance of cascade correlation neural networks using different multimodal functions. They proposed three techniques to improve generalization as early stopping, ensemble averaging and patch working.

*Beno, et al. (2011)* proposed the a novel approach for estimation the rotor position of a Switched Reluctance Motor (SRM) drive system using cascade correlation neural networks. Using CCNN, they can evaluate rotor position by measuring the three phase voltages and currents using magnetic characteristics of SRM. The experimental results showed greater accuracy and successful rotor position estimation.
Ghate & Dudul (2011) developed the radial basis function multilayer perceptron cascade neural network for detecting the fault for three phase induction motors. During the training, simple statistical parameters of static current are considered as input vectors and Principal Component Analysis is used to select suitable inputs to the network. Experiments results showed robustness to the uniform and Gaussian noises with better classification accuracy.

Li & Li (2011) proposed cascade correlation algorithm with trainable activation functions expressed as primitive functions with lower order derivatives. They defined the cascade correlation architecture as parallel and series constructed schemes of the activation functions. The efficiency of algorithm was demonstrated with two spiral classification and Mackay Glass time series prediction problem.

Rung-Ching, et al.(2011) used the cascade correlation neural networks for an indoor position location system, and provide location service for user. They collect the global positioning system (GPS) of reference point to train the hybrid neural network models and train the cascade correlation network model to track the location of object.

Hodnett & Hsieh (2012) applied cascade correlation neural networks in stock selection models for global equities. They train the output side weights via back propagation learning rule and extended Kalman filter learning rule until mean squared error (MSE) is minimized.
(Yan Xia & Wei, 2009) proposed a new text categorization method using Singular Value Decomposition (SVD) and Cascade Correlation (CC) neural networks. They use the SVD to represent relations between large number of words and natural text passages rather than using Vector Space Model (VSM) and classify further using Cascade Correlation neural networks.

Nassif, et al. (2012) proposed a novel model to predict software effort estimation in the early stages of software life cycle using cascade correlation neural network approach. The proposed model was estimated using MMER and PRED metrics on 214 industrial and 26 educational projects and compared with multiple linear correlation and use case point model.

Diamantopoulou, (2012) conducted a comparative study to assess a reliable modelling approach for estimating sustainability of forests using Cascade Correlation Neural Networks (CCNN). They concluded that CCNN models are superior to other neural network models for better estimation of tree sustainable parameters.

Gao, et al. (2012) proposed a novel constructive orthogonal least squares algorithm for training cascade neural networks. During training of hidden units, they used the orthogonal least squares (OLS) method to derive objective function by which sum of squared errors (SSE) can be maximally reduced for less hidden units and better generalization performance. The simulation results showed that proposed algorithm leads
to smaller networks with better generalization performance compared to classic cascade correlations algorithms and its variants. (Balázs) conducted a brief survey on cascade correlation neural networks and its applications.

*Chunpeng, et al. (2012)* proposed the cascaded heterogeneous convolutional neural networks for handwritten digit recognition. To reach the certain performance compared with networks built from same type, heterogeneous cascaded convolutional neural network can reduce the number of parameters for tuning.

*Kannan & Rathinam (2012)* presented a new technique based on wavelet transform (WT) and artificial neural networks for detecting high impedance faults in a real distributed networks. The classification results are compared using six neural networks as back propagation network, cascade correlation network etc. The results showed cascade correlation neural network perform better classification accuracy compared to other networks.

*Tengfei & Dingyun (2012)* proposed an extension of A CasPer, as Layered CasPer by constructing an architecture of hidden neurons formed as layers and there are no connections between neurons in the same layer. New neurons were added beside previous neurons to form a new layer. Each new neuron receives a connection from network’s original inputs, but not output from previous neurons. The algorithm produces similar generalization and classification results as A_CasPer, but due to its layered architecture, it is useful for big datasets with large inputs.
Borenovic, et al. (2013) investigated the vehicle location algorithm based on the received signal strength (RSS from Global System for Mobile Communications (GSM) networks. The performances are compared using cascade correlation neural network and other multilayer artificial neural network architectures. The proposed positioning algorithm outperforms KNN and extended Kalman Filter (EKF) neural network architecture trained models.

Nicoletti et al. (2013) proposed the constructive neural networks for online prediction of the feeding phase in high cell density cultivation of E.coli. The study investigated the data and methodology used for different types of CONNs and showed that among training of all different neural networks, cascade correlation neural network shows higher performance and better generalization.

Lashkarbolooki, et al. (2013) proposed the modelling of spearmint oil extraction in a packed bed using SC-Co\textsubscript{2} trained using cascade forward back propagation neural network. The proposed cascade correlation neural network consisted of one hidden layer and trained using Levenberg-Marquardt algorithm to correlate the obtained extraction yield of spearmint oil in a wide range of operations.

3.5. Extreme Machine Learning

Huang et al. (2006) proposed a new novel algorithm as Extreme Machine Learning (ELM) for single hidden layer feed forward neural network which has less computational time and faster speed even on large
datasets. The main working principle of ELM is random initialization of weights rather than learning through slow process via iteratively gradient based learning as backpropagation \cite{Huang2006}. In Extreme machine learning, the number of hidden nodes and their weights are randomly assigned, which distinguishes the linear differentiable between the output of hidden layer and output layer. The output weights can be determined by linear least square solution of hidden layer outputs through activation function and the data samples targets. This type of ELM tends to have small training error, good generalization performance, and high computational speed and tuned hidden layers parameters randomly.

\textit{Huang et al. (2006)} proposed a new novel algorithm as extreme learning machines (ELM) for single hidden layer feed forward neural networks (SLFNs) which randomly choose hidden nodes and determine the output weights of SLFNs. ELM improves the disadvantages of SLFNs where networks parameters are iteratively tuned and gradient based learning algorithms are slow in training neural networks. Using this ELM, input weights and hidden layer biases of SLFNs can be randomly assigned if the activation functions in the hidden layer are infinitely differentiable. After the random assignment of input weights and hidden layer biases, SLFNs can be considered as linear system and the output weights of SLFNs can be analytically determined through simple generalised inverse operation of the hidden layer output matrices. The proposed ELM tends to
have small training error, good generalization performance, and high computational speed and tuned hidden layers parameters randomly.

*Huang, et al. (2008)* proposed an incremental extreme learning machine (I-ELM), which randomly adds hidden nodes incrementally and analytically determines the output weights. Although hidden nodes are generated randomly, the network constructed by I-ELM remains as a universal approximator. This study extends I-ELM from the real domain to the complex domain and showed that, as long as the hidden layer activation function is complex continuous discriminatory or complex bounded nonlinear piecewise continuous, I-ELM can still approximate any target functions in the complex domain. The universal capability of the I-ELM in the complex domain is further verified by two function approximations and one channel equalization problems.

*Rong, et al. (2008)* addressed the architectural design of the ELM classifier network, since too few/many hidden nodes employed would lead to under fitting/over fitting issues in pattern classification. In particular, they described the proposed pruned-ELM (P-ELM) algorithm as a systematic and automated approach for designing ELM classifier network. P-ELM uses statistical methods to measure the relevance of hidden nodes. Beginning from an initial large number of hidden nodes, irrelevant nodes are then pruned by considering their relevance to the class labels. As a result, the architectural design of ELM network classifier can be automated. Empirical study of P-ELM on several commonly used
classification benchmark problems and with diverse forms of hidden node functions show that the proposed approach leads to compact network classifiers that generate fast response and robust prediction accuracy on unseen data, compared with traditional ELM and other popular machine learning approaches.

Wang, et al. (2008) proposed an Extreme Learning Machine (ELM) based protein secondary structure prediction framework which can provide good performance at extremely fast speed. To achieve better performance, the three secondary structures are independently predicted by a binary ELM classifier first; and a probability based combination (PBC) method is then proposed to combine these binary prediction results into the expected three-classification results. Helix post processing (HPP) method is finally proposed to further improve the overall performance of the framework based on biological features. Experiments conducted on the real data sets CB513 and RS126 demonstrated proposed algorithm can achieve as good prediction accuracy as other popular methods at very fast learning speed.

Lan, et al. (2009) propose an ensemble of online sequential extreme learning machine (EOS-ELM) based on OS-ELM. The results show that EOS-ELM is more stable and accurate than the original OS-ELM.

Tang & Han (2009) mentioned that there are two problems preventing the further development of extreme learning machine (ELM). First, the ill-conditioning of hidden layer output matrix reduces the stability of ELM. Second, the complexity of singular value decomposition (SVD)
for computing Moore–Penrose generalized inverse limits the learning speed of ELM. For these two problems, there work proposes the partial Lanczos ELM (PL-ELM) which employs the hybrid of partial Lanczos bidiagonalization and SVD to compute output weights. Experimental results indicate that, compared with ELM, PL-ELM not only effectively improves the stability and generalization performance but also raises the learning speed.

*Lan, et al. (2010)* attempted to address the architectural design of ELM regress or by applying a constructive method on the basis of ELM algorithm. After the nonlinearities of ELM network are fixed by randomly generating the parameters, the network will correspond to a linear regression model. The selection of hidden nodes can then be regarded as a subset model selection in linear regression. This constructive hidden nodes selection for ELM (referred to as CS-ELM) selects the optimal number of hidden nodes when the unbiased risk estimation based criterion CP reaches the minimum value. A comparison of the CS-ELM with other model selection algorithms of ELM is evaluated on several real benchmark regression applications and the empirical study shows that CS-ELM leads to a compact network structure automatically.

*Li, et al. (2010)* proposed a structure-adjustable online learning neural network (SAO-ELM) based on the extreme learning machine (ELM) with quicker learning speed and better generalization performance. Firstly, ELM is changed into a structure-adjustable learning machine, in
which the number of nodes in its single hidden layer can be adjusted. Then, a special strategy is developed to handle the difficulty that the new added hidden nodes outputs corresponding to the discarded training data cannot be obtained. After that, an iterative equation is presented to update the output matrix when hidden nodes are added. Results of numerical comparison based on data from the real world benchmark problems and an actual continuous casting process show that the performance of SAO-ELM has significant advantages over that of the typical online learning algorithms on generalization performance. In addition, SAO-ELM retains the merit of quick learning characteristic of ELM.

*Yoan et al. (2010)* proposed the optimally pruned extreme leaning machine (OP-ELM) algorithm which is an extension of original ELM algorithm with pruning of neurons using ranking multi-response sparse regression (MRSR) method to design optimal neural architecture removing irrelevant variables.

*Cho & White (2011)* proposed the Wald ELM (WELM) test statistic to easily compute and has a large-sample standard chi-squared distribution under the null hypothesis of correct specification. They provided associated theory for time-series data to affirm their hypothesis with certain Monte Carlo experiments. The goals of the WELM test was straightforward, to seek a test powerful against essentially arbitrary alternatives that is simple to compute and has a standard asymptotic null distribution for the test statistic that is not model dependent. Computation of the WELM statistic
is convenient because it computes the associated function integrals using ELM methods.

*Deng, et al. (2011)* proposes a novel algorithm for automatic construction of a nonlinear system model based on the extreme learning machine. This is achieved by effectively integrating the ELM and leave-one-out (LOO) cross validation with our two-stage stepwise construction procedure. The main objective is to improve the compactness and generalization capability of the model constructed by the ELM method. Numerical analysis shows that the proposed algorithm only involves about half of the computation of orthogonal least squares (OLS) based method. Simulation examples are included to confirm the efficacy and superiority of the proposed technique. It is convenient and effective to solve nonlinear problems with a model that has a linear-in-the-parameters (LITP) structure. However, the nonlinear parameters (e.g. the width of Gaussian function) of each model term needs to be pre-determined either from expert experience or through exhaustive search. An alternative approach is to optimize them by a gradient-based technique (e.g. Newton's method). Unfortunately, all of these methods still need a lot of computations. Recently, the extreme learning machine (ELM) has shown its advantages in terms of fast learning from data, but the sparsity of the constructed model cannot be guaranteed.

*Huang et al.(2011)* presented a comprehensive survey on extreme learning machines and its applications. The study reviewed the basic ELM and its variants such as online sequential ELM (OS-ELM), Incremental
ELM (I-ELM), Pruning ELM, Constructive ELM and SVM with ELM feature mapping. The study concluded that ELM and its variants provide better generalization and convergence compared to conventional learning algorithms in different applications.

Wang, et al. (2011) discussed the effectiveness of ELM and proposed an improved algorithm called effective Extreme Learning Machine (EELM) that makes a proper selection of input weights and bias before calculating the output weights. The experiments results showed that the learning speed of EELM is generally faster than ELM with robust accuracy and generalization.

Miche, et al. (2011) proposed the optimally pruned extreme machine learning (OP-ELM) as wrapper methodology around the extreme learning machine (ELM) to reduce the sensitivity of the ELM to irrelevant variables. The proposed algorithm uses cascade of two regularization penalties: L₁ penalty to rank the neurons in the hidden layer, and L₂ penalty on the regression weights for numerical stability and effective pruning of neurons.

Miche et al. (2011) proposed an improvement of the optimally pruned extreme learning machine (OP-ELM) in the form of a L₂ regularization penalty applied within the OP-ELM. The OP-ELM originally proposes a wrapper methodology around the extreme learning machine (ELM) meant to reduce the sensitivity of the ELM to irrelevant variables and obtain more parsimonious models thanks to neuron pruning.
The proposed modification of the OP-ELM uses a cascade of two regularization penalties: first a $L_1$ penalty to rank the neurons of the hidden layer, followed by a $L_2$ penalty on the regression weights (regression between hidden layer and output layer) for numerical stability and efficient pruning of the neurons. The new methodology is tested against state of the art methods such as support vector machines or Gaussian processes and the original ELM and OP-ELM, on 11 different data sets; it systematically outperforms the OP-ELM (average of 27% better mean square error) and provides more reliable results – in terms of standard deviation of the results – while remaining always less than one order of magnitude slower than the OP-ELM.

The optimally pruned extreme learning machine (OP-ELM) was proposed to attempt to solve the problem that ELM faces with irrelevant (or highly correlated) variables present in the data set that can corrupt some of the neurons. The idea is to build a wrapper around the original ELM, with a neuron pruning strategy. First comes a ranking of the neurons by a least angle regression (LARS in practice the MRSR implementation of LARS is used for it also applies to multi-output cases), which sorts them by their usefulness regarding the output. And then a leave-one-out criterion is used to determine how many of the sorted neurons should be kept for the final OP-ELM model structure. The OP-ELM was proposed in the first place as a wrapper around ELM to improve its robustness by adding a neuron pruning strategy based on LARS ($L_1$ penalty) and leave-one-out (LOO).
Here the LOO criterion is modified to add a $L_2$ penalty (Tikhonov regularization) to the estimate, in order to regularize the matrix computations and hence make the MSE computation more reliable. The modified OP-ELM (TROP-ELM), therefore, uses “in cascade” $L_1$ and $L_2$ penalties, avoiding the large computational times problems commonly encountered when attempting to intertwine the two penalties (as in the elastic net or the composite absolute penalty approaches).

The TROP-ELM shows better performance than the original OP-ELM; with an average of 27% better MSE for the considered data sets (and improvements between 0% and 96% over the data sets used). Also notable is the decrease of the standard deviation of the results over the multiple repetitions for each data set, illustrating that the regularization introduced has a visible effect. In the end, the TROP-ELM performs rather similarly to the Gaussian processes on more than half the data sets tested, for a computational time which remains two to three orders of magnitude below and less than an order of magnitude slower than the OP-ELM, in the worst case among the data sets used.

Martínez et al. (2011) proposed a new strategy to prune the ELM networks using regularized regression methods to acquire optimal tuned parameters. The algorithm can acquire optimal tuned parameters by identifying the degree of relevance of the weights that connects the $k^{th}$ hidden element with the output layer using lasso and ridge regression.
Regularized version of least squares regression with several penalties on coefficient vector are used to remove the irrelevant or low relevance hidden nodes to achieve compact neural networks.

Mohammed, et al. (2011) proposed a new human face recognition algorithm based on two dimensional bidirectional principal component analysis (B2DPCA) and extreme learning machine (ELM). The human faces are decomposed using curvelet image decomposition and reduced dimensionally using an improved dimensionality reduction technique and discriminative feature sets are generated using B2DPCA and classified using extreme learning machines. Using this proposed method, there was significant improvement in classification rate, better generalization and less training time. Extensive experiments were performed using challenging databases and results were compared against other techniques.

Frénay & Verleysen (2011) proposed a new parameter-insensitive kernel inspired from extreme learning used for non-linear SVR. Hence, the practitioner has only two meta-parameters to optimise. This approach reduces significantly the computational complexity, and experiments show that it yields performances that are very close from the state-of-the-art. Unlike previous works which rely on Monte-Carlo approximation to estimate the kernel, this work also showed that the proposed kernel has an analytic form which is computationally easier to evaluate. This paper proposed a new asymptotic view for the ELM kernel when the number of units grows to infinity. In that limit case, this paper also shows that the
proposed kernel has an analytical form under certain assumptions on the hidden units of the extreme learning machine.

Fernández-Navarro, et al. (2011) proposed a methodology for training a new model of artificial neural network called the generalized radial basis function (GRBF) neural network. This model is based on generalized Gaussian distribution, which parameterizes the Gaussian distribution by adding a new parameter $\tau$. The generalized radial basis function allows different radial basis functions to be represented by updating the new parameter. For example, when GRBF takes a value of $\tau=2$, it represents the standard Gaussian radial basis function. The model parameters are optimized through a modified version of the extreme learning machine (ELM) algorithm. In the methodology proposed (MELM-GRBF), the centers of each GRBF were taken randomly from the patterns of the training set and the radius and $\tau$ values were determined analytically, taking into account that the model must fulfil two constraints: locality and coverage. A thorough experimental study is presented to test its overall performance. Fifteen datasets were considered, including binary and multi-class problems, all of them taken from the UCI repository. The MELM-GRBF was compared to ELM with sigmoidal, hard-limit, triangular basis and radial basis functions in the hidden layer and to the ELM-RBF methodology proposed by Huang et al. (2004). The MELM-GRBF obtained better results in accuracy than the corresponding sigmoidal, hard-limit, triangular basis and radial basis functions for almost
all datasets, producing the highest mean accuracy rank when compared with these other basis functions for all datasets.

Wang, et al. (2011) studied the image deblurring problem from a brand new perspective classification. They generalize the traditional PDE model to a more general case, using the theories of calculus of variations. Furthermore, inspired by the theories of approximation of functions, they transform the operator-learning problem into a coefficient-learning problem by means of selecting a group of basis, and build a filter-learning model. Based on extreme learning machine (ELM, an algorithm is designed and a group of filters are learned effectively. Then a generalized image deblurring model, learned filtering PDE (LF-PDE), is built. Image deblurring is a basic and important task of image processing. Traditional filtering based image deblurring methods, e.g. enhancement filters, partial differential equation (PDE) etc., are limited by the hypothesis that natural images and noise are with low and high frequency terms, respectively. Noise removal and edge protection are always the dilemma for traditional models. The experiments verify the effectiveness of our models and the corresponding learned filters. It is shown that proposed model can overcome many drawbacks of the traditional models and achieve much better results.

Zhang & Zhang (2011) proposed a new approach to the optimal control with constraints to achieve a desired end product quality for nonlinear processes based on new kernel extreme learning machine
(KELM). In existing ILC algorithm, the model was built only between manipulated input variables U and output variables Y without considering the state variables. However, the states variables Xstate are important in the industrial processes, which are usually constrained. In this study, the variables are divided into state variables Xstate, manipulated input variables U and output Y in the process of modelling. Then DU can be obtained by batch-to-batch iterative learning control separately. Kernel algorithm is added to ELM. Constraints of state variables Xstate and the input variables U are considered in the current version. PSO is used to solve the optimization problem and Kernel trick is introduced to improve accuracy of ELM modelling. The particle swarm optimization algorithm is used to search for the optimal value based on the iterative learning control (ILC). The proposed approach has been shown to be effective and feasible by applying bulk polymerization of the styrene batch process and fused magnesium furnace.

Zong & Huang (2011) applied the extreme learning machine for face recognition systems as multi class applications. For effective representation of faces for classification, they defined face representation in two categories as holistic based and featured based methods. The holistic features take the whole face region as raw inputs whereas feature based methods extract the local features as eyes, nose, mouth and their statistical correlation. To transom the feature data from high dimensional space to low dimensional space, they used different dimensionality reduction
algorithms such as principal component analysis (PCA), linear discriminant analysis (LDA) etc. The features vectors are fed into different classifiers as SVM, ELM etc. They concluded that ELM performs better results compared to other classifier with better optimized networks.

Yuan, et al. (2011) proposed the solution for regression problem as optimized approximation based on extreme learning machine. As in previous studies, it had been proved that ELM can achieve better results than support vector machine (SVM) in regression and classification. For optimal conditions of ELM, they reformulate the hidden matrix as Moore-Penrose inverse of hidden matrix H having at least one solution rather than no solution in most cases.

Leung et al. (2012) has also proposed an online sequential learning algorithm called online sequential extreme learning machine (OS-ELM), which can learn the data one-by-one or chunk-by-chunk with fixed or varying chunk size. It has been shown that OS-ELM runs much faster and provides better generalization performance than other popular sequential learning algorithms.

Avci & Coteli (2012) proposed an automatic system for target recognition of echo signals of high resolution range (HRR) radars by wavelet extreme learning machine. The study combines the combination of feature extraction and classification of echo signals waveforms by using X band pulse radars. The experimental results showed better generalization
performance much faster than conventional learning algorithms for feed forward neural networks.

*Cao, et al. (2012)* proposed a novel extreme learning machine as voting based extreme learning machine (V-ELM) for classification problems. In V-ELM, several individuals ELMs with the same number of hidden nodes and the same activation functions in each hidden nodes are used. The final class label is determined by majority voting on all the results obtained by these independent ELMs showed higher classification results compared to old ELM.

*Liu, et al. (2012)* conducted a comparative analysis of support vector machines and extreme learning machines from two viewpoints that are different from previous works: one is the Vapnik–Chervonenkis (VC) dimension, and the other is their performance under different training sample sizes. It is shown that the VC dimension of an ELM is equal to the number of hidden nodes of the ELM with probability one. Additionally, their generalization ability and computational complexity are exhibited with changing training sample size. ELMs have weaker generalization ability than SVMs for small sample but can generalize as well as SVMs for large sample. Remarkably, great superiority in computational speed especially for large-scale sample problems is found in ELMs. The conclusions can be reached that (i) unlike the VC dimension of the SVM, the VC dimension of the ELM is always limited and is equal to the number of hidden neurons with probability one, which makes it relatively easy to
construct an ELM model with the best VC dimension; (ii) the ELM has poorer generalization ability than the SVM when the size of the training set is small, while it has the potential to yield generalization behaviour as good as the SVM when the size of the training set becomes large, the possible reason for which is the presence of over fitting in the training process of the ELM; (iii) the ELM has superior computational speed compared with the SVM, and this superiority will increase drastically as the size of training set grows; and (iv) as an additional result, the LSSVM shows very strong learning ability, avoiding over fitting, as well as strong generalization performance, but high computational complexity is observed from the examples studied for this algorithm. All these results may provide a possibility for obtaining deeper understanding on the relationship and difference between the SVM and the ELM, and can also serve as possible indication for selecting an appropriate tool from the SVM, ELM, and LSSVM in practical applications.

Romero & Alquézar (2012) conducted a comparative study between error minimized extreme learning machine (EM-ELMs) and support vector sequential feed forward neural networks. Error minimized extreme learning machine is a simple approach by building single hidden layer feed forward neural networks sequentially by adding random hidden nodes one by one and update the output weights incrementally to minimize the sum of squares error in training set. The experimental results showed that originally proposed EM-ELMs can be improved with respect to cost of
computation and optimal output layer weights by using approximation with optimal coefficients and interacting frequencies (SAOCIF) method.

*Savitha, et al. (2012)* presented a fast learning fully complex-valued extreme learning machine classifier, referred to as Circular Complex-valued Extreme Learning Machine (CC-ELM) for handling real-valued classification problems. CC-ELM is a single hidden layer network with non-linear input and hidden layers and a linear output layer. A circular transformation with a translational/rotational bias term that performs a one-to-one transformation of real-valued features to the complex plane is used as an activation function for the input neurons. The neurons in the hidden layer employ a fully complex-valued Gaussian-like (‘sech’) activation function. The input parameters of CC-ELM are chosen randomly and the output weights are computed analytically. This study also presents an analytical proof to show that the decision boundaries of a single complex-valued neuron at the hidden and output layers of CC-ELM consist of two hyper-surfaces that intersect orthogonally. These orthogonal boundaries and the input circular transformation help CC-ELM to perform real-valued classification tasks efficiently.

Performance of CC-ELM is evaluated using a set of benchmark real-valued classification problems from the University of California, Irvine machine learning repository. Finally, the performance of CC-ELM is compared with existing methods on two practical problems, viz., the acoustic emission signal classification problem and a mammogram classification problem.
These study results show that CC-ELM performs better than other existing (both) real-valued and complex-valued classifiers, especially when the data sets are highly unbalanced.

Zhao, et al. (2012) proposed an improved version of online sequential extreme learning machine (EOS-ELM) as online sequential extreme learning machine with forgetting mechanism (FOS-ELM) by discarding the out-dated data quickly in the process of learning to reduce their bad effect. The experimental results were conducted on stock market prediction using proposed algorithm with higher accuracy in terms of training time and better stability.

Cao, et al. (2013) proposed a new approach for image classification based on extreme k-means and effective extreme learning machine (EELM). The proposed method decomposes the image with curvlet transform, reduces the dimensionality using discriminative locality alignment (DLA), and generates a set of features using extreme k–means clustering and classified using effective extreme machine leaning. Extensive experiments are applied on various challenging databases and shows higher performances in better image classification compared to traditional methods.

Benoît, et al. (2013) proposed a new algorithm by adding one additional layer in extreme learning machines for nonlinear features selection to optimise the subset of selected features. The proposed
methodology allows making a trade-off between feature selection sparsity and generalization error providing optimal set of features.

*Khan, et al.(2013)* proposed a learning scheme based on Extreme Learning Machine (ELM) and $L_{1/2}$ regularization for a double parallel feed forward neural network. A key problem for ELM is the choice of the (minimum) number of the hidden nodes. To resolve this problem, they proposed to combine the $L_{1/2}$ regularization method that becomes popular in recent years in informatics, with ELM. It is shown in experiments that the involvement of the $L_{1/2}$ regularizer in DPFNN with ELM results in less hidden nodes but equally good performance.

*Yu et al.(2013)* proposed a new method using regularization in extreme learning machine for missing data by using $L_1$ and $L_2$ penalty to regularize the matrix computations. The whole algorithm is applied on optimized pruned Extreme Machine Learning (OP – ELM) as discussed before which uses LARS(Yu et al., 2013) to rank the neurons of the hidden layers in ELM and select the optimal number of neurons by leave one out (LOO).

*Neumann & Steil (2013)* proposed method which uses batch intrinsic plasticity as optimization method for ELMs with a fixed hidden layers size and identified the use of shaping output distribution of hidden layer neurons in an ELM. Batch intrinsic plasticity is a novel and effective scheme for input specific tuning of nonlinear transfer functions and ridge
regression that can be used to optimize the ELMs without searching for a suitable hidden layer size.

Horata, et al. (2013) proposed the new extension of ELM algorithms as Extended Complete Orthogonal Decomposition (ECOD), Iteratively Reweighted Least Squares (IRWLS-ELM), and Multi-weighted Least Trimmed squares ELM (MLTS-ELM) to solve the computational and outlier robustness problem.

He, et al. (2013) proposed efficient parallel ELM for regression problem based on MapReduce framework. The enlarged volume of datasets makes regression by ELM on very large scale datasets is a challenging task. This new algorithm demonstrates efficiently handling of very large datasets with good performance on different evaluations criterions.

Bueno-Crespo, et al. (2013) proposed the architecture for selection of extreme learning machines based on OP-ELM algorithm to design the MLP network in fast and complete way. The proposed algorithm trained the MLP network by OP-ELM and compared with the MLP network trained with back propagation algorithm.

Wang & Alhamdoosh (2013) presented an evolutionary approach for assembling extreme learning machine with size control. The proposed algorithm employs the diversity as fitness function to direct the selection of base learners, and produce an optimal solution with ensemble size control. The reporting results of proposed method outperform other
ensemble techniques, including simple average, bagging and adaboost in term of performance and efficiency.

*Xue, Yao, Wu, & Yang (2013)* proposed the hybrid model of ELM which adopts genetic algorithms (Gas) to produce a group of candidate networks and then according to ranking strategy, some of the networks are selected to ensemble a new network. To measure the robustness of the hybrid model, empirical comparisons was done with ELM and its variants in terms of better generalization and performance. Genetic algorithms are used as optimization algorithm to train a set of networks with optimal parameters to reduce the negative impact of un-optimal parameters. After the optimization procedure, a group of candidate networks are generated and use ensemble model to combine multiple trained ELMs complementing each other reducing overall prediction errors.

*Balasundaram, et al (2013)* proposed a novel 1-norm extreme learning machine (ELM) for regression and multi-classification problems. The proposed method used as linear programming problem whose solution is obtained by solving its dual exterior penalty problem as an unconstrained minimization problem using fast Newton method. The proposed method leads to a sparse model representation so that many components of the optimal solution vector will become zero and there the decision function can be determined using much less number of hidden nodes in comparison to ELM.
Wang, et al. (2013) (in press) proposed a parallelized ELM ensemble based on the Min-Mac modular network that decomposes classification problems into smaller sub problems, then train an ELM for each sub problems and ensembles theses ELMs with the min-max modular network. The experimental results showed that min-max modular ELM network speed up the training process by 1.6-4.6 times and reduces the test errors by 0.37-19.51% compared to normal ELM. (Yu et al.) proposed a Delta Test (DT) ELM (DT-ELM) which operates in an incremental way to create less complex ELM structures and determine the number of hidden nodes automatically. It uses Bayesian Information Criterion (BIC) as well as Delta Test (DT) to restrict this search as well as to consider the size of network to prevent over fitting. To get good generalization, ensemble modelling is used on different DT-ELM models for good results.

He, et al.(2013) (in press) proposed a ELM with hierarchical structure to deal high dimensional datasets with noise. The proposed HELM consists of two parts: some groups of subnets and a main net. The subnets are based on some well-trained auto associative neural networks (AANNs), which can reduce dimension and filter noise. The main net is based on traditional ELM. Data Attributes Extension Classification (DAEC), are applied to avoid designing in subnets and for faster process and better generalization.

Han, et al.(2013) proposed a hybrid learning algorithm to improve the drawback of ELM by using particle swarm optimization (PSO) to select
input weights and hidden biases and Moore Penrose generalized inverse to determine the output weights. The optimization using PSO optimized the input weight and biases according to the root mean square error (RMSE) on validation set and also the norm of the output weights.

Ye, et al. (2013) contributed an online version of ELM, as online sequential extreme learning machine in non-stationary environments where sequential arrival or large number of training data occurs. Online sequential ELM is based on time varying neural networks (TV-NN) where input weights and output weights change with time both in training and testing phases. The experimental results showed better generalization performances in non-stationary and non-linear system identification tasks.

Li, et al. (2013) (in press) proposed a fast sparse approximation schemes of extreme learning machine (ELM) names as FSA-ELM compelling low complexity and sparse solution features. Experiments on benchmark datasets show that the proposed algorithm obtains sparse classifiers at low complexity without sacrificing the generalization performance.

Zong, et al. (2013) proposed a weighted extreme learning machine for imbalanced class distribution maintaining the good generalization performance on well balanced data and imbalanced data. During implementation, they used wide ranges of feature mapping and kernels and later classified using weighted extreme learning machines (ELM). The
experimental results showed better performance in imbalanced data by weighted ELM compared to unweighted ELM.

Chen, et al. (2013) (in press) a novel extreme learning machine (ELM) for time processing of large scale remote sensing data of land cover change detection. The evaluations results showed that ELM performs better classification results in classifying different land categories compared to traditional classification algorithms like SVM, MLP etc.

Xi-zhao, et al.(2013) extends the initial localized generalization error model (LGEM) for single feed forward neural networks (SLFNNs) trained with extreme learning machines to improve the generalized capability of SLFNs. The initial localized generalization error model (LGEM) aims to find an upper bound of error between a target function and a radial basis function neural network (RBFNN) within a neighbourhood of the training samples. Experimental results on number of benchmarks data sets showed that an approximately optimal architecture in terms of number of neurons of single feed forward neural networks can be found using LGEM trained with ELM.

Gu, et al. (2013) (in press) proposed a novel machine learning algorithm as timeless online sequential extreme learning machine algorithm (TOSELM) which improves the online sequential extreme learning machine (OSELM) with central tendency and dispersion characteristics, if data has to deal with timeliness problem. The performance of proposed algorithm is validated on different datasets
showing TOSELM utilizing adaptive weight scheme and iteration scheme achieve higher accuracy, faster convergence and better stability than other machine learning methods.

*Du, et al. (2013)* proposed an automatic two stage locally regularized classifier construction (TSLRCC) method using extreme learning machines. In this algorithm, the nonlinear parameters in each term, such as the width of the Gaussian function and the power of polynomial term are determined by ELM. An initial classifier is then generated by the direct evaluation of these candidates’ models according to the leave-one-out (LOO) misclassification rate in the first stage. The significance of each selected regressor term is also checked and insignificant ones are replaced in the second stage. To reduce the computational complexity, a proper regression context is defined which allows fast implementation of the proposed method.

*Figueiredo & Ludermir (2013)* conducted a study of investigating the use of alternative topologies on performance of the PSO-ELM. In recent years, the Extreme Learning Machine (ELM) has been hybridized with the Particle Swarm Optimization (PSO) and such hybridization is called PSO-ELM. In most of these hybridizations, the PSO uses the Global topology. However, other topologies were designed to improve the performance of the PSO. In the literature, it is well known that the performance of the PSO depends on its topology, and there is not a single best topology for all problems. In this work, they investigate the effect of
eight different PSO topologies on performance of the PSO-ELM. The results showed empirically that the Global topology was more promising than all other topologies in optimizing the PSO-ELM according to the root mean squared error (RMSE) on the validation set in most of the evaluated datasets. However, no correlation was detected between this good performance on the RMSE and the testing accuracy.

Lim (2013) proposed an algorithm entitled “partitioned OS-ELM” (POS-ELM) that partitions a large data matrix into small matrices, applies an RLS (Recursive Least Square) scheme in each of the small sub-matrices and assembles the whole estimation vector by the concatenation of the sub-vectors from the RLS outputs of the sub-matrices. They proposed an algorithm entitled “partitioned OS-ELM” (POS-ELM) that partitions a large data matrix into small matrices, applies an RLS (Recursive Least Square) scheme in each of the small sub-matrices and assembles the whole estimation vector by the concatenation of the sub-vectors from the RLS outputs of the sub-matrices. Consequently, the algorithm is less complex than the conventional OS-ELM and maintains an almost compatible estimation performance.

Li, et al. (2013) proposed a Boosting weighted ELM, which embedded weighted ELM seamlessly into a modified AdaBoost framework, to solve the above problem. Intuitively, the distribution weights in AdaBoost framework, which reflect importance of training samples, are input into weighted ELM as training sample weights.
Furthermore, AdaBoost is modified in two aspects to be more effective for imbalanced learning: (i) the initial distribution weights are set to be asymmetric so that AdaBoost converges at a faster speed; (ii) the distribution weights are updated separately for different classes to avoid destroying the distribution weights asymmetry. Experimental results on 16 binary datasets and 5 multiclass datasets from KEEL repository show that the proposed method could achieve more balanced results than weighted ELM.

Pan, et al. (2013) proposed an ELM-based Q learning as an ELM as a Q-value function approximator, which is suitable for large-scale or continuous space problems. Because the number of ELM hidden layer nodes is equal to that of training samples, large sample size will seriously affect the learning speed. Therefore, a rolling time-window mechanism is introduced into the ELM-based Q learning to reduce the size of training samples of the ELM. In addition, in order to reduce the learning difficulty of new tasks, transfer learning technology is introduced into the ELM-based Q learning. The transfer learning technology can reuse past experience and knowledge to solve current issues. They proposed a multi-source transfer ELM-based Q learning (MST-ELMQ), which can take full advantage of valuable information from multiple source tasks and avoid negative transfer resulted from irrelevant information. According to the Bayesian theory, each source task is assigned with a task transfer weight and each source sample is assigned with a sample transfer weight. The task
and sample transfer weights determine the number and the manner of transfer samples. Samples with large sample transfer weights are selected from each source task, and assist Q learning agent in quick decision-making for the target task. Simulations results concerning a boat problem show that MST-ELMQ has better performance than that of Q learning algorithms without or with a single source task, i.e., it can effectively reduce learning difficulty and find an optimal solution with lesser training.

Sun, et al. (2013) proposed the classification algorithms based on conventional and optimized ELM to conduct classification over uncertain data. They view the instances of each uncertain data as the training data for learning. Then, the probabilities of uncertain data in any class are computed according to learning results of each instance. Finally, using a bound-based approach, they implement the final classification. They also extend the proposed algorithms to classification over uncertain data in a distributed environment based on OS-ELM and Monte Carlo theory. The experiments verify the performance of our proposed algorithms.

Luo & Zhang (2014) proposed a hybrid approach sparse representation based extreme learning machine for image classification. The hybrid approach is proposed because ELM is extremely fast in image classification but cannot handle noise, whereas sparse representation is robust to noise, thus combining both algorithms provides better classification in image classification.
Javed, et al. (2014) proposed a new structure of connectionist network, the summation wavelet extreme learning machine (SW-ELM) that has better accuracy and generalization performance compared to traditional ELM networks. SW-ELM is based on extreme leaning machine algorithm for fast batching, but with dual activation functions in the hidden nodes. For initialization, they introduced a priori parameter. This method was compared using different benchmarks with several neural networks algorithms on six industrial datasets.

Olatunji, et al. (2014) proposed a hybrid model through the fusion of type-2 fuzzy logic systems and extreme leaning machines for modelling permeability prediction. Type-2 Fuzzy logic systems has been used to precursor to ELM in order to better handle uncertainties in reservoir data and better generalized prediction of output. The empirical results showed that proposed method outperforms the ELM and SVM models in terms of accuracy and performance.

3.6. Conclusion

This chapter review the past literature and work done in making ANN faster and efficient on all data problems. It covers the review of existing and proposed algorithm variants in constructive, cascade correlation and extreme learning machines. The existing algorithms proposed by different authors covering the objectives and scope of thesis is reviewed and using the motivation of existing algorithms in extreme learning machines and constructive neural networks and considering the limitations of existing
variants proposed by different authors cited in chapter 3, a novel RANSAC multi model response regularization for multiple models in Chapter IV is proposed and its application to validate its accuracy and performance on different real time datasets in further three chapters.

From literature review, we can articulate that the network training utilizing ELM is faster than other algorithms and with improvement in generalization performance, there are still two major unresolved problems: (1) the only parameter that needs to be determined for ELM is the number of hidden nodes in the hidden layer. In former studies, this parameter is usually obtained by trial and error method that may not be optimal. How to choose the most suitable network structures for different applications is still unknown. (2) ELM sometimes requires a large network structure (large number of hidden nodes in the hidden layer) due to the random process in the initial stage. The issue is whether the network complexity can be further reduced without affecting the generalization performance.

Based on concept of randomization of hidden weights and biases, a new learning algorithm for SLFNs has been proposed in this thesis, known as Pruned Optimized Annular ELM, and is described in detail in Chapter IV. This algorithm results in a learning speed thousand times faster than the traditional SLFN learning algorithm like back propagation (BP) algorithm and classic ELM to obtain better generalization performance. Compared to the traditional learning algorithm, the proposed pruned annular ELM tends
to reach smallest training error with the smallest norms of weights. Experimental results on different benchmark datasets and real time problems has been done in further three chapters of the thesis which showed that proposed algorithm optimally prunes the hidden nodes, better generalization and higher classification accuracy compared to other algorithms, including SVM, OP-ELM for binary and multi-class classification and regression problems.

3.7. Roadmap of scientific contribution of thesis as follows

Chapter IV proposed an annular ELM based on RANSAC multi model response regularization to optimally prune the hidden nodes in a network and improve better generalization and classification accuracy. Experimental evaluation was conducted using an extensive comparative study with the proposed RANSAC multi model response regularization based annular ELM network on different benchmark datasets for binary and multiclass classification and regression problems.

Chapter V compared various classifiers and proposed an improved and more accurate hybrid ensemble classifiers based on Support Vector Machines (SVM and Extreme Learning Machines for protein folding recognition. In contrast to protein folds prediction, it’s very hard to classify its various folds with its different amino acids attributes due to the limited training data availability. Thus, our proposed classifier involves dimensionality reduction using PCA and LDA prior to classification.
Chapter VII presented a principled approach for investigating brain abnormalities based on wavelet based feature extraction, PCA based feature selection and deep and optimized Pruned extreme machine learning based classification comparative to various other classifiers.

Chapter VIII presented the proposed architectural design for email personalization using ourdeal database based on grad boost with optimized pruned extreme learning machines as base estimators. In chapter VIII, we also conducted a depth dive in data analysis to find each members behaviour and important attributes which plays a significant role in increasing clicks rates in personalized emails.
Chapter IV

PRUNED ANNULAR EXTREME LEARNING MACHINES
OPTIMIZATION BASED ON RANSAC MULTI MODEL
RESPONSE REGULARIZATION

4.1. Extreme Learning Machine Networks

Fortunately, due to exponential expansion in the technology, the improvement in machine learning and optimizing the parameters of statistical models improved by availability of large datasets and abundance of information of a studied phenomenon with maximum number of variables and samples. But, on the other hand, increase in number of variables with respect to samples in large dataset can create ill posed problems. Currently, most of the machines learning models are based on deterministic learning algorithms rather than non-deterministic approach which narrow down its learning applications in real time datasets. (Huang et al., 2006) proposed a new algorithm known as Extreme Machine Learning (ELM) for single hidden layer feed forward neural network which has less computational time and faster speed even on large datasets. The main working principle of ELM is random initialization of weights rather than learning through slow process via iteratively gradient based learning as back-propagation (Abid et al., 2001). In Extreme machine learning, the number of hidden nodes and their weights are randomly assigned, which distinguishes the linear differentiable between the output of hidden layer...
and output layer. The output weights can be determined by linear least square solution of hidden layer output through activation function and the data samples targets. For N arbitrary distinct samples \((x_i, t_i)\), where \(x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbb{R}_n\) and \(t_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbb{R}_m\), standard SLFNs with N hidden nodes and activation function \(g(x)\) are mathematically modelled as

\[
\sum_{i=1}^{N} \beta_i g_i(x_j) = \sum_{i=1}^{N} \beta_i g(w_i \cdot x_j + b_i) = 0, \quad j = 1, \ldots, N,
\]

(1)

Where \(w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T\) is the weight vector connecting the \(i^{th}\) hidden node and the input nodes, \(\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T\) is the weight vector connecting the \(i^{th}\) hidden node and the output nodes, and \(b_i\) is the threshold of the \(i^{th}\) hidden node. \(w_i \cdot x_j\) denotes the inner product of \(w_i\) and \(x_j\). The output nodes are chosen linear in this study.

That standard SLFNs with N hidden nodes with activation function \(g(x)\) can approximate these N samples with zero error means that

\[
\sum_{j=1}^{N} ||o_j - t_j|| = 0.
\]

i.e., there exist \(\beta_i, w_i\) and \(b_i\) such that

\[
\sum_{i=1}^{N} \beta_i g_i(w_i \cdot x_j + b_i) = t_j, \quad j = 1, \ldots, N.
\]

(2)

The above N equations (2) can be written compactly as
\[ H \beta = T, \]

Where

\[
H \left( w_1, \ldots, w_N, b_1, \ldots, b_N, x_1, \ldots, x_N \right)
\]

\[
= \begin{bmatrix}
g(w_1 \cdot x_1 + b_1) & \cdots & g(w_N \cdot x_1 + b_N) \\
\vdots & \ddots & \vdots \\
g(w_1 \cdot x_N + b_1) & \cdots & g(w_N \cdot x_N + b_N)
\end{bmatrix}_{N \times N}
\]

(3)

\[
\beta = \begin{bmatrix}
\beta_1^T \\
\vdots \\
\beta_N^T
\end{bmatrix}_{N \times m} \quad \text{and} \quad T = \begin{bmatrix}
t_1^T \\
\vdots \\
t_N^T
\end{bmatrix}_{N \times m}
\]

(4)

H is called the hidden layer output matrix of the neural network; the \(i^{th}\) column of H is the \(i^{th}\) hidden node output with respect to inputs \(x_1, x_2, \ldots, x_N\).

As per the nomenclature used in (Huang et al., 2006), H is called the hidden layer output matrix of the neural network; the \(i^{th}\) column of H is the \(i^{th}\) hidden node output with respect to inputs. (Huang et al., 2011) presented a comprehensive survey on extreme learning machines and its applications. Optimally pruned extreme leaning machine (OP-ELM) algorithm which is an extension of original ELM algorithm with pruning of neurons using ranking multi-response sparse regression (MRSR) method to design optimal neural architecture removing irrelevant variables was proposed by (Yoan et al., 2010). (Martínez-Martínez et al., 2011) proposed a new strategy to prune the ELM networks using regularized regression methods to acquire optimal tuned parameters. The algorithm can acquire optimal
tuned parameters by identifying the degree of relevance of the weights that connects the k\textsuperscript{th} hidden element with the output layer using lasso and ridge regression. Regularized version of least squares regression with several penalties on coefficient vector are used to remove the irrelevant or low relevance hidden nodes to achieve compact neural networks. (L. Singh & Chetty, 2012) proposed LDA-ELM for classification of brain abnormalities in magnetic resonance images using pattern recognition and machine learning. (Lavneet Singh, 2012) proposed a Novel Approach for protein Structure prediction using Pattern Recognition and Extreme Machine Learning.

To overcome the drawbacks of regularization or penalty method, using sparse model and removing reductant variables for better generalization and prediction accuracy, in this thesis work a new algorithm, RANSAC multi model response regularization is proposed, which implements a L1 penalty on the output weights by performing RANSAC multi model response regression between the hidden and output layer.

4.2. Regularized Extreme Learning Machines

Significant work have been done in past for better generalization, faster learning and rate of convergence. But, unfortunately, ELM also suffers with some limitations as outliers, irrelevant variables and number of hidden
nodes. To resolve these limitations of ELM, constructive and heuristic approaches have been proposed in the literature.

In recent years, regularization or penalty approach seems to be significant in resolving the ELM limitations. As in extreme machine learning, there is linear behavior between hidden layer and output layer, thus as a problem of linear regression, regularization helps to reduce the number of predictors in hidden layer by using sparse model. Least square solution with regularization is fitted to the model to find the nonzero coefficients as output weight of output layer. Using regularized sparse model, most of the predictors are moved to zero with increase in lambda. Thus, it creates a sparse model of output of hidden layer of finding the beta coefficients with respect to lambda with minimum deviance or minimum convergence with respect to Mean square error. Regularization is applied to regression problems to select the relevant hidden units, by addressing over fitting trade off with respect to network size. The big architectures are selected for the network because regularization approach prunes the network with optimal hidden neurons. In regularization based ELM, the weights of the input layer connected to the hidden layer are chosen randomly. The output weights of the output layer are determined through regularized regression removing hidden units with an optimized size network. The approach used for regularization for ELM stated as

1. Lasso regularization as $L_1$ penalty
2. Ridge Regression or $L_2$ penalty

3. Elastic net combining both $L_1$ and $L_2$ penalty

To define the general case of regularization, as a single output regression represented as –

$$Y = Xw + \varepsilon$$  \hspace{1cm} (5)

With $X = (X_1, X_2, \ldots, X_n)^T$ are the inputs of a dataset and $Y = (Y_1, Y_2, \ldots, Y_n)^T$ are the output and $W = (w_1, w_2, \ldots, w_p)^T$ are the regression weights of the hidden layer. As discussed, the model possess a linear regression between input layer and output layer, thus the simple least square solution (OLS) is a heuristic approach to solve single output regression formulated as

$$\min \hat{W}(y - X\hat{w})^T(y - X\hat{w}) \hspace{1cm} (6)$$

Or in least square form

$$\min \hat{W}\sum_{i=1}^{n}(y_i - x_i\hat{w})^2 \hspace{1cm} (7)$$

with $\hat{w}=(\hat{w}_1, \ldots, \hat{w}_n)^T$ the estimated regression weights.

The solution of Equation (7) is then obtained by a pseudo inverse (Moore-Penrose) as
\[ \hat{w}_{\text{OLS}} = (X^T X)^{-1} X^T y \]  \hspace{1cm} (8)

Moore Penrose is not useful in every numerical problem if X in Equation (3) is not full rank. Thus, in the proposed method, the SVD of X is used to compute its pseudo-inverse for least square solution providing faster computations and numerical stability.

To improve better generalization and prediction accuracy, OLS doesn’t provide a complete solution to remove irrelevant variables. OLS doesn’t use sparse models to get related variables with respect to output. To resolve the abovementioned issues with simple OLS, and regularization factors or penalty term is added to minimization cost function in Equation (3) to get the sparse model. This sparse model tried to shift most of the irrelevant variables to zero. Regularization of penalty term lambda with its weights is added to minimization problem with its nonzero coefficients to get beta coefficients of particular model.

4.3. The L1 penalty as LASSO Regularization

The minimization problem with L1 penalty is formulated as

\[ \lambda \min_{\hat{w}} \left[ \sum_{i=1}^{n} (y_i - x_i \hat{w})^2 + \lambda \sum_{j=1}^{p} |\hat{w}_j| \right] \]  \hspace{1cm} (9)

Taking an example of simple linear regression with a output as dependent variable \( y \in \mathbb{R} \) and independent vector \( S \in \mathbb{R}^Q \) to approximate the regression function
\[
\min_{\lambda, \hat{w}} \left[ \sum_{i=1}^{n} (y_i - x_i \hat{w})^2 + \lambda \sum_{j=1}^{p} |\hat{w}_j| \right]
\]  

(10)

As \( \lambda \in [0, \infty] \) increases from minimum to maximum \( \lambda \), more SW predictor’s moves towards becoming zero, tracing out the path with respect to beta coefficients, and exhibiting sparsity. The computation of Equation (10) is a classic quadratic programming problem iteratively finding the nonzero coefficients of variable correlated to output, exhibiting sparsity to less correlated variables. The number of predictors in a regression model reduces using lasso regularization. Lasso regularization identifies the important predictors among the redundant predictors and produce shrinkage estimates with potentially lower predictive errors than ordinary least squares.

**Figure 4.1.** Lasso plot of lambda on diabetes dataset with respect to its Mean Square Error (MSE) using regularization approach of L1 penalty
Figure 4.2. Lasso plot of lambda parameter using regularization approach with L1 penalty on output of Hidden Layer

Figure 4.1 and Figure 4.2 shows Lasso plot of lambda on diabetes dataset with respect to its Mean Square Error (MSE) and lambda values for each predictors as hidden units using regularization approach of L1 penalty on output of Hidden Layer. Lasso Regularization calculates shrinkage estimates with potentially lower predictive errors than ordinary least squares with least predictors as hidden units with the nonzero coefficients in the regression for various values of the Lambda regularization parameter. Larger values of Lambda appear on the left side of the graph, meaning more regularization, resulting in fewer nonzero regression coefficients.

The dashed vertical lines represent the Lambda value with minimal mean squared error (on the right), and the Lambda value with minimal mean squared error plus one standard deviation. This optimal value of Lambda is
estimated by performing cross validation up to 5 folds. The upper part of
the plot shows the degrees of freedom (df), meaning the number of nonzero
coefficients in the regression, as a function of Lambda. On the left, the large
value of Lambda causes most of the coefficient to be 0 producing sparsity.
On the right all five coefficients are nonzero, though the plot shows only
four clearly. For small values of Lambda (toward the right in the plot), the
coefficient values are close to the least-squares estimate.

A common drawback of the L1 penalty as Lasso regularization is that it
tends to be produce more sparse model with respect to increasing lambda,
where \( w_j = 0 \). Thus, using lasso regularization, reducing the number of
predictors further with increasing sparse estimates, is quite challenging to
tune.

4.4. The L2 Penalty as Tikhonov regularization

Tikhonov regularization is almost same as Lasso regularization only the
difference is in minimization problem, it involve a penalty of square of
regression coefficients formulated as

\[
\min_{\lambda, \hat{w}} \left[ \sum_{i=1}^{n} (y_i - x_i \hat{w})^2 + \lambda \sum_{j=1}^{p} \hat{w}_j^2 \right]
\]

(11)

The L2 venality provides lower MSE and outperforms the Lasso
regularization in achieving better prediction accuracy.
The major drawback of this regularization method is similar to the ordinary least square solution, since all variables are filtered due to L2 penalty, it does not give any unique solution. The computation times is also a concern while applying L2 penalty regularization, as it use cross validation to find the lambda weight with minimum MSE specially when the number of predictors are large enough.

To resolve the limitation and overcome the drawbacks of the two regularization approaches, hybrid penalties are proposed using both L1 and L2 penalty in same minimization problem as

$$\min \left[ \sum_{i=1}^{n} (y_i - x_i \hat{\mathbf{w}})^2 + \frac{\lambda_1}{\sqrt{1 + \lambda_2^2}} \sum_{j=1}^{p} |\hat{w}_j| \right]$$

(12)

The above minimization equation is also termed as elastic net which creates great shrinkage effect on the regression coefficients than original lasso regularization.

Though elastic net is an effective way of implementing shrinkage estimates on regression coefficients, there are still two parameters \(\lambda_1\) and \(\lambda_2\), that need to be optimized and tuned using cross validation (CV) which is costly search in two dimensional matrix. This is hardly feasible if computation times of ELM are taken under consideration. To overcome the drawbacks of regularization or penalty method, using sparse model and removing reductant variables for better generalization and improved prediction
accuracy, multi RANSAC based regularization is proposed in this thesis work, which implements a L1 penalty on the output weights by performing multi RANSAC response regression between the hidden and output layer.

4.5. ANNULAR ELM

Circular Back Propagation (CBP) networks defined in Equation (13) improves over the basic formulation of MLP; the CBP model augments the input vector by one additional dimension, which is computed as the norm of the input vector itself. In a classic set-up involving a single layer network, the estimation process supported by the enhanced CBP network can be expressed as

$$y_{\text{CBP}}(x) = \xi \left( b + \sum_{j=1}^{N} \hat{y}_{j} x_{j} + \sum_{i=1}^{M} \tilde{w}_{i,j} x_{i} + w_{j,M} + 1 \left\| x \right\|^{2} \right)$$ (13)

Based on circular back-propagation network, in this a new architecture network for extreme learning machines known as Annular ELM is proposed. The first formation of the annular ELM is augmented by adding one more dimension in both training and testing data, which can be computed as

Training data_{ij} = Training data_{ij} + 1

Where \( i = \{1 \ldots m\} \) as number of observation and \( j = \{1 \ldots n\} \) as number of features. Thus, Similarly
Testing data_{pj} = Testing data_{pj+1}

As the annular topology is implemented to the input layer and fed into the hidden layer in ELM network, the random weights and the hidden layer and the bias is redefined as

\[ A_j(x, \hat{w}_j, \hat{b}_j) = \left( \tilde{\xi}(\hat{w}_j, \tilde{x} + \hat{b}_j) \right) \]  

(14)

which could then be rewritten as

\[ A_j(x, \hat{w}_j, \hat{b}_j) = \tilde{\xi}(z_j, \|x - c_j\|^2 + \bar{b}_j) \]  

(15)

Where

\[ Z_j = \hat{w}_{j,M+1} \]  

(16)

\[ c_j = \begin{bmatrix} \hat{w}_{j,3} \\ \frac{\hat{w}_{j,2}}{2 \hat{w}_{j,M+1} + 1} \\ \vdots \\ \frac{\hat{w}_{j,M}}{2 \hat{w}_{j,M+1} + 1} \end{bmatrix} \]  

(17)

\[ \bar{b}_j = \frac{1}{\hat{w}_{j,M+1}} \left( \sum_{k=1}^{M} \frac{\hat{w}_{j,k}^2}{4 \hat{w}_{j,M+1}} - \hat{b}_j \right) \]  

(18)

Input data (X_{ij}) = [X_{ij}, \ldots, X_{ij+1}]
Figure 4.3: Architectural Design of Annular ELM based on RANSAC Multi Model Response Regularization

The annular based ELM is able to map both linear and circular separation boundaries by boosting the ability of ELM network. In annular based ELM, the initial hidden weights are chosen randomly along with the bias similar to ELM. But, later, new random weights and bias are approximated by applying proposed annular functions to estimate the output of hidden layer known as hidden matrix.

Later, proposed RANSAC multi response regularization is applied to the output of hidden layer using annular ELM topology to prune the hidden units for better generalization and higher accuracy.

4.6. RANSAC

The RANSAC (Random Sample Consensus) algorithm was proposed by (Fischler & Bolles, 1981) to estimate the parameters of a certain model from a set of data with large number of outliers. RANSAC take out the outliers from a data if it doesn’t fit with a set of parameters within the error
threshold with respect to maximum deviation. RANSAC can handle outliers greater than 50% of the entire dataset known as breakdown point.

RANSAC first hypothesizes minimum sample sets (MSSs) randomly selected from the input dataset and the parameters of the model are estimated using MSS. To test the estimated parameters of the model using MSS, RANSAC checks which element of the entire dataset are consistent with the defined model known as consensus set (CS). RANSAC ranks the consensus with a set of iterations with respect to estimated probability at certain threshold. RANSAC is extensively used in computer vision, motion detection and features matching of images and is optimized using different parameters (Raguram, Frahm, & Pollefeys, 2008), (Nister, 2005), (Chum, Matas, & Kittler, 2003).

4.7. RANSAC Multi Model Response Regularization

4.7.1. RANSAC Multi Model Response Regularization for Regression problems

To implement the RANSAC algorithm on regression problems, we proposed a RANSAC multi model response regularization which implements the sequential RANSAC on multiple models. To implement RANSAC, which in our case, are the irrelevant hidden nodes as predictor variables, and H is the hidden matrix as input from equation. In our case, the output weights follow a linear regression between hidden and output layer defined as
\[ Y = mx + \epsilon \] (19)

Where \( Y \) is the output of instances of data, \( m \) is the predictor’s weights or slope and \( x \) is the input data and \( c \) is the constant. The output weights follow a linear regression between hidden and output layer defined as

\[ Y = \text{Output weights} \cdot H + \epsilon \]

Where

\[
H = \begin{bmatrix}
g(w_1 \cdot x_1 + b_1) & \cdots & g(w_N \cdot x_1 + b_N) \\
\vdots & \ddots & \vdots \\
g(w_1 \cdot x_N + b_1) & \cdots & g(w_N \cdot x_N + b_N)
\end{bmatrix}_{N \times N}
\] (20)

\[
\text{OutputWeights} = \text{Multi - RANSAC} \left[ \begin{bmatrix}
g(w_1 \cdot x_1 + b_1) & \cdots & g(w_N \cdot x_1 + b_N) \\
\vdots & \ddots & \vdots \\
g(w_1 \cdot x_N + b_1) & \cdots & g(w_N \cdot x_N + b_N)
\end{bmatrix}_{N \times N} \right]^T
\] (21)

\[
\text{CS} = \text{RANSAC} \left( \sum_{w=1}^{W} \left( \sum_{j=1}^{N} D_{wj} \right) \right)
\] (22)

\( D_{xwj} = \{ x_{1w}, x_{12}, \ldots, x_{wj} \} \) be the sets of data \( H \) with \( w \)th observations in rows and \( j \)th as hidden nodes predictors in columns of \( D \) matrix. For regression problems, sequential RANSAC implements on the set of all inliners, \( D \) that are generated by \( W \) different models where \( W_m = \text{rand}(H_M) \). The numbers of models are randomly generated using 20% of the input data.

Multi Models are selected from hidden matrix and RANSAC is implemented by bootstrapping so as to get maximum sparse coefficients of \( \text{CS} \) for hidden nodes. Multi models from hidden matrix are randomly
selected as 20% of data so as to get maximum sparse CS coefficients to prune the hidden nodes. CS ranking help to determine zero and non-zero coefficients of hidden nodes which is efficiently determined by bootstrapping of RANSAC on selected multi models.

To estimate the parameters of W models, each one is represented by k dimensional parameter vector \( \theta_w \) at each iteration iter. CS is estimated using MSS of each W model. The iteration runs M times, and is calculated before after removing the inliers from data D. The total number of inliers at iteration iter is less than total number of inliers at iteration iter-1. The complete formulation of multiple RANSAC response is defined in Equation (21). The set of all inliers D is generated by W different models has cardinality CS as

\[
N_I = (N_{I,1} + N_{I,2} + \ldots \ldots + N_{I,W})
\]

As RANSAC is a parametric model, the set of parameters need to be defined before implementing RANSAC. These set of parameters are defined as

- Epsilon = False alarming rate as the probability of the algorithm throughout all the iterations will never sample a MSS containing only inliers
- Probability of inlier = the probability that a point whose fitting error is less or equal than is actually an inlier
• Sigma = Gaussian noise
• Estimate function = function that returns the estimate of the parameter vector starting from a set of data.
• Mean square function = function that returns the fitting error of the data.
• CS ranking Algorithm = Algorithm to rank the CS of data
• Minimum number of iterations
• Maximum number of iterations
• W = 20% of the training data

Let $M_w(\theta_w)$ defines the manifold of dimension $k_w$ of all points with respect to parameter $\theta_w \in R^{k_w}$ for the specified model for $1 \leq w \leq W$ with a subset $S_w$ from $D$ of $k_w$ elements at iteration $i$ called minimal sample set (MSS). To estimate the parameters of $W$ models, each one represented parameter vector $\theta_w$. At each $i$ iterations, MSS for each $W$ model is defined and CS is estimated removing all outliers.

To combine and fuse the estimated CS computed from $i(W)$ iterations, the whole RANSAC multi model response algorithm can be summarize as follows

**RANSAC Multi Model Response Regularization Algorithm**

| $S_w^{(i)} = S(\theta_w)$ be the CS of $w^{th}$ model at $i^{th}$ iteration. |
| The all combined updated CS of $S(\theta_w)$ is updated as $M_{(iter)} = 100; i = 0$ |
| For $i \leq M_{(iter)}$ (maximum number of iterations) |
| Do $i = i + 1$ |
| $\{S_w^{(i-1)}, S_w^{(0)}\}$ |
| While $1 \leq w \leq W$ |
| $S_i(\theta) = S^{(0)} \cup \bigcap_{i} S^{(i)} \theta$ |
To reduce the number of hidden units with respect to ranked CS estimated using RANSAC multi-response algorithm is calculated as

Hidden Layer Output ($H_1$) = $S(i)\theta^*H$ (23)

which reduces the hidden matrix as hidden layer output $H_1$ into zero and nonzero coefficients of ranked elements with respect to estimated CS. Nonzero coefficients are extracted from sparse matrix which reduces the no of hidden units to which are ranked less giving the hidden units coefficients highly correlated. Thus

Output weights = $(H_i^T H_i)^{-1} H_i^T T$ (24)

4.7.2. RANSAC Multi Model Response Regularization for binary and multiclass problems

The proposed RANSAC multi model response regularization for binary and multiclass problems for ELM is implemented using one against all (OAA) method. In OAA method, $j$ binary classifiers will be constructed in which all the training examples will be used at each time of training. The training data having the original class label $j_n = (1, \ldots, n)$ have each $j_n$ elements of positive one class and the remaining training data will be of zero class, creating $j_n$ models implementing proposed RANSAC multi model response regularization on $(j_n)$ binary classes. Finally, CS defined as $S'(\theta)$ of $j(n)$ classes is computed as
\[ S'(\theta)_j = \sum_{j=1}^{n} S^{(\omega)}_j \cup S^{(\omega-1)}_j \]  \hspace{1cm} (25)

\[ S'(\theta) = \sum_{j=1}^{n} S^{(\omega)}_j \cup S^{(\omega)}_n \]  \hspace{1cm} (26)

For this, consider the ELM for multi-class classification problem, formulated as k binary ELM classification problem with the following form:

\[ H_{w1} = y_1 \ldots H_{wj} = y_j; \]

Where for each j, w_j is the output weights from the hidden layer to the output layer with output vector \( y_j = (y_{ij}, \ldots, y_{mj})' \in \mathbb{R}_m \). Thus the output of the hidden layer as H hidden matrix defines with respect to multiclass binary classifiers as

\[ H_j = \sum_{j=1}^{n} H \ast Y \begin{pmatrix} Y_j = 1 \\ o \end{pmatrix} \]  \hspace{1cm} (27)

Where H is the hidden layer output matrix and Y is the j binary classes with \( m^{th} \) observations of training data and n binary classes as columns vectors. Thus, we get H_j hidden matrix where each H_j belong to each binary class and RANSAC multi response regularization is implemented to acquire CS for each binary class as \( S'(\theta)_j \). It can be concluded that RANSAC multi response regularization for binary and multiclass problems work in similar fashion as OAA-ELM with j binary classes with a difference of \( j^{th} \) label with positive class and rest other classes with -1 class.
To improve better generalization and prediction accuracy, OLS doesn’t provide a complete solution to remove irrelevant variables. OLS doesn’t use sparse models to get related variables with respect to output. To resolve the mentioned issues with simple OLS, regularization factors or penalty approach added to minimization cost function in Equation (20) to get sparse model of OLS to acquire sparse model which tries to shift most of the irrelevant variables to zero.

\[ X = \text{LSQR} \left( A, B \right) \] attempts to solve the system of linear equations \( A^*X=B \) for \( X \) if \( A \) is consistent, otherwise it attempts to solve the least squares solution \( X \) that minimizes \( \| B - A^*X \| \). LSQR is an iterative method to find Ordinary Least Squares solution for large sparse matrix which is analytically equivalent to the standard equation of conjugate gradients, but possess more favourable numerical properties.

The next section defines the experimental results and comparative analysis between proposed algorithm and with other algorithms used in prior work, on benchmark datasets in term of testing accuracy, RMSE and number of hidden nodes.

4.8. Experimental Results

To evaluate the proposed algorithm performance, a comparative study with original ELM, SVM and other machine learning methods using publicly available benchmarks datasets of regression, binary and multiclass classification was done.
Further, for the proposed algorithm, different activation functions were also used, called as $G(a, b, x)$, with existing algorithms like ELM, MLP, and SVM. The different activation functions of hidden layer is defined as

Sigmoid Function

$$G(a, b, x) = \frac{1}{1 + \exp(-(a^T x + b))}$$

Radial Basis Function

$$G(a, b, x) = \left( \| x - a \|_2^2 + b^2 \right)^{1/2}$$

For experimental study, the input weights and the hidden nodes are chosen randomly at the beginning of the iterations and were fixed in rest of the iterations. The optimal values of average training and testing accuracy, number of hidden nodes corresponding to different algorithms including the proposed algorithm was determined by using 10-fold cross validation on both training and testing datasets.

All experiments were conducted in MATLAB R2010 platform running on windows 7-64 bit operating system with 3.0 GHz Intel® core 2 i5 processor having 8 GB of RAM. LIBSVM is used for the implementation of SVM in matlab and weka platform. OP-ELM Matlab toolbox is used for implementation of OP-ELM (Miche, Sorjamaa, & Lendasse, 2008). RANSAC toolbox is used for implementation and extended for implementing sequential RANSAC (Zuliani, 2008). In our experiments, several benchmark datasets were chosen. The data sets were collected from the University of California at Irvine (UCI) Machine Learning Repository.
(Blake & Merz, 1998) and they were all processed using 10 different random permutations; for each permutation, two thirds are taken for the training set, and the remaining third for the test set by using cross validation function in MATLAB (crossvalind). Training sets are then normalized (zero-mean and unit variance) and test sets are also normalized using the very same normalization factors than for the corresponding training set.

Datasets

In our experiments, several benchmark problems were chosen. The data sets were collected from the University of California at Irvine (UCI) Machine Learning Repository and their different attributes for the data sets are summarized in Table 4.1 and 4.2.

Table 4.1: Classification datasets attributes and classes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Attributes</th>
<th>Classes</th>
<th>Training data size</th>
<th>Testing data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDBC</td>
<td>30</td>
<td>2</td>
<td>427</td>
<td>142</td>
</tr>
<tr>
<td>Wincosin_BC</td>
<td>10</td>
<td>2</td>
<td>525</td>
<td>174</td>
</tr>
<tr>
<td>Cleveland</td>
<td>13</td>
<td>2</td>
<td>228</td>
<td>75</td>
</tr>
<tr>
<td>Australian Credit</td>
<td>14</td>
<td>2</td>
<td>518</td>
<td>172</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>34</td>
<td>2</td>
<td>264</td>
<td>87</td>
</tr>
<tr>
<td>Diabetes</td>
<td>8</td>
<td>2</td>
<td>576</td>
<td>192</td>
</tr>
<tr>
<td>Liver Disorders</td>
<td>6</td>
<td>2</td>
<td>259</td>
<td>86</td>
</tr>
<tr>
<td>Iris</td>
<td>4</td>
<td>3</td>
<td>113</td>
<td>37</td>
</tr>
<tr>
<td>Wine</td>
<td>13</td>
<td>3</td>
<td>134</td>
<td>44</td>
</tr>
<tr>
<td>Glass</td>
<td>9</td>
<td>6</td>
<td>161</td>
<td>53</td>
</tr>
<tr>
<td>Auto Vehicle</td>
<td>18</td>
<td>4</td>
<td>635</td>
<td>211</td>
</tr>
<tr>
<td>Page Blocks</td>
<td>10</td>
<td>5</td>
<td>4105</td>
<td>1368</td>
</tr>
<tr>
<td>Image Seg</td>
<td>19</td>
<td>7</td>
<td>1733</td>
<td>577</td>
</tr>
<tr>
<td>Satellite</td>
<td>36</td>
<td>6</td>
<td>4827</td>
<td>1608</td>
</tr>
</tbody>
</table>
The data sets were all processed using 10 different random permutations; for each permutation, two thirds are taken for the training set, and the remaining third for the test set (see Table 1) by using cross validation function in MATLAB (crossvalind). Training sets are then normalized (zero-mean and unit variance) and test sets are also normalized using the same normalization factors that was used for the corresponding training set. The 10 fold cross validation also enables to obtain an estimate of the standard deviation of the results presented (see Table 4.2).

Table 4.2. – Regression datasets attributes and classes

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Attributes</th>
<th>Training data size</th>
<th>Testing data size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-MPG</td>
<td>8</td>
<td>294</td>
<td>98</td>
</tr>
<tr>
<td>Machine-CPU</td>
<td>7</td>
<td>157</td>
<td>52</td>
</tr>
<tr>
<td>Servo</td>
<td>5</td>
<td>126</td>
<td>41</td>
</tr>
<tr>
<td>Forest-Fires</td>
<td>13</td>
<td>388</td>
<td>129</td>
</tr>
<tr>
<td>Boston</td>
<td>14</td>
<td>380</td>
<td>126</td>
</tr>
<tr>
<td>Concrete-CS</td>
<td>9</td>
<td>773</td>
<td>257</td>
</tr>
<tr>
<td>Abalone</td>
<td>8</td>
<td>3133</td>
<td>1044</td>
</tr>
<tr>
<td>Wine-(white)</td>
<td>12</td>
<td>3674</td>
<td>1224</td>
</tr>
<tr>
<td>Wine-(Red)</td>
<td>12</td>
<td>1200</td>
<td>399</td>
</tr>
<tr>
<td>Parkinson</td>
<td>22</td>
<td>4407</td>
<td>1468</td>
</tr>
<tr>
<td>Kin-8</td>
<td>9</td>
<td>6144</td>
<td>2048</td>
</tr>
<tr>
<td>Demo</td>
<td>5</td>
<td>1536</td>
<td>512</td>
</tr>
<tr>
<td>Ailerons</td>
<td>40</td>
<td>5366</td>
<td>1788</td>
</tr>
</tbody>
</table>

4.8.1. Classification Results

Table 4.3 depicts the comparative analysis of proposed RANSAC multi model response regularized ELM with support Vector Machine (SVM) and other ELM variants. For each dataset, training data is trained with higher number of hidden nodes so as to prune the network with optimal hidden
nodes with better classification accuracy. For binary classification, it can be seen from Table 4.3 using sigmoid function; most of the binary datasets shows the higher testing accuracy compared to SVM, ELM and OP-ELM. Using the RBF function in Table 4.4, the proposed model did not exhibit better testing accuracy results as compared to other algorithms. But for both activation functions, RANSAC multi model response regularized ELM prunes the number of hidden nodes improving the optimality of the ELM network.

Table 4.5 depicts the comparative analysis of proposed algorithm with SVM and other variants for multi-class classification. As can be seen from Table 4.5 using sigmoid and RBF kernel., RANSAC multi model response ELM shows significant improvement in testing accuracy results compared to other algorithms for wine, glass, auto and segmentation datasets. Table 4.6 defines the number of hidden nodes pruned using RANSAC multi model response ELM with optimal higher testing accuracy.

Table 4.3. Experimental results in terms of testing accuracy for binary classification using Sigmoid kernel

<table>
<thead>
<tr>
<th>Datasets</th>
<th>HN</th>
<th>SVM</th>
<th>OP-ELM</th>
<th>ELM</th>
<th>RANSAC-ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDBC</td>
<td>200</td>
<td>95.77</td>
<td>90.85</td>
<td>95.39</td>
<td>95.81</td>
</tr>
<tr>
<td>Win-BC</td>
<td>200</td>
<td>94.82</td>
<td>89.66</td>
<td>96.53</td>
<td>96.97</td>
</tr>
<tr>
<td>Cleveland</td>
<td>200</td>
<td>76.00</td>
<td>78.67</td>
<td>90.40</td>
<td>83.31</td>
</tr>
<tr>
<td>Aus-credit</td>
<td>200</td>
<td>83.72</td>
<td>84.88</td>
<td>81.40</td>
<td>86.03</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>200</td>
<td>78.16</td>
<td>79.31</td>
<td>79.45</td>
<td>88.92</td>
</tr>
</tbody>
</table>
Table 4.4. Experimental results in terms of testing accuracy for binary classification using RBF kernel

<table>
<thead>
<tr>
<th>Datasets</th>
<th>HN</th>
<th>SVM</th>
<th>OP-ELM</th>
<th>ELM</th>
<th>RANSAC-ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDBC</td>
<td>200</td>
<td>97.18</td>
<td>90.14</td>
<td>83.20</td>
<td>93.20</td>
</tr>
<tr>
<td>Win-BC</td>
<td>200</td>
<td>96.55</td>
<td>97.13</td>
<td>90.74</td>
<td>95.78</td>
</tr>
<tr>
<td>Cleveland</td>
<td>200</td>
<td>76.00</td>
<td>78.61</td>
<td>90.55</td>
<td>81.25</td>
</tr>
<tr>
<td>Aus-credit</td>
<td>200</td>
<td>83.72</td>
<td>86.63</td>
<td>73.84</td>
<td>84.72</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>200</td>
<td>89.65</td>
<td>95.40</td>
<td>78.41</td>
<td>92.90</td>
</tr>
<tr>
<td>Diabetes</td>
<td>200</td>
<td>75.00</td>
<td>79.69</td>
<td>69.20</td>
<td>77.30</td>
</tr>
<tr>
<td>Liver Disorders</td>
<td>200</td>
<td>74.41</td>
<td>68.60</td>
<td>57.47</td>
<td>67.30</td>
</tr>
</tbody>
</table>

As can be seen from Table 4.6, for binary and multi-class datasets, the RANSAC multi model response ELM significantly prunes the number of hidden nodes from higher number of hidden nodes maintaining the higher testing accuracy, faster implementation and better generalization performance for most of the binary and multiclass classification.

Table 4.5. Experimental results in terms of testing accuracy for multiclass classification

<table>
<thead>
<tr>
<th>Datasets</th>
<th>H</th>
<th>Testing(sigmoid)</th>
<th>Testing(RBF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>200</td>
<td>SVM 91.89 EL M 78.32 RANSAC-EL M 83.89</td>
<td>SVM 97.29 EL M 85.24 RANSAC-EL M 89.46</td>
</tr>
<tr>
<td>Wine</td>
<td>200</td>
<td>SVM 93.18 EL M 92.77 RANSAC-EL M 95.45</td>
<td>SVM 97.72 EL M 84.85 RANSAC-EL M 94.36</td>
</tr>
<tr>
<td>Dataset</td>
<td>ELM Hidden Nodes</td>
<td>RANSAC-ELM hidden nodes(sig)</td>
<td>RANSAC-ELM hidden nodes(RBF)</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------</td>
<td>-------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>WDBC</td>
<td>200</td>
<td>40</td>
<td>48</td>
</tr>
<tr>
<td>Wincosin_Breast_cancer</td>
<td>200</td>
<td>34</td>
<td>30</td>
</tr>
<tr>
<td>Cleveland</td>
<td>200</td>
<td>52</td>
<td>53</td>
</tr>
<tr>
<td>Australian Credit</td>
<td>200</td>
<td>33</td>
<td>36</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>200</td>
<td>90</td>
<td>92</td>
</tr>
<tr>
<td>diabetes</td>
<td>200</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>Liver Disorders</td>
<td>200</td>
<td>61</td>
<td>70</td>
</tr>
<tr>
<td>Iris</td>
<td>200</td>
<td>94</td>
<td>71</td>
</tr>
<tr>
<td>Wine</td>
<td>200</td>
<td>60</td>
<td>58</td>
</tr>
<tr>
<td>Glass</td>
<td>200</td>
<td>65</td>
<td>123</td>
</tr>
<tr>
<td>Auto Vehicle</td>
<td>500</td>
<td>90</td>
<td>118</td>
</tr>
<tr>
<td>Page Blocks</td>
<td>500</td>
<td>97</td>
<td>230</td>
</tr>
<tr>
<td>Image Segmentation</td>
<td>500</td>
<td>232</td>
<td>245</td>
</tr>
<tr>
<td>Satellite</td>
<td>500</td>
<td>250</td>
<td>252</td>
</tr>
</tbody>
</table>

4.8.2. Regression Results

Table 4.7:- Experimental results in terms of training root mean square accuracy for regression using sigmoid kernel

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ELM</th>
<th>RANSAC-ELM</th>
<th>ELM</th>
<th>RANSAC-ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoMPG</td>
<td>200</td>
<td>1.4316</td>
<td>1.4339</td>
<td>1.5571</td>
</tr>
<tr>
<td>CPU</td>
<td>200</td>
<td>72.1778</td>
<td>48.2834</td>
<td>75.2482</td>
</tr>
<tr>
<td>Datasets</td>
<td>HN</td>
<td>OP-ELM</td>
<td>ELM</td>
<td>RANSAC-ELM</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Auto-MPG</td>
<td>200</td>
<td><strong>11.7282</strong></td>
<td>91.2388</td>
<td>11.9512</td>
</tr>
<tr>
<td>Machine-CPU</td>
<td>200</td>
<td>2.0269e+04</td>
<td>6.2133e+08</td>
<td>1.4293e+03</td>
</tr>
<tr>
<td>Servo</td>
<td>200</td>
<td>0.5501</td>
<td>11.4059</td>
<td>0.2861</td>
</tr>
<tr>
<td>Forest-Fires</td>
<td>200</td>
<td>4.6637</td>
<td>5.5861</td>
<td>1.9943</td>
</tr>
<tr>
<td>Boston</td>
<td>200</td>
<td>29.8807</td>
<td>36.5606</td>
<td>8.7404</td>
</tr>
<tr>
<td>Concrete-CS</td>
<td>200</td>
<td>223.4046</td>
<td>58.5347</td>
<td>28.3328</td>
</tr>
<tr>
<td>Abalone</td>
<td>200</td>
<td>6.1798</td>
<td>10.6831</td>
<td>4.1517</td>
</tr>
<tr>
<td>Wine-(white)</td>
<td>200</td>
<td>0.5494</td>
<td>0.5251</td>
<td>0.4637</td>
</tr>
<tr>
<td>Wine-(Red)</td>
<td>200</td>
<td>0.4698</td>
<td>0.4738</td>
<td>0.3627</td>
</tr>
<tr>
<td>Parkinson</td>
<td>200</td>
<td>0.0025</td>
<td>0.0014</td>
<td>0.0012</td>
</tr>
<tr>
<td>Kin-8</td>
<td>200</td>
<td>0.0481</td>
<td><strong>0.0189</strong></td>
<td>0.0219</td>
</tr>
<tr>
<td>Demo</td>
<td>200</td>
<td>1.7995</td>
<td>54.3682</td>
<td>1.4459</td>
</tr>
<tr>
<td>Ailerons</td>
<td>200</td>
<td>0.7569</td>
<td>3.9234e-08</td>
<td>2.5698e-08</td>
</tr>
</tbody>
</table>

Table 4.8. - Experimental results in terms of testing root mean square accuracy for regression using Sigmoid kernel

<table>
<thead>
<tr>
<th>Datasets</th>
<th>HN</th>
<th>OP-ELM</th>
<th>ELM</th>
<th>RANSAC-ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-MPG</td>
<td>200</td>
<td>41.4800</td>
<td>49.6352</td>
<td>8.0027</td>
</tr>
<tr>
<td>Machine-CPU</td>
<td>200</td>
<td>9.5348e+03</td>
<td>6.8199e+07</td>
<td>1.4612e+03</td>
</tr>
<tr>
<td>Servo</td>
<td>200</td>
<td>0.5550</td>
<td>2.1920</td>
<td>0.3258</td>
</tr>
<tr>
<td>Forest-Fires</td>
<td>200</td>
<td>2.4981</td>
<td>4.0079</td>
<td>1.8157</td>
</tr>
<tr>
<td>Boston</td>
<td>200</td>
<td>33.7665</td>
<td>60.3066</td>
<td>10.3812</td>
</tr>
</tbody>
</table>

Table 4.9. - Experimental results in terms of testing root mean square accuracy for regression using RBF kernel
Table 4.7 depicts the training accuracy for different regression problem datasets using ELM and proposed algorithm with respect to different kernel function. For both activation functions, for regression problems, RANSAC multi model response regularized ELM can prune the number of hidden nodes improving the optimality of the ELM network and has least RMSE compared to other algorithms.

For regression problems, Table 4.8 and 4.9 depicts the comparative analysis of proposed RANSAC multi model response regularized ELM with traditional ELM and other ELM variants using various kernels. As can be seen from Table 4.8 and 4.9, RANSAC multi model response annular ELM shows the significant higher testing accuracy results compared to other algorithms on different datasets. Table 4.10 defines the number of pruned hidden nodes using RANSAC multi model response annular ELM with optimal higher testing accuracy.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>200</th>
<th>227.5232</th>
<th>95.1963</th>
<th>31.114</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concrete-CS</td>
<td>200</td>
<td>5.9829</td>
<td>4.9659</td>
<td>3.9806</td>
</tr>
<tr>
<td>Abalone</td>
<td>200</td>
<td>0.6074</td>
<td>1.4766</td>
<td>0.4808</td>
</tr>
<tr>
<td>Wine-(white)</td>
<td>200</td>
<td>0.5203</td>
<td>1.4369</td>
<td>0.3837</td>
</tr>
<tr>
<td>Wine-(Red)</td>
<td>200</td>
<td>0.0031</td>
<td>0.0065</td>
<td>0.0012</td>
</tr>
<tr>
<td>Parkinson</td>
<td>200</td>
<td>0.0426</td>
<td>0.0361</td>
<td>0.0224</td>
</tr>
<tr>
<td>Kin-8</td>
<td>200</td>
<td>1.7991</td>
<td>2.5114</td>
<td>1.3197</td>
</tr>
<tr>
<td>Demo</td>
<td>200</td>
<td>0.8324</td>
<td>2.0346e-07</td>
<td>2.6605e-08</td>
</tr>
</tbody>
</table>

Table 4.10. - Experimental results in terms of number of hidden nodes pruned by proposed method using sigmoid and radial basis functions for regression
<table>
<thead>
<tr>
<th>Dataset</th>
<th>ELM Hidden Nodes</th>
<th>RANSAC-ELM hidden nodes(sig)</th>
<th>RANSAC-ELM hidden nodes(RBF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-MPG</td>
<td>200</td>
<td>164</td>
<td>132</td>
</tr>
<tr>
<td>Machine-CPU</td>
<td>200</td>
<td>96</td>
<td>102</td>
</tr>
<tr>
<td>Servo</td>
<td>200</td>
<td>110</td>
<td>87</td>
</tr>
<tr>
<td>Forest-Fires</td>
<td>200</td>
<td>93</td>
<td>81</td>
</tr>
<tr>
<td>Boston</td>
<td>200</td>
<td>78</td>
<td>113</td>
</tr>
<tr>
<td>Concrete-CS</td>
<td>200</td>
<td>139</td>
<td>138</td>
</tr>
<tr>
<td>Abalone</td>
<td>200</td>
<td>118</td>
<td>97</td>
</tr>
<tr>
<td>Wine-(white)</td>
<td>200</td>
<td>96</td>
<td>79</td>
</tr>
<tr>
<td>Wine-(Red)</td>
<td>200</td>
<td>113</td>
<td>97</td>
</tr>
<tr>
<td>Parkinson</td>
<td>200</td>
<td>148</td>
<td>127</td>
</tr>
<tr>
<td>Kin-8</td>
<td>200</td>
<td>62</td>
<td>147</td>
</tr>
<tr>
<td>Demo</td>
<td>200</td>
<td>133</td>
<td>100</td>
</tr>
<tr>
<td>Ailerons</td>
<td>200</td>
<td>87</td>
<td>125</td>
</tr>
</tbody>
</table>

Figure 4.4 represents the ROC curve using extreme learning machine algorithm of auto dataset for multiclass classification. As can be seen in the figure, the ROC curve (between true positive and false positive rate) is linear at lower true positive rates for different classes. Figure 4.5 depicts the confusion matrix using ELM for auto dataset for multiclass classification.
Figure 4.4: ROC Curve using Extreme Learning Machine Algorithm of Auto Dataset for Multiclass Classification.

Confusion Matrix

<table>
<thead>
<tr>
<th>Target Class</th>
<th>Output Class</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>43</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Class 2</td>
<td>2</td>
<td>24.6%</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Class 3</td>
<td>2</td>
<td>0.9%</td>
<td>33</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Class 4</td>
<td>52</td>
<td>0.9%</td>
<td>0</td>
<td>0</td>
<td>52</td>
</tr>
</tbody>
</table>

Figure 4.5: Confusion Matrix depicting true positive and false positive rate using Extreme Learning Machine Algorithm of Auto Dataset for Multiclass Classification.
Figure 4.6: ROC Curve using Proposed Extreme Learning Machine Algorithm of Auto Dataset for Multiclass Classification.

Figure 4.7: Confusion Matrix depicting true positive and false positive rate using Extreme Learning Machine Algorithm of Auto Dataset for Multiclass Classification

As we can see from the Figure 4.5, the false positive rate is higher for output class with respect to target class. Compared with Figure 4.6 showing ROC curve using proposed ELM algorithm of auto dataset for multiclass classification, the true positive rate is higher with respect to false positive
rate for multi classes. Further, as is seen in Figure 4.7, the true positive rate is higher of output class with respect to target class.

Thus, from above experiments, it can be stated that proposed method based on RANSAC regularization performs very well in terms of training and testing accuracy compared to traditional ELM and its variants. The above experiment also reduces the size of network to optimal size, decreasing the computation time and resulting in faster performance of the network.

4.9. Conclusions

In this Chapter, we proposed an annular ELM based on RANSAC multi model response regularization to optimally prune the hidden nodes in a network and improve better generalization and classification accuracy. Experimental evaluation was conducted using an extensive comparative study with the proposed RANSAC multi model response regularization based annular ELM network on different benchmark datasets for binary and multiclass classification and regression problems.

It can be concluded from experimental results that proposed RANSAC multi model response regularized based annular ELM works performs significantly well in terms of higher classification accuracy with optimally pruned hidden units. Further, the proposed algorithm is faster compared to other algorithms in the study, since it implements the ELM with less pruned hidden units without sacrificing the higher generalization capability of ELM network.
Chapter V

Protein Folding Recognition using Pruned Annular Extreme Learning Machines

5.1. Introduction

Protein folding recognition is one of the challenging and most important problems in the area of bioinformatics. The structure of a protein plays an important role in its biological and genetic function (Chan & Dill, 1993). Thus, to know about protein and genetic sequences, which are basically sequences of various amino acids in protein molecules, we first need to know its structure, and then know how it is folded. With increase in available computational power, several research works have been reported in the genome sequencing area with the objective of amino acid sequencing, although there are still challenges in determining the three-dimensional (3-D) structure of proteins. Several machine-learning methods have been introduced in previous studies to predict protein folding structures and amino acid sequences. (Ding & Dubchak, 2001) used support vector machine (SVM) and artificial neural network (ANN) classifiers to extract features and various properties, based on which certain predefined protein folds were predicted. (Shen & Chou, 2006) proposed a model based on a nearest-neighbour algorithm and its modification called the K-local hyperplane (H-KNN) method. (Nanni, 2006) proposed a model using
Fisher’s linear classifier and H-KNN classifier. (Eddy, 1996) used hidden Markov models (HMM) for protein folding recognition. Their model predicted the most accurate rate of protein folding. However, the disadvantage of using HMM is that it needs high computational power working on large datasets of protein folding for training and testing, although the reduced state-space HMM method with smaller architecture can also be used (Lampros, Papaloukas, Exarchos, Goletsis, & Fotiadis, 2007), (Lampros, Papaloukas, Exarchos, Fotiadis, & Tsalikakis, 2009).

Basically, in protein folding problems, classification is done by using two types of classifiers. The probabilistic approach uses training data to map the probability estimates of each class and find the posterior probability of test data. The second type of classifier uses the likely neighbourhood or weights between different classes based on instances of training data. In our study, we use the proposed pruned annular ELM, with generative quadratic discriminant analysis (QDA) for classification of protein folds.

Many researchers have used the fusion of different classifiers to increase the performance and accuracy of the recognition rate in bioinformatics (Ghanty & Pal, 2009). (Guo & Gao, 2008) proposed a fused classifier for large-scale human protein subcellular location prediction, and used SVM classifiers fused with max rule algorithms. The challenge in terms of the classification and accuracy in the protein folding problem is that, after converting data into m × n matrices, the number of features is large
(sometimes thousands) with lesser training data, which makes it harder to implement learning models. To address this problem, a better approach is to reduce the high dimensionality of the features by using some feature selection techniques so that, with \( n \) observations and instances, we have \( n \) features for the learning or training phase. Reducing the features can be achieved by using statistical or probabilistic approaches. There are two methods for feature reduction: feature selection and feature transformation, to convert high-dimensional data into a new space with reduced dimensionality.

The major challenge in the above-mentioned methods and approaches lies in the complexity of the data, due to the large number of folding classes with only a small number of training samples and the multiple, heterogeneous feature groups, making it harder to discover patterns. Due to this problem, the classification accuracy achieved in protein folding prediction problems in most of the previously reported work is no more than 60\%, which is in general lower than typical pattern recognition problems in other application domains.

In this study, we examined several learning classifiers, such as SVM, linear regression classifiers, ANN with multilayer perceptron (MLP), and random forest, and then finally we compared the performance of these classifiers with our proposed hybrid pruned annular ELM using a fusion of QDA and principal component analysis (PCA) with SVM. Experimental validation
on publicly available protein databases showed that the recognition accuracy achieved with our proposed classifiers is significantly higher, than achieved with the methods proposed in previous work. SVM is a binary classifier, whereas protein folding recognition is a multiclass problem. Thus, in this study, we use the strategy of assigning weights using a discriminant function based on QDA.

5.2. Protein Folding Recognition Problem

Proteins are macromolecules composed of 20 different amino acids linked by peptide bonds in a linear order (Sanger & Thompson, 1953). The linear polypeptide chain is called the primary structure of the protein. The primary structure can be represented as a sequence of 20 different letters, where each letter denotes an amino acid. In the native state, the amino acids (or residues) of a protein fold into local secondary structures including alpha helix, beta sheet, and non-regular coil. The secondary structure elements are further packed to form tertiary structure due to hydrophobic forces and side-chain interactions between amino acids. The tertiary structures of several related proteins can bind together to form a protein complex called the quaternary structure. In a cell, proteins with native tertiary structures interact to carry out all kinds of biological functions, including enzymatic catalysis, transport and storage, coordinated motion, mechanical support, immune protection, generation and transmission of nerve impulses, and control of growth and differentiation. Extensive
biochemical experiments in the past have shown that a protein’s function is determined by its structure.

Thus, elucidating a protein’s structure is the key to understand its function, which has fundamental significance in biological and medical sciences. Currently, there are about 30,000 proteins with determined structures deposited in the protein data bank (PDB) (Berman, Henrick, Nakamura, & Markley, 2007). These diverse and abundant structures provide invaluable data to understand how a protein folds into its unique 3-D structure and to predict the structure from its sequence.

![Figure 5.1 Three-dimensional model of converting primary and secondary structure into the 3-D protein structure for various folds](image)

Figure 5.1 Three-dimensional model of converting primary and secondary structure into the 3-D protein structure for various folds

5.3. **Protein Sequence, Structure, and Function**

Since the pioneering experiments (Sanger & Thompson, 1953) showed that a protein’s structure is dictated by its sequence, predicting the structure of a protein from its sequence has become one of the most fundamental problems in structural biology. In the postgenomic era, with the application of high-throughput DNA and protein sequencing
technologies, the number of protein sequences has increased exponentially, whereas experimental determination of protein structure remains very expensive, time consuming, labour intensive, and some-time impossible. Currently, only about 1.5% of protein sequences (about 30,000 out of 2 million) have solved structures, and the gap between proteins with known and unknown structures is still increasing.

Thus, predicting the structure of a protein from its sequence is increasingly imperative and useful. Protein structure prediction is becoming a vital tool for understanding phenomena in modern molecular and cell biology and has important applications in medical sciences, e.g., in drug design.

5.4. 1-D, 2-D, and 3-D Protein Structure Prediction

Protein structure prediction is often classified into three levels: one-dimensional (1-D), two-dimensional (2-D), and 3-D (Cheng, Randall, & Baldi, 2006). One-dimensional prediction is used to predict structural features such as the secondary structure and the solvent accessibility of each residue along the one-dimensional protein sequence. Two-dimensional prediction is used to predict the relationship between residues (e.g., contact map prediction and disulphide bond prediction). Three-dimensional prediction is used to predict the 3-D coordinates of all residues or all atoms in a protein. Although the ultimate goal is to predict the 3-D structure, 1-D and 2-D prediction are of great interest to biologists and represent important steps toward 3-D structure prediction. The existing,
rather practical, method for 3-D and 2-D structure prediction shown in Figure 5.1 is the template based approach including comparative modelling (or homology modelling) and fold recognition (or threading). This approach is based on the observation that nature tends to reuse existing structures/folds to accommodate new protein sequences and functions during evolution.

More than half a century ago, evidence began to accumulate that a major part of most proteins folded structure consists of two regular, highly periodic arrangements, designated $\alpha$ and $\beta$. The key to both structures is the hydrogen bond. A hydrogen atom is nothing more than a proton with a surrounding electron cloud. When one of these atoms is chemically bonded to an electron-withdrawing atom such as nitrogen or oxygen, much of the electron cloud moves toward the nitrogen or oxygen. The proton is thus left almost bare, with its positive charge largely unshielded. If it comes close to another atom with a small extra negative charge, typically an oxygen or nitrogen atom, and the partial positive and negative charges will attract each other. It is this attraction that produces the hydrogen bond and stabilizes the $\alpha$ and $\beta$ structures. The structure now called a $\alpha$-helix is a right-handed spiral stabilized by hydrogen bonds between each amino acids nitrogen atom and the oxygen atom of the fourth one up the chain. This means that there are 3.6 amino acids for each turn of the helix. The main part of the amino acid (the side-chain, designated R in Figure 5.2) sticks out from this spiral backbone like the bristles on a bottle brush. The
β structure is now called a β-sheet. It is essentially flat, with the side-chains sticking out on alternate sides.

Figure 5.2 (a) A model α-helix shows the hydrogen bonds (dotted lines) between oxygen and hydrogen atoms of the fourth amino acid up the chain. (b) β-Sheets are also held together by hydrogen bonds. Transparent arrows show the direction of individual β-strands. Chains running in the same direction (left pair) are called parallel β-sheet; strands running in opposite directions (right pair) are said to be antiparallel β-sheet. Atom coloring is as follows: carbon = green, oxygen = red, nitrogen = blue, and hydrogen = white. (Courtesy: Stanley Krystek, Bristol-Myers Squibb, Pharmaceutical Research Institute)

β-Sheet is also stabilized by hydrogen bonds between nitrogen and oxygen atoms. In this case, however, the hydrogen-bonded atoms belong to different amino acid chains running alongside each other. The sheets are parallel if all the chains run in the same direction and antiparallel if alternate chains run in opposite directions. Antiparallel sheets are often, but not always, formed by a single chain looping back upon itself. When a single chain loops back on itself to form an antiparallel β-sheet, the one to three amino acids linking the two strands are known as a β-turn. Today,
scientists recognize the \( \beta \)-turn as one of the fundamental elements of protein structure. All other local arrangements of amino acids are described as random coil, although they are random only in the sense of not being periodic.

Thus, although the protein sequence space (the number of protein sequences) is very large, the protein structure space (the number of unique protein folds) is relatively small and expected to be limited. Currently, millions of protein sequences have been collected, but the number of unique structures (folds) in protein classification databases such as SCOP (structural classification of proteins) and CATH (protein structure classification database developed by University College, London), which are the publically available databases for protein structures classified domains based on their structures and amino acid sequences, among them only about 1000 (out of 30 000 protein structures). Moreover, among the protein structures newly determined in structural genomics projects, the novel folds account for only a small portion (about 10%), and the overall fraction of new folds has continued to decrease over the past 15 years. Thus, most protein sequences, particularly similar protein sequences within the same family and superfamily evolving from common ancestors, have structures similar to other proteins. So, given a query protein without a known structure, template-based prediction is used first to identify a template protein (if one exists) with a solved, similar structure to the query
protein. The structure of the template protein is then used to model the structure of the query protein based on the alignment between the query sequence and the template structure.

5.5. Protein Database and Its Features

5.5.1. Protein Database

In this study, we used datasets derived from the structural classification of proteins (SCOP) database (Berman et al., 2007). Details of these protein sets are described and presented in Table 5.1. All feature vectors are standardized and normalized to the range of $[-1; +1]$ before applying any classifiers. The proteins in both the training and test sets belong to 27 different protein folds, corresponding to four major structural classes: $\alpha$, $\beta$, $\alpha/\beta$, and $\alpha + \beta$. In this study, we compare classification results of protein folds using the overall accuracy $Q$, defined as the percentage of correctly recognized proteins among all proteins in the test dataset, which can be expressed as $Q = c/n$, where $c$ is the number of query proteins whose folds have been correctly recognized and $n$ is the total number of proteins in the test dataset. Table 5.1 presents the parameters extracted from the protein sequence with a 125-dimensional feature vector for each protein in this dataset.
5.5.2. Feature Vectors

In our experiments, we used the features described by (Ding & Dubchak, 2001). These feature vectors are based on six parameters: amino acid composition (C), predicted secondary structure (S), hydrophobicity (H), normalized van der Waals volume (V), polarity (P), and polarizability (Z). Each parameter consists of 21 features, except for the amino acid composition (C), which consists of 20 features. We then concatenate all features into one feature vector (C, S, H, V, P, Z) containing 125 features. All feature vectors are standardized and normalized to the range of $[-1; +1]$ before applying any classifiers. Standardizing and normalizing the features results in features with smaller numeric ranges, making it much easier for classifiers to do the classification task.

<table>
<thead>
<tr>
<th>Protein feature</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amino acid composition</td>
<td>20</td>
</tr>
<tr>
<td>Predicted secondary structure</td>
<td>21</td>
</tr>
<tr>
<td>Hydrophobicity</td>
<td>21</td>
</tr>
<tr>
<td>van der Waals volume</td>
<td>21</td>
</tr>
<tr>
<td>Polarity</td>
<td>21</td>
</tr>
<tr>
<td>Polarizability</td>
<td>21</td>
</tr>
</tbody>
</table>

5.6. Feature Selection and Classification

5.6.1. Support Vector Machine Classifiers (SVM)

The support vector machine (SVM) is a well-known large margin classifier proposed by Vapnik(Cortes & Vapnik, 1995). The basic concept of the
SVM classifier is to find an optimal separating hyper-plane, which separates two classes. The decision function of the binary SVM is

\[ f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b \] (3)

where \( b \) is a constant, \( y_i \in \{-1,1\} \), \( 0 \leq \alpha_i \leq C \), \( I = 1,2, \ldots \ldots \ldots N \) are non-negative Lagrange multipliers, \( C \) is a cost parameter, that controls the trade-off between allowing training errors and forcing rigid margins, \( x_i \) are the support vectors and \( K(x_i, x) \) is the kernel function.

After that we follow on SVM as multiclass problem using one against one method. This method constructs \( k(k-1)/2 \) classifiers where each one trains data from two classes. For training data from the \( i^{th} \) and the \( j^{th} \) classes, we solve the following binary classification problem. In this study, we use the max win voting strategy. If \( \text{sign} \left((w^{ij})^T \emptyset (x) + b^{ij}\right) \) says \( x \) is in the \( i^{th} \) class, then the vote for the \( i^{th} \) class is added by one. Otherwise, the \( j^{th} \) is increased by one. Then the largest vote will be given to specific class on variable \( x \).

We used the software LIBSVM library for experiments. LIBSVM is a general library for support vector classification and regression, which is available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. As mentioned above, there are different functions to map data to higher dimensional spaces, practically we need to select the kernel function \( K(x_i; x_j) = \emptyset (x_i)^T \emptyset (x_j) \). There are several types of kernels that can be used with all kinds
of problems. Each kernel has different parameters for different problems. For example, some well-known problems with large amount of features, such as text classification and DNA problems, are reported to be classified more correctly with the linear kernel. In our study, we use the RBF kernel. A learner with the RBF kernel usually performs no worse than others do, in terms of the generalization ability. In this study, we did some simple comparisons and observed that using the RBF kernel the performance is a better than the linear kernel \( K(x^i, x^j) = \phi(x^i)^T \phi(x^j) \) for all the problems we studied. Therefore, for the three data sets instead of staying in the original space, a non-linear mapping to a higher dimensional space seems useful. Another important issue is the selection of parameters. For SVM training, few parameters such as the penalty parameter \( C \) and the kernel parameter of the RBF function must be determined in advance. Choosing optimal parameters for support vector machines is an important step in SVM design. We use the cross validation on different parameters for the model selection.

5.6.2. Quadratic Discriminant Analysis

Quadratic discriminant analysis (QDA) (Fukunaga, 2013) describe the likelihood of a class as a Gaussian distribution and then uses the posterior distributions estimates to estimate the class for a given test vector. This approach leads to the function:
\[ d_k(x) = (x - \mu_k)^T \sum_k^{-1} (x - \mu_k) + \log \sum_k k - 2 \log p(k) \quad (4) \]

Where \( \sum_k \) is the covariance matrix, \( x \) is the test vector, \( \mu_k \) is the mean vector, and \( p(k) \) is the prior probability of the class \( k \). The Gaussian parameters for each class can be estimated from the training dataset, so the values of \( \sum_k \) and \( \mu_k \) are replaced in above formula by its estimates \( ^\wedge \sum_k \) and \( ^\wedge \mu_k \). However, when the number of training samples is small, compared to the number of dimensions of the training vector, the covariance estimation can be ill-posed. The approach to resolve the ill-posed estimation is to regularize the covariance matrix \( \sum_k \). It can be replaced by the average matrix i.e.

To apply QDA, our first goal is to reduce the dimension of the data by finding a small set of important features which can give good classification performance. Feature selection algorithms can be roughly grouped into two categories: filter methods and wrapper methods. Filter methods rely on general characteristics of the data to evaluate and to select the feature subsets without involving the chosen learning algorithm (QDA in our case). Filters are usually used as a pre-processing step since they are simple and fast. A widely-used filter method for bioinformatics data is to apply a univariate criterion separately on each feature, assuming that there is no interaction between features. We apply the t-test on each feature and
compare p-value (or the absolute values of t-statistics) for each feature as a measure of how effective it is at separating groups.

5.6.3. Feature Selection

As the dataset in our protein folding recognition is ill posed, we have greater number of features and samples are small, which may be insufficient to solve the problem. Thus, in this case, we have to use feature selection algorithm to reduce the dimensionality of feature space. There are many feature selection algorithms used in the past literature reviewed, but we try to approach efficient and faster feature selection algorithm to reduce the dimensionality space. Thus, in this study we use three different methods for feature selection on protein dataset.

5.6.4. Generalized Linear Model

In this model, we have used the generalized linear model for feature selection based on the probability value as (p-value) using t test statistics. We have used as a pre-processing step since they are simple and fast. We apply a univariate criterion separately on each feature, assuming that there is no interaction between features. We apply the t-test on each feature and compare p-value (or the absolute values of t-statistics) for each feature as a measure of how effective it is at separating groups. In order to get a general idea of how well-separated the two groups are by each feature, we plot the empirical cumulative distribution function (CDF) of the p-values:
There are about 35% of features having p-values close to zero and over 50% of features having p-values smaller than 0.05, meaning there are more than 60 features among the original 125 features that have strong discrimination power. One can sort these features according to their p-values (or the absolute values of the t-statistic) and select some features from the sorted list. However, it is usually difficult to decide how many features are needed unless one has some domain knowledge or the maximum number of features that can be considered has been dictated in advance based on outside constraints. One quick way to decide the number of needed features is to plot the MCE (misclassification error, i.e., the number of misclassified observations divided by the number of
observations) on the test set as a function of the number of features. The resubstitution MCE is over-optimistic. It consistently decreases when more features are used and drops to zero when more than 69 features are used. However, if the test error increases while the resubstitution error still decreases, then overfitting may have occurred. This simple filter feature selection method gets the smallest MCE on the test set when 28 features are used. The plot shows overfitting begins to occur when 28 or more features are used. The smallest MCE on the test set is 12.5%.

5.6.5. Sequential Feature Selection Algorithm

We have also extracted the features in our dataset using sequential feature selection algorithm. Sequential feature selection is one of the most widely used techniques. It selects a subset of features by sequentially adding (forward search) or removing (backward search) until certain stopping conditions are satisfied. In this study, we use forward sequential feature selection to find important features. More specifically, since the typical goal of classification is to minimize the MCE. The training set is used to select the features and to fit the QDA model, and the test set is used to evaluate the performance of the finally selected feature. During the feature selection procedure, to evaluate and to compare the performance of the each candidate feature subset, we apply stratified 7-fold cross-validation to the training set.
5.6.6. The PCA/LDA-Based Pruned Annular Extreme Learning Machines Algorithm

Step 1: Use PCA on training and testing dataset

\[
\hat{S} = \frac{1}{N} \sum_{j=1}^{K} \sum_{i=1}^{N_j} (x_{ji} - \mu_j)(x_{ji} - \mu)^T
\]

Step 2: Use LDA on training and testing dataset

\[
\hat{S}_w = \frac{1}{N} \sum_{j=1}^{K} \sum_{i=1}^{N_j} (x_{ji} - \mu_j)(x_{ji} - \mu)^T
\]

\[
\hat{S}_b = \sum_{j=1}^{K} N_j (\mu_j - \mu)(\mu_j - \mu)^T
\]

\[
J(T) = \frac{\hat{S}_b}{\hat{S}_w}
\]

Where J(T)=Linear Discriminant Function

Given a training set

\[
\hat{S} = \{(x_i, y_i) \in \mathbb{R}^{m+n}, y_i \in \mathbb{R}^{m}\}_{i=1}^{\mathbb{N}}, \text{ for activation function } g(x) \text{ and the number of hidden neurons } \mathbb{N};
\]

Step 2: For \(k = 1, ..., \mathbb{N}\) randomly assign the input weight vector \(w_k \in \mathbb{R}^{n}\) and bias \(b_k \in \mathbb{R}\)

Step 3: Determine the hidden layer output matrix \(H\).

Step 4: Calculate \(H^+\).

Step 5: Calculate the output weights matrix \(\hat{\beta}\) by \(\hat{\beta} = H^+T\).

Step 6: Calculate the decision function of Pruned Annular ELM as

182
\[ f(x) = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x) + b \]

We have used RBF, Sigmoid (Sig) and Linear (Lin) as activation functions in ELM and SVM. In later section, a comparative study has been shown on multi kernels with ELM respect to their learning classification accuracy rate. The parameters C for SVM and g have certain parametric value. Both values has been experimentally chosen, which was done using a cross-validation procedure on the training dataset. The best recognition ratio was achieved using parameter values gamma = 0.5 and C = 300.

Fig.5.4.PCA features of protein sequences. x-Axis: principle components (eigenvectors); y-axis: the weights(importance) of the principle components to discriminate different data points (protein sequences) from the original feature space
5.7. Experiments

Our experiments include implementation of proposed Annular ELM and other classifiers on the prescribed protein dataset and to do a comparative study to achieve the higher accuracy rate. For this purpose, firstly we choose certain parameters. We use the cross validation technique to avoid over fitting. We use k-fold cross validation with k=7, because there must be at least 7 samples of each class in the training dataset. Then the second parameter is to decide the kernel for SVM classifier. The RBF kernel is

\[ K(x_i, x) = -\gamma (x - x_i)^2 \gamma > 0 \]  

We used other kernels also, as the linear, polynomial and Gaussian kernels. The RBF kernel gave the best results in our experiments. The parameters C from Equation(3) and g have certain parametric value. Both values has been experimentally chosen, which was done using a cross-validation procedure on the training dataset. The best recognition ratio was achieved using parameter values g = 0.7 and C = 300.

5.7.1. Feature Selection algorithms

The best selection method for feature selection is to check all combinations. However, the number is too high so can’t go ahead with combinations only. So, we combine all features based on parameters C, S, H, V, P, Z to make an subset. Then the various feature selection algorithms as previously described was implemented to reduce the dimensionality space. Or first
approach is to use simple Wrapper feature selection which reduces the features to 28 features. Our second approach is to use sequential forward selection algorithm which gives a combination 69 features matrix with observed instances. The combination of 69 features gives the best recognition ratio results using cross validation procedure.

We also use the probabilistic feature selection algorithm which uses the generalized linear model as t- test statistics to find the significant features. Using this algorithm, we sort out the 48 features and given average results using cross validation procedure.

5.7.2. Exemplar Results

In this study we have used SVM and the proposed hybrid classifier for increasing the performance and accuracy. In this study, accuracy is measured in terms of percentage classified recognized ratio. Suppose there is \( N = n_1 + n_2 + n_3 + \ldots + n_p \) test proteins, where \( n_i \) is the number of proteins which belongs to the class \( i \). Suppose that \( c_i \) of proteins from \( n_i \) are correctly recognized (as belonging to class \( i \)). So the total number of \( C = c_1 + c_2 + c_3 + \ldots + c_p \) proteins is correctly recognized. Therefore the total accuracy is \( Q = C/N \).

The SVM classifiers follow on a feature vector to each class with the minimum value of discriminant function. So for every instance, we calculate the function value of each class. In this way we create \( m \times n \)
matrices of two different feature vectors. The SVM classifiers were used with 126D feature vector. All binary classifiers were trained, with each instance being assigned to each class. The recognition ratio using SVM classifiers is 63.89%.

The final step was to use three feature selection algorithm matrices. The results are shown in Table 5.2. There were three different voted table and weights for three different matrices with different selected features vectors. The recognition ratio was about 65.26% which is slightly higher than using SVM classifiers alone. Table 5.3 describes the comparative study of using various classifiers including SVM and using our proposed hybrid pruned annular classifiers.

In this study, we present a combined generative based classifier and use of feature selection algorithm for protein fold recognition. Table 5.2 explains the comparison between different methods examined in terms of recognition ratio. In comparison with other methods, our proposed PCA and LDA based ELM classifier show higher accuracy rate up to 65.32% and 68.12%. Table 5.4 presents the recognition ratio of using multiple kernels with proposed PCA and LDA based SVM and PCA and LDA based ELM. It can be seen in Table 5.4 that in both classifiers, RBF kernels gives the higher accuracy comparative to other kernels. In addition, PCA and LDA based ELM classifier from Table 5.3 shows more promising results in terms of recognition rate with up to 82.45% after 50 trials using RBF
kernels. As a result, it can be seen that from all classifiers including SVM and proposed SVM, our proposed Extreme Learning Machines shows promising results in terms of higher accuracy rate of protein folds. Table 5.5 depicts the ELM training time and testing time ranging from 31.2 ms to 67.5 ms as a function of number of hidden neurons (N). This indicates a fast training process unlike training of a gradient descent based BPN which usually gets trapped in multiple local minima’s and thus waste time.

Table 5.2 Recognition ratio obtained using various feature selection algorithms

<table>
<thead>
<tr>
<th>Selection method (number of features)</th>
<th>Recognition ratio Pruned Annular ELM (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized linear model (28)</td>
<td>55.36</td>
</tr>
<tr>
<td>SFSa (69)</td>
<td>60.19</td>
</tr>
<tr>
<td>QDA (28)</td>
<td>65.17</td>
</tr>
<tr>
<td>PCA (60)</td>
<td>66.43</td>
</tr>
</tbody>
</table>

* SFS: sequential feature selection

Table 5.3:- Comparison among different methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Recognition Ratio (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>63.75</td>
</tr>
<tr>
<td>H-KNN</td>
<td>57.4</td>
</tr>
<tr>
<td>Bayesian Naives</td>
<td>52.30</td>
</tr>
<tr>
<td>Random Forest</td>
<td>53.72</td>
</tr>
<tr>
<td>MLP</td>
<td>54.72</td>
</tr>
<tr>
<td>LDA-SVM</td>
<td>65.32</td>
</tr>
<tr>
<td>PCA-SVM</td>
<td>68.12</td>
</tr>
<tr>
<td>Pruned Annular ELM-QDA (proposed method)</td>
<td>77.67</td>
</tr>
<tr>
<td>Pruned Annular ELM-PCA (proposed method)</td>
<td>82.45</td>
</tr>
</tbody>
</table>

Table 5.4 Comparison among different methods using multiple kernels

<table>
<thead>
<tr>
<th>Method</th>
<th>Sigmoid Kernels</th>
<th>RBF Kernel</th>
<th>Linear Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA-SVM</td>
<td>57.23</td>
<td>65.32</td>
<td>52.11</td>
</tr>
<tr>
<td>PCA-SVM</td>
<td>65.78</td>
<td>68.12</td>
<td>51.89</td>
</tr>
<tr>
<td>LDA-ELM</td>
<td>72.34</td>
<td>77.67</td>
<td>68.90</td>
</tr>
<tr>
<td>PCA-ELM</td>
<td>78.56</td>
<td>82.45</td>
<td>71.35</td>
</tr>
</tbody>
</table>
### Table 5.5: Training and Testing Time computed time spans for PCA/LDA based ELM

<table>
<thead>
<tr>
<th>Number of Hidden Neurons (N)</th>
<th>ELM Training Time (Sec)</th>
<th>Testing Time (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0.312</td>
<td>0.456</td>
</tr>
<tr>
<td>40</td>
<td>0.312</td>
<td>0.545</td>
</tr>
<tr>
<td>60</td>
<td>0.675</td>
<td>0.521</td>
</tr>
</tbody>
</table>

**Figure 5.5:** Classification of 27 fold protein sequences using QDA-ELM Classifiers
Figure 5.6: Classification of 27 fold protein sequences using PCA-SVM. The result shows that our proposed hybrid classifiers show better performance and accuracy in terms of recognition ratio of proteins instances of various classes.

5.8. Conclusions and Future Work

In this study, we compared various classifiers and proposed an improved and more accurate hybrid ensemble classifiers based on Support Vector Machines (SVM and Extreme Learning Machines for protein folding recognition. In contrast to protein folds prediction, it’s very hard to classify its various folds with its different amino acids attributes due to the limited training data availability. Our proposed classifier involves dimensionality reduction using PCA and LDA prior to classification. This results in significant improvement in recognition accuracy. The combined PCA-SVM and LDA-SVM classifiers results in an accuracy of (65.32%) and (68.12%), as compared to SVM (56.70%) and other above mentioned classifiers. The similar PCA/LDA feature selection for Extreme Learning Machines also results in improvement in recognition accuracy up to (77.67%) and (82.45%). We also compare our proposed classifiers using multiple kernels approach resulting in promising results. Table 4.3 reveals that RBF Kernel give the better classification accuracy rate for both ELM and SVM classifiers compared to other kernels. These results seem to be
positive for classification for protein sequences as compared to results reported in the previous work.

Additionally, all our experiments were done on the original protein features developed by Ding and Dubchak. Further work can be done using different feature selection and classification algorithms on other high dimensional protein sequences to reduce the computational power, more accuracy and higher recognition rate.
Chapter VI

Using Pruned Annular Extreme Learning Machines for Identifying the Brain Abnormalities from MRI Structural Images

6.1. Introduction

Magnetic Resonance Images (MRI) is an advance technique used for medical imaging and clinical medicine and an effective tool to study the various states of human brain. MRI images provide the rich information of various states of brain, and can be used to study, diagnose and carry out unparalleled clinical analysis of brain to find out if the brain is normal or abnormal. However, the data extracted from the images is very large and it is hard to make a conclusive diagnosis based on such raw data. In such cases, we need to use various image analysis tools to analyse the MRI images and to extract conclusive information to classify into normal or abnormalities of brain. The level of detail in MRI images is increasing rapidly with availability of 2-D and 3-D images of various organs inside the body.

Fully automatic normal and diseased human brain classification from magnetic resonance images (MRI) is of great importance for research and clinical studies. Recent work (Chaplot, Patnaik, & Jagannathan, 2006),
(Maitra & Chatterjee, 2008) has shown that classification of human brain in magnetic resonance (MR) images is possible via machine learning and classification techniques such as artificial neural networks and support vector machine (SVM), and unsupervised techniques such as self-organization maps (SOM) and fuzzy c-means combined with appropriate feature extraction techniques (Maitra & Chatterjee, 2008). Other supervised classification techniques, such as k-nearest neighbours (k-NN), which group pixels based on their similarities in each feature image can be used to classify the normal/pathological T2-weighted MRI images.

MRI is often the medical imaging method of choice when soft tissue delineation is necessary. This is especially true for any attempt to classify brain tissues (Fletcher-Heath, Hall, Goldgof, & Murtagh, 2001). The most important advantage of MR imaging is that it is a non-invasive technique (Chaplot et al., 2006). Use of computer technology in medical decision support is now widespread and pervasive across a wide range of medical areas such as cancer research, gastroenterology, heart diseases, brain tumors, etc. (L. Singh & Chetty, 2014). Fully automatic classification of normal and diseased human brain based on MRI is of great importance for research and clinical studies.

Out of several debilitating ageing related health conditions, white matter lesions (WMLs) are commonly detected in elders and in patients with multiple brain abnormalities like Alzheimer’s disease, Huntington’s
disease and other neurological disorders. Therefore, segmentation and quantification of white matter lesions via texture analysis is very important in understanding the impact of aging and diagnosis of various brain abnormalities. Manual segmentation of WM lesions, which is still used in clinical practices, shows the limitation to differentiate brain abnormalities using human visual abilities. Such methods can produce a high risk of misinterpretation and can also contribute to variation in correct classification. The signal of homogeneity and heterogeneity of abnormal areas in Region of Interest (ROI) in white matter lesions of brain in T2-MRI images can be quantified by texture analysis algorithms. The ability to measure small differences in MRI images is essential and important to reduce the diagnosis errors of brain abnormalities.

In this chapter, we present the study on magnetic resonance imaging (MRI) of various states of brain, by extracting the most significant features for classification into normal and abnormal brain images. We propose a novel method based on the wavelet transform to initially decompose the images, and then use various feature selection algorithms to extract the most significant features of brain from the MRI images. By using different classifiers to detect the abnormality of brain images from a publicly available neuroimaging dataset, we found that a principled approach involving wavelet-based feature extraction, followed by selection of the most significant features using the proposed Optimised Pruned Extreme
Learning Machines, and classification using learning-based classifiers results in a significant improvement in accuracy as compared with previously reported studies.

The supervised feature classification from T2 MRI images, however, suffers from two problems. First, because of the large variability in image appearance between different datasets, the classifiers need to be retrained from each data source to achieve good performances. Second, these types of algorithms rely on manually labelled training datasets to compute the multi-spectral intensity distribution of the white matter lesions making the classification unreliable. Inspired by new segmentation algorithms in computer vision and machine learning, we propose an efficient semi-automatic and deep learning algorithm for white matter (WM) lesion segmentation around Region of Interest (ROI) based on extreme and deep machine learning. Further, we compare this novel approach with some of the other supervised machine learning techniques reported previously.

6.1.1. Background

Two types of multiplicative noise often arise in several imaging modalities: speckle and Poisson noise. Both types are called multiplicative in the sense that their variance is not constant but depends on the parameters to be estimated (Unser & Aldroubi, 1996). The ultimate goal of post scanning noise removal methods in MRI is to obtain piecewise-constant or slowly varying signals in homogeneous tissue regions while preserving tissue
boundaries. In the literature, both statistical approaches and diffusion filter methods have been used to remove noise from digital images. An early study of image improvement in MRI using a statistical approach was described in (Laine, 2000). Previous general reviews of wavelets in biomedical image processing, including some early work on functional MRI (fMRI) that measures brain activity by detecting associated changes in blood flow, are provided in (Bullmore et al., 2003). Statistical issues in wavelet analysis of time series are addressed comprehensively in (Bruce, Donoho, & Gao, 1996). Several research groups have pioneered applications of wavelets to various issues in fMRI data analysis. The authors in (Barra & Boire, 2000) explored two- and three-dimensional wavelet transforms as spatial filters of radio ligand binding potential maps measured using positron emission tomography (PET) which uses a nuclear medical imaging technique that produces a three dimensional image of functional processes in the body. The work in (Barra & Boire, 2000) reported a technique for brain tissue classification or segmentation of structural MRI based on fuzzy clustering of wavelet coefficients.

6.2. Materials and Methods

6.2.1. Datasets

The input dataset consists of axial, T2-weighted, 256 X 256 pixel MR brain images (Figure 6.1). These images were downloaded from the (Harvard Medical School website (http://med.harvard.edu/AANLIB/). Only those
sections of the brain in which lateral ventricles are clearly seen are considered in our study. The number of MR brain images in the input dataset is 60 of which 6 are of normal brain and 54 are of abnormal brain. The abnormal brain image set consists of images of brain affected by Alzheimer’s and other diseases. The remarkable feature of a normal human brain is the symmetry that it exhibits in the axial and coronal images. Asymmetry in an axial MR brain image strongly indicates abnormality. Hence symmetry in axial MRI images is an important feature that needs to be considered in deciding whether the MR image at hand is of a normal or an abnormal brain. A normal and an abnormal T2-weighted MRI brain image are shown in Figure 6.1 (a) and 1(b), respectively. Indeed, for multilayer learning models like deep and extreme machine learning algorithms, there is a need for big datasets for training. However due to lack of availability of proper datasets in MRI imaging, we used this dataset for examining the performance of proposed approaches for this study.

With wavelet decomposition techniques, at each decomposition level, the length of the decomposed signals is half the length of the signal in the previous stage. Hence the size of the approximation component obtained from the first level decomposition of an NXN image is N/2 X N/2, second level is N/4 X N/4 and so on. As the level of decomposition is increased, compact but coarser approximation of the image is obtained. Thus,
wavelets provide a simple hierarchical framework for interpreting the image information.

6.3. Feature Extraction

6.3.1. Decomposition of images Using Wavelets

Wavelets are mathematical functions that decompose data into different frequency components and then study each component with a resolution matched to its scale. Wavelets have emerged as powerful new mathematical tools for analysis of complex datasets. The Fourier transform provides representation of an image based only on its frequency content. Hence this representation is not spatially localized while wavelet functions are localized in space. The Fourier transform decomposes a signal into a spectrum of frequencies whereas the wavelet analysis decomposes a signal into a hierarchy of scales ranging from the coarsest scale. Hence Wavelet transforms which provides representation of an image at various resolutions is a better tool for feature extraction from images.
6.3.2. Discrete wavelets transform (DWT)

The DWT is an implementation of the wavelet transform using a discrete set of the wavelet scales and translation obeying some defined rules. For practical computations, it is necessary to discretise the wavelet transform. The scale parameters are discretized on a logarithmic grid. The translation parameter ($\tau$) is then discretized with respect to the scale parameter, i.e. sampling is done on the dyadic (as the base of the logarithm is usually chosen as two) sampling grid. The discretized scale and translation parameters are given by, $s = 2^{-m}$ and $t = n2^{-m}$, where $m, n \in Z$, the set of all integers. Thus, the family of wavelet functions is represented in Eq. (1) and (2),

$$\psi_{m,n}(t) = 2^m \psi(2^m t - n)$$

$$W\psi(a,b) = \int_{-\infty}^{\infty} f(x) * \psi_{a,b}(t) dx$$
In case of images, the DWT is applied to each dimension separately. This results in an image $Y$, decomposed into a first level approximation component $Y_{a1}$ and detailed components $Y_{h1}$, $Y_{v1}$ and $Y_{d1}$ corresponding to horizontal, vertical and diagonal details. Figure 6.1 depicts the process of an image being decomposed into approximate and detailed components.

The approximation component ($Y_a$) contains low frequency components of the image while the detailed components ($Y_h$, $Y_v$ and $Y_d$) contain high frequency components. Thus,

$$Y = Y_{a1} + \{ Y_{h1} + Y_{v1} + Y_{d1} \}$$

(7)

At each decomposition level, the length of the decomposed signals is half the length of the signal in the previous stage. Hence the size of the approximation component obtained from the first level decomposition of an NXN image is $N/2 \times N/2$, second level is $N/4 \times N/4$ and so on. As the level of decomposition is increased, compact but coarser approximation of the image is obtained. Thus, wavelets provide a simple hierarchical framework for interpreting the image information.

**6.3.3. Coarse Image Segmentation**

Colour image segmentation is useful in many applications. From the segmentation results, it is possible to identify regions of interest and objects in the scene, which is very beneficial to the subsequent image analysis or annotation. However, due to the difficult nature of the problem, there are
few automatic algorithms that can work well on a large variety of data. The problem of segmentation is difficult because of image texture. If an image contains only homogeneous color regions, clustering methods in color space are sufficient to handle the problem. In reality, natural scenes are rich in color and texture. It is difficult to identify image regions containing color-texture patterns. The approach taken in this work assumes the following:

- Each region in the image contains a uniformly distributed colour-texture pattern.
- The colour information in each image region can be represented by a few quantized colors, which is true for most color images of natural scenes.
- The colours between two neighbouring regions are distinguishable - a basic assumption of any color image segmentation algorithm.

6.4. Image Segmentation

6.4.1. K-Means clustering based Coarse Image Segmentation

K-Means clustering algorithm is a well-known unsupervised clustering technique to classify any given input dataset. This algorithm classifies a given dataset into discrete k-clusters using which k-centroids are defined, one for each cluster. The next step is to take each point in the given input data set and associate it to the possible nearest centroid. This process is
repeated for all the input data points, based on which next level of clustering and the respective centroids are obtained. This procedure is iterated until it converges. This algorithm minimizes the following objective function.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{k} \|x_i^j - c^j\|^2$$

Where $\|x_i^j - c^j\|^2$ is a chosen distance measure between a data point $(x_i)$ and the cluster centre, $c_j$ is an indicator of the distance of the $k$ data points from their respective cluster centers. The proposed unsupervised segmentation algorithm uses the principle of K-means clustering.

The proposed technique segments the region of interest (ROI) of an input image (input_img) by an interactive user defined shape of square or rectangle to obtain select_img. Then, the number of bins for coarse data computation (bin size), the size of overlapping kernel to partition (w-size) and the maximum number of clusters for segmentation (max_class) are fed as input data for the computation of coarse data. The coarse data identified by each kernel is aggregated to form the final_coarse_data which is further clustered using the principle of K-means clustering in order to produce the segment_img. The algorithmic description of the proposed technique is given here:
6.4.2. K-Means clustering based Coarse Image Segmentation

Algorithm

1. Read a grayscale image as input_img (Define the area to be segmented as a runtime interactive input. The shape of the selection can either be a square or a rectangle)
   
2. Let select_img is the selected subimage of input_img

Assign:

a. binsize=5 (number of bins for coarse data computation)

b. wsize= 7 (wsize is the size of overlapping kernel to partition the select_img)

c. max_class= 3 (maximum number of clusters for segmentation)

3. Repeat step 5 and 6 in algorithm until the select_img is read

4. Read select_img in the order of (wsize*wsize) as window_img

5. Compute coarse_img for window_img as coarse_win_data

6. Aggregate coarse_win_data for select_img as final_coarse_data

7. Cluster final_coarse_data using K-means clustering technique using max_class in order to obtain segment_img
Figure 6.2. (a) Coarse Segmented MRI Image based on above algorithm (b) ROI segmented image of White Lesions

This algorithm can segment an object either fully or partially based on user’s choice. If the image has a background and object(s) then it partitions the object from the background and displays its coarse image. If the image has no background, then the segmented image reveals the inner details of the object.

6.5. Trained Classifiers

In this study, apart from deep learning based on Restricted Boltzmann machines and extreme machine learning based on Single hidden Layer Feed forward Neural Network (SLFN) architecture as classifiers, several other classifiers are also examined in terms of accuracy and performance, including K-nearest neighbor, SVM, Naive Bayes, MultiboostAB, RotationForest, VFI, J48 and Random Forest.
J48 is an implementation of C4.5 algorithm that produces decision trees from a set of labeled training data using the concept of information entropy. It examines the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data into smaller subsets. To make the decision, the attribute with the highest normalized information gain is used. The KNN algorithm compares the test sample with the available training samples and finds the ones that are more similar (“nearest”) to it. When the k-nearest training samples are found, the class label in majority is assigned to the new sample. Learning in the VFI algorithm is achieved by constructing feature intervals around each class for each attribute (basically discretization) on each feature dimension. Class counts are recorded for each interval on each attribute and classification is performed by a voting scheme.

The Naïve Bayesian Classifier assumes that features are independent. Given the observed feature values for an instance and the prior probabilities of classes, the a posteriori probability that an instance belongs to a class is estimated. The class prediction is the class with the highest estimated probability. The SVMs first map the attribute vectors into a feature space (possibly with higher dimensions), either linearly or nonlinearly, according to the selected kernel function. Then, within this feature space, an optimized linear division is sought; i.e., a hyper plane is constructed which separates two classes (this can be extended to multiple classes).
MultiBoosting is an extension to the highly successful AdaBoost technique for forming decision committees. MultiBoosting can be viewed as combining AdaBoost with bagging. It is able to harness both AdaBoost's high bias and variance reduction with bagging's superior variance reduction. Using C4.5 as the base learning algorithm, Multi-boosting is demonstrated to produce decision committees with lower error than either AdaBoost or bagging significantly more often than the reverse over a large representative cross-section of data. To reduce the dimensionality of the large set of features of dataset, in our study, we propose the use of three optimal attribute selection algorithms: correlation based feature selection (CFS) method, which evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them, secondly an approach based on wrappers which evaluates attribute sets by using a learning scheme. Also in this study, three search methods are also examined: the Best First, Greedy Stepwise and Scatter Search algorithms. These search algorithms are used with attribute selector’s evaluators to process the greedy forward, backward and evolutionary search among attributes of significant and diverse subsets. In total, these feature selection algorithms were tested to select nearly 10 optimal and significant features out of 1024 features.

Feature Selection
In machine learning, during the training of the classifiers, if the numbers of image features are large, it can lead to ill-posing and over fitting, and reduce the generalization of the classifier. One way to overcome this problem is to reduce the dimensionality of features. To reduce the dimensionality of the large set of features of dataset, in our study, we propose the use of three optimal attribute selection algorithms: correlation based feature selection (CFS) method, which evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them, secondly an approach based on wrappers which evaluates attribute sets by using a learning scheme. Also in this study, three search methods are also examined: the Best First, Greedy Stepwise and Scatter Search algorithms. These search algorithms are used with attribute selector’s evaluators to process the greedy forward, backward and evolutionary search among attributes of significant and diverse subsets. In total, these feature selection algorithms were tested to select nearly 10 optimal and significant features out of 1024 features.

When we do PCA, we need to do an eigen-decomposition of the covariance matrix. The procedure of PCA is as follows:

1. Compute the mean:

\[ \bar{x} = \frac{1}{m} \sum_{i=1}^{m} x_i \]
2. Generate the zero-mean data matrix:

\[ x_i = \bar{x}_i - \bar{x} \]

\[ A = (x_1, x_2, \ldots, x_m) \]

3. Construct the covariance matrix:

\[ C = AA^T \]

The covariance matrix \( C \) is symmetric and positive definite. So the eigenvalues of \( C \) is real and non-negative.

4. Eigen-decomposition:

The eigenvalues \( \lambda_i \) and the eigenvectors \( v_i \) of \( C \) satisfy

\[ Cv_i = \lambda v_i \]

5. So we have the eigen-decomposition of the covariance matrix:

\[ C = V \Lambda V^{-1} = V \Lambda V^T \]

6.6. Experiments and Results

6.6.1. Level of wavelet decomposition

We obtained wavelet coefficients of 60 brain MR images, each of whose size is 256 X 256. Level-1 HAR wavelet decomposition of a brain MR image produces 16384 wavelet approximation coefficients; while level-2 and level-3 produce 4096 and 1024 coefficients, respectively. The
preliminary experimental analysis of the wavelet coefficients through simulation in Matlab 7.10., showed that level-2 features are the best suitable for different classifiers, whereas level-1 and level-3 features results in lower classification accuracy. We also use the DAUB-4 (Daubachies) as mother wavelets to get decomposition coefficients of MRI images at Level 2 for comparative evaluation of two wavelets decomposition methods in terms of classification accuracy.

![Image of proposed model](image)

**Fig 6.3. Proposed Model of using various classifiers and attribute selection for finding the abnormality in T1 MRI Brain Images**

### 6.6.2. Attribute Selection and Classification

The second step after Wavelet decomposition of MRI images is to select significant features among whole set of coefficients. Table 6.1 shows the accuracy of classification (percentage of correctly classified samples), True Positive Rate (TP), False Positive Rate (FP) and Average Accuracy (ACC) over all pair-wise combination with different feature evaluators and search algorithms with respect to multi-class classification.
Table 6.1 shows the performance of several learning classifiers, including K-nearest neighbor, SVM, Naive Bayes, MultiboostAB, Rotation Forest, VFI, J48 and Random Forest. Among the pair-wise classification, the lowest accuracy is observed for the classification VFI classifiers of 74.16% and the highest accuracy for the classification by Rotational forest of 97.06%. Moreover, the combination of CFS feature evaluator with the Best First search algorithm gives the highest classification accuracy compared to other feature evaluators and search algorithms.

While Table 6.1 shows the performance of individual classifiers, Table 6.2 defines the comparative results of various combined search techniques and feature evaluators using above prescribed classifiers. Table 6.3 compares the proposed method against a popular dimensionality reduction method, known as Principal Component Analysis (PCA). PCA applies an orthogonal linear transformation that transforms data to a new coordinate system of uncorrelated variables called principal components. We have applied PCA to reduce the number of attributes or feature to 18 attributes and plotted the ROC curves using several above mentioned learning classifiers in terms of True Positive and False Positive Rate, as seen in Figure 6.3. As can be seen in Figure 6.3, the ROC curves for all the trained learning classifiers examined in this study, the curves lie above the diagonal line validating better classification rather than any other random classifiers. The optimal points of various trained classifiers are indicated.
by bold solid points as False Positive rate (FP) and True Positive rate (TP).
These optimal points in ROC curves show the maximum optimal value (FP, TP) of all trained classifiers.

Table 6.1. Various Classifiers comparision with respect to Average Classification Accuracy(%) and other parameters

<table>
<thead>
<tr>
<th>Classifiers</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>(ACC %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>0.935</td>
<td>0.917</td>
<td>0.826</td>
<td>0.853</td>
<td>0.839</td>
<td>91.04</td>
</tr>
<tr>
<td>SVM</td>
<td>0.912</td>
<td>0.912</td>
<td>0.831</td>
<td>0.912</td>
<td>0.87</td>
<td>91.17</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.868</td>
<td>0.916</td>
<td>0.828</td>
<td>0.868</td>
<td>0.847</td>
<td>86.76</td>
</tr>
<tr>
<td>MultiboostAB</td>
<td>0.91</td>
<td>0.91</td>
<td>0.829</td>
<td>0.91</td>
<td>0.868</td>
<td>91.04</td>
</tr>
<tr>
<td>Rotation Forest</td>
<td>0.971</td>
<td>0.285</td>
<td>0.971</td>
<td>0.971</td>
<td>0.968</td>
<td>97.06</td>
</tr>
<tr>
<td>VFI</td>
<td>0.742</td>
<td>0.049</td>
<td>0.93</td>
<td>0.742</td>
<td>0.796</td>
<td>74.16</td>
</tr>
<tr>
<td>J48</td>
<td>0.96</td>
<td>0.314</td>
<td>0.958</td>
<td>0.96</td>
<td>0.957</td>
<td>95.98</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.97</td>
<td>0.271</td>
<td>0.97</td>
<td>0.97</td>
<td>0.968</td>
<td>97.01</td>
</tr>
</tbody>
</table>

Table 6.2. Comparison of pair wise combination of various Attribute Selectors and classifiers with respect to ACC (%)

<table>
<thead>
<tr>
<th>Evaluator</th>
<th>Search Algorithm</th>
<th>Classifier</th>
<th>N</th>
<th>ACC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFS</td>
<td>Best First</td>
<td>K-NN</td>
<td>6</td>
<td>91.04</td>
</tr>
<tr>
<td>CFS</td>
<td>Greedy Stepwise</td>
<td>K-NN</td>
<td>2</td>
<td>89.70</td>
</tr>
<tr>
<td>CFS</td>
<td>Scatter Search</td>
<td>K-NN</td>
<td>4</td>
<td>88.23</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Best First</td>
<td>K-NN</td>
<td>5</td>
<td>89.32</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Greedy Stepwise</td>
<td>K-NN</td>
<td>4</td>
<td>87.56</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Scatter Search</td>
<td>K-NN</td>
<td>4</td>
<td>88.20</td>
</tr>
<tr>
<td>CFS</td>
<td>Best First</td>
<td>SVM</td>
<td>6</td>
<td>91.17</td>
</tr>
<tr>
<td>CFS</td>
<td>Greedy Stepwise</td>
<td>SVM</td>
<td>6</td>
<td>89.23</td>
</tr>
<tr>
<td>CFS</td>
<td>Scatter Search</td>
<td>SVM</td>
<td>4</td>
<td>91.04</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Best First</td>
<td>SVM</td>
<td>2</td>
<td>90.65</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Greedy Stepwise</td>
<td>SVM</td>
<td>2</td>
<td>90.65</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Scatter Search</td>
<td>SVM</td>
<td>5</td>
<td>89.56</td>
</tr>
<tr>
<td>CFS</td>
<td>Best First</td>
<td>Naive Bayes</td>
<td>8</td>
<td>86.76</td>
</tr>
<tr>
<td>CFS</td>
<td>Greedy Stepwise</td>
<td>Naive Bayes</td>
<td>8</td>
<td>82.78</td>
</tr>
<tr>
<td>CFS</td>
<td>Scatter Search</td>
<td>Naive Bayes</td>
<td>7</td>
<td>82.12</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Best First</td>
<td>Naive Bayes</td>
<td>4</td>
<td>85.44</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Greedy Stepwise</td>
<td>Naive Bayes</td>
<td>2</td>
<td>85.44</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Scatter Search</td>
<td>Naive Bayes</td>
<td>2</td>
<td>80.12</td>
</tr>
<tr>
<td>CFS</td>
<td>Best First</td>
<td>MultiboostAB</td>
<td>5</td>
<td>91.04</td>
</tr>
<tr>
<td>CFS</td>
<td>Greedy Stepwise</td>
<td>MultiboostAB</td>
<td>5</td>
<td>91.04</td>
</tr>
<tr>
<td>CFS</td>
<td>Scatter Search</td>
<td>MultiboostAB</td>
<td>4</td>
<td>86.54</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Best First</td>
<td>MultiboostAB</td>
<td>5</td>
<td>89.39</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Greedy Stepwise</td>
<td>MultiboostAB</td>
<td>5</td>
<td>90.45</td>
</tr>
<tr>
<td>Wrapper</td>
<td>Scatter Search</td>
<td>MultiboostAB</td>
<td>4</td>
<td>88.76</td>
</tr>
<tr>
<td>CFS</td>
<td>Best First</td>
<td>Rotation Forest</td>
<td>9</td>
<td>97.06</td>
</tr>
<tr>
<td>Classifier</td>
<td>PCA (%)</td>
<td>CFS-Best First (%)</td>
<td>Wrapper-Best First (%)</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>----------</td>
<td>---------------------</td>
<td>-------------------------</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>91.38</td>
<td>91.04</td>
<td>89.32</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>96.24</td>
<td>91.17</td>
<td>90.65</td>
<td></td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>85.63</td>
<td>86.76</td>
<td>85.44</td>
<td></td>
</tr>
<tr>
<td>MultiboostAB</td>
<td>94.52</td>
<td>91.04</td>
<td>89.39</td>
<td></td>
</tr>
<tr>
<td>Rotation Forest</td>
<td>97.06</td>
<td>97.06</td>
<td>93.78</td>
<td></td>
</tr>
<tr>
<td>VFI</td>
<td>77.12</td>
<td>74.16</td>
<td>72.22</td>
<td></td>
</tr>
<tr>
<td>J48</td>
<td>95.34</td>
<td>95.98</td>
<td>95.98</td>
<td></td>
</tr>
<tr>
<td>Random Forest</td>
<td>97.34</td>
<td>97.01</td>
<td>96.25</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3. Comparison using PCA and other feature attribute evaluators in terms of ACC (%)
Figure 6.4. Shows the ROC curve of the above mentioned trained classifiers

Table 6.4 describes the classification results using proposed optimized Pruned Extreme Machine Learning and Deep Machine Learning. In Table 6.4, we show the training time, testing time and classification error using extreme and deep machine Learning. As we can see in the table both learning algorithms are processed with many hidden layers and their evaluations is done in terms of various factors. As depicted in Table 6.4, it clearly shows that deep machine learning plays a major role in reducing the classification error. As deep and extreme machine learning are designed to work on large datasets, it is difficult to compare the performance. However, they result in acceptable accuracy levels, and we are currently examining several other publicly available large MRI datasets for enhancing the
performance of these two novel approaches (Deep learning and Extreme machine learning approaches).

**Table 6.4 Classification results using Extreme Machine Learning and Deep Machine Learning**

<table>
<thead>
<tr>
<th>Hidden Layers</th>
<th>Training Time(s)</th>
<th>Testing Time(s)</th>
<th>Classification Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>Deep Learning</td>
<td>0.56</td>
<td>0.47</td>
<td>0.72</td>
</tr>
<tr>
<td>Pruned Extreme Learning Machines</td>
<td>0.31</td>
<td>0.31</td>
<td>0.61</td>
</tr>
</tbody>
</table>

However, the deep learning networks do not need any particular feature reduction algorithms because of the inherent capability for feature reduction in terms of deep learning (learning through multiple layers). In case of extreme machine learning, the learning proceeds through random assignment of weights and hidden nodes (unlike gradient descent based techniques). Due to this, there is a significant improvement in training and testing time as depicted in Table 6.3.

### 6.7. Conclusions

In this Chapter, we have presented a principled approach for investigating brain abnormalities based on wavelet based feature extraction, PCA based feature selection and deep and optimized Pruned extreme machine learning based classification comparative to various other classifiers. Experiments
on a publicly available brain image dataset show that the proposed principled approach performs significantly better than other competing methods reported in the literature. The classification accuracy of more than 93% in case of deep machine learning and 94% in case of pruned extreme machine learning demonstrates the utility of the proposed method. In this study, we have applied this method only to axial T2-weighted images at a particular depth inside the brain. The same method can be employed for T1-weighted, proton density and other types of MR images. With the help of above approaches, one can develop software for a diagnostic system for the detection of brain disorders like Alzheimer’s, Huntington’s, Parkinson’s diseases etc. Further, the proposed approach uses reduced data by incorporating feature selection algorithms in the processing loop and still provides an improved recognition and accuracy. The training and testing time for the whole study used by deep and extreme machine learning is much less as compared to SVM and other traditional classifiers reported in the literature.
Chapter VII

Email Personalization and User Profiling using
RANSAC Multi Model Response Regression based
Optimized Pruning Extreme Learning Machines and
Gradient Boosting Trees

7.1. Introduction

Personalization is the idea of interactive marketing with respect to the customization of some or all elements of the marketing strategy to an individual level. Personalization is a refined version of customization, where marketing is done automated on behalf of customer’s user’s profiles, rather than customer requests on his own behalf. There is very thin line between customization and personalization which is achieved by leveraging customer level information using analytical tools (Montgomery & Smith, 2009).

E-commerce is growing fast, and with this growth companies are willing to spend more on improving the online experience. In Commerce Software Takes Off (Schmitt, Manning, Paul, & Roshan, 2000), the authors from Forrester Research wrote that online business to consumer retail spending in 1999 was $20.3 billion and estimated to grow to $144 billion by 2003. Global 2500 companies have spent 72% more on e-commerce in 2000 than they did in 1999. Existing sites are using primitive measures, such as page
views, but the need for more serious analysis and personalization is growing quickly with the need to differentiate. In Measuring Web Success (Miceli, Ricotta, & Costabile, 2007), the authors claim that "Leaders will use metrics to fuel personalization" and that "firms need web intelligence, not log analysis." Data mining tools aid the discovery of patterns in data. Until recently, companies that have concentrated on building horizontal data mining modelling tools, have had little commercial success.

Personalization is a key component for adapting a standardized product or service to an individual customer’s needs. Application of personalization fits nicely into notions of Internet to provide a rich environment for well suited interaction and segmentation. The exponential growth of World Wide Web enables new methods of development and design of online information services. Most web development is large and complicated and users receive ambiguous results during web navigation. But on the other hand, rapidly evolving and need of e-commerce marketplaces in World Wide Web anticipate the need of customers to distinct level. Therefore, the requirement for predicting user needs in order to improve the usability and user retention of a Web site can be addressed by personalizing it (Vesanen, 2007). Web personalization is defined as any action that adapts the information or services provided by a Web site to the needs of a particular user or a set of users, taking advantage of the knowledge gained from the
users’ navigational behaviour and individual interests, in combination with the content and the structure of the Web site. The objective of a Web personalization system is to “provide users with the information they want or need, without expecting from them to ask for it explicitly” (Mobasher, Cooley, & Srivastava, 2000).

The core difference between customization and personalization is in fact the customization, wherein, the site and emails can be sent and adjusted to each user’s preferences with respect to its structure and presentation. Every time a registered user logs in on page, their customized page is loaded manually or semi automatically. Compare to personalization’s systems, the content and structure of web site is dynamic which includes

(a) Segmentation and pre-processing of web data

(b) Extraction of correlations between across different kinds of data

(c) Determination of the actions recommended by personalization systems.

Web data are those that can be collected and used in the context of Web personalization. These data are classified in four categories according to Srivastava et al. (Srivastava, Cooley, Deshpande, & Tan, 2000)

- Content data, presented as structured data based on simple text, images and information retrieved from the databases
• Structure data represents the data entities and attributes within web page, as XML tags and hyperlinks connecting one page to another

• Usage data includes usage, clicks, page views, number of visitors, visitor IP address, time and date of access and other attributes that included in customers events while accessing the website

• User profile data includes the information about users which includes demographic information (such as name, age, country, marital status, education, interests etc.) for each user of a Web site, as well as information about users’ interests and preferences. Such information can be acquired during web site account registration and questionnaires during feedback.

Overall, web personalization process is distributed into five modules as follows (Montgomery, Li, Srinivasan, & Liechty, 2004)

7.1.1. User Profiling

User profiling is the process of creating taste profiles to specific customer either explicitly or implicitly. Taste profiles of users created by using demographic information, their interests and their behaviour from web traffic data. Using user’s taste profiles, customization and content of emails marketing and website can be exploited with customer’s specific needs.

A. Log analysis and crunching web traffic data
Log analysis and crunching of web traffic data is achieved by applying data mining techniques

- Extract statistical information and discover interesting usage patterns
- Cluster or Segmenting the users into groups according to their navigational behaviour
- Discover potential correlations between Web pages and user groups.

Using log analysis and analysing web traffic data, it is easy to design user profiles according to their behaviour and patterns on visiting and doing transaction on websites. Data mining and data analysis of web traffic data is the core engine of designing email personalization for each customer. In later section, we will discuss more on finding the optimistic and accurate method based on data mining to predict customer behaviour and on estimated probability estimates, and how email personalization strategy can be designed and implemented in dynamic environment.

7.1.2. Content Management

Content Management is the process of classifying the content into categorical semantic categories of website and emails in order to make information retrieval and presentation better for the users. Content management call is tuned according to the pattern and behaviour for user profiling using data mining techniques which is also a significant part of
email and website personalization. Using personalization techniques, the process is sensitive to the content which changes dynamically on a daily or weekly basis, such as news portals.

7.1.3. Web site publishing

A publishing mechanism is used in order to present the content stored locally in a Web server and/or some information retrieved from other Web resources in a uniform way to the end-user.

7.1.4. Information acquisition and searching

Searching and relevance ranking techniques must be employed both in the process of acquisition of relevant information and in the publishing of the appropriate data to each group of users. Recommendation systems are also implemented where based on analysis of log data; websites recommended the similar products and categories where customers showed the highest interest for higher engagement leading to an increase in ROI.

7.2. Email Personalization

Email personalization is the process of customizing the content and structure of email according to members' specific and individual needs, taking advantage of members' navigational behavior. Email personalization can be followed and employed using content-based filtering, which is based on individual user’s preferences. The recommendation
systems track the member’s behaviour in past and recommend similar items based on historical data

Collaborative filtering rates and ranks deals and promotions and segment this information. Based on probability estimates of ranked deals, email personalization makes an assumption that members have similar behaviour and have ranking-predicted analogous interests.

Rule based filtering is based on information extracted from set of questions. These questions are derived from a decision tree, and based on set of trees, personalized content is sending in emails.

Email personalization strongly relies on statistical and data mining methods to the Web log data, resulting in a set of useful patterns that indicate users’ navigational behaviour. In past, the data mining methods that are employed are: association rule mining, sequential pattern discovery, clustering, and classification. This knowledge is then used from the system in order to personalize the site according to each user’s behaviour and profile.

7.3. **Gradient Boost Trees**

Gradient Tree Boosting or Gradient Boosted Regression Trees (GBRT) (Mazumder, Friedman, & Hastie, 2011) is a generalization of boosting to arbitrary differentiable loss functions. GBRT is an accurate and effective off-the-shelf procedure that can be used for both regression and
classification problems. Gradient Tree Boosting Models (GBM) are used in a variety of areas including Web search ranking and ecology.

The advantages of GBRT are:

- Natural handling of data of mixed type (= heterogeneous features)
- Predictive power
- Robustness to outliers in output space (via robust loss functions)

GBM trains many models turn by turn and each new model gradually minimizes the loss function of the whole system using Gradient Descent method. Assuming each individual model $i$ is a function $h(X; p_i)$ (which we call “base function” or “base learner”) where $X$ is the input and $p_i$ is the model parameter. Now let’s choose a loss function $L(y, y')$ where $y$ is the training output, $y'$ is the output from the model. In GBM, $y' = \sum_{i=1}^{M} \beta_i h(X, p_i)$, where $M$ is the number of base learners. This can be more defined as

$$\beta, P = \arg \min_{(\beta, P)} L(y, \sum_{i=1}^{M} \beta_i h(X; p_i))$$ (1)

However this is not easy to achieve optimal parameters. Instead we can try a greedy approach that reduces the loss function stage-by-stage:

$$\beta_n, P_n = \arg \min_{\beta, P} L(y, F_{n-1}(X) + \beta h(X; p))$$ (2)
And then we update:

\[ F_m = F_{m-1}(X) + \beta_m h(X; p_m) \]  

(3)

In order to reduce the loss function, \( L(y, F_m) \), an obvious way is to step toward the direction where the gradient of \( L \) descents:

\[ -g_m(X) = \left[ \frac{\partial L(y, F(X))}{\partial F(X)} \right] F(X) = F_{m-1}(X) \]  

(4)

However what we want to find out is \( \beta_m \) and \( p_m \) so that

\[ F_m - F_{m-1} - g_m \]  

which is defined as

\[ \beta_m, p_m = \arg \min_{\beta, p} \sum_{i=1}^{N} \left[ -g_m(x_i) - \beta h(x_i; p) \right]^2 \]  

(5)

where \( h_m(x) \) are the basis functions which are usually called weak learners in the context of boosting. Gradient Tree Boosting uses decision trees of fixed size as weak learners. Decision trees have a number of abilities that make them valuable for boosting, namely the ability to handle data of mixed type and the ability to model complex functions.

At each stage the decision tree \( h_m(x) \) is chosen to minimize the loss function \( L \) given the current model \( F_{m-1} \) and its fit \( F_{m-1}(x_i) \).

Gradient Boosting attempts to solve this minimization problem numerically via steepest descent: The steepest descent direction is the
negative gradient of the loss function evaluated at the current model \( F_{m-1} \) which can be calculated for any differentiable loss function. The algorithms for regression and classification only differ in the concrete loss function used.

During learning via gradient boosting trees, regularization strategy can be implemented that scales the contribution of each weak learner by a factor of \( V \):

\[
F_m(x) = F_{m-1}(x) + \nu \gamma_m h_m(x)
\]  \hspace{1cm} (6)

The parameter \( \nu \) is also called the learning rate because it scales the step length of the gradient descent procedure; it can be set via the learning rate parameter. The parameter learning rate strongly interacts with the parameter n-estimators, the number of weak learners to fit. Smaller values of learning rate require larger numbers of weak learners to maintain a constant training error.

Often features do not contribute equally to predict the target response; in many situations the majority of the features are in fact irrelevant. When interpreting a model, the first question usually is: what are those important features and how do they contributing in predicting the target response?

Individual decision trees intrinsically perform feature selection by selecting appropriate split points. This information can be used to measure the importance of each feature; the basic idea is: the more often a feature is
used in the split points of a tree, the more important that feature is. This notion of importance can be extended to decision tree ensembles by simply averaging the feature importance of each tree.

In further section, we discuss about proposed optimized extreme learning machines based on RANSAC regularization as base estimator in gradient boosting trees.

The complete classification model is implemented on OURDEAL database to design user profiles and email personalization for their marketing strategy. The model spits up the customer plan each morning at 8.00 am morning with personalized customer plan for each customer and the personalized deals can be send to each customers with respect to their web usage behaviour on website, as well as the clicks and open rate of emails. Due to sparse database available for new and non-engaging users, model works perfectly in terms of defining the personalized emails. The whole complete strategy is briefly defined in next few sections. For the sake of continuity, extreme learning machines is briefly described again.

### 7.4. Extreme Learning Machines

(Huang et al., 2006) proposed a new novel algorithm as Extreme Machine Learning (ELM) for single hidden layer feed forward neural network which has less computational time and faster speed even on large datasets. The main working core of ELM is random initialization of weights rather than
learning through slow process via iteratively gradient based learning as compared to back-propagation. In Extreme machine learning, the number of hidden nodes and their weights are randomly assigned, which distinguishes the linear differentiable between the output of hidden layer and output layer. The output weights can be determined by linear least square solution of hidden layer output through activation function and the data samples targets.

For $N$ arbitrary distinct samples $(x_i, t_i)$, where $x_i = [x_{i1}, x_{i2}, ..., x_{in}]^T \in \mathbb{R}^n$ and $t_i = [t_{i1}, t_{i2}, ..., t_{im}]^T \in \mathbb{R}^m$, standard SLFNs with $N$ hidden nodes and activation function $g(x)$ are mathematically modelled as

$$\sum_{i=1}^{N} \beta_i g_i(x_j) = \sum_{i=1}^{N} \beta_i g(w_i \cdot x_j + b_i) = 0_j,$$ (7)

$j = 1, ..., N$,

where $w_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ is the weight vector connecting the $i^{th}$ hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im}]^T$ is the weight vector connecting the $i^{th}$ hidden node and the output nodes, and $b_i$ is the threshold of the $i^{th}$ hidden node. $w_i \cdot x_j$ denotes the inner product of $w_i$ and $x_j$. The output nodes are chosen linear in this work.
That standard SLFNs with \( N \) hidden nodes with activation function \( g(x) \) can approximate these \( N \) samples with zero error means that

\[
\sum_{j=1}^{\hat{N}} \| o_j - t_j \| = 0, \quad \text{i.e., there exist } \beta_i, w_i \text{ and } b_i \text{ such that}
\]

\[
\sum_{i=1}^{\hat{N}} \beta_i g_i \left( w_i \cdot x_j + b_i \right) = t_j, \quad j = 1, \ldots, N.
\]

The above \( N \) equations can be written compactly as

\[
H \beta = T,
\]

where

\[
H \left( w_1, \ldots, w_{\hat{N}}, b_1, \ldots, b_{\hat{N}}, x_1, \ldots, x_{\hat{N}} \right)
\]

\[
= \begin{bmatrix}
g \left( w_1 \cdot x_1 + b_1 \right) & \cdots & g \left( w_{\hat{N}} \cdot x_1 + b_{\hat{N}} \right) \\
\vdots & \ddots & \vdots \\
g \left( w_1 \cdot x_{\hat{N}} + b_1 \right) & \cdots & g \left( w_{\hat{N}} \cdot x_{\hat{N}} + b_{\hat{N}} \right)
\end{bmatrix}_{\hat{N} \times \hat{N}}
\]

\[
\beta = \begin{bmatrix}
\beta_1^T \\
\vdots \\
\beta_{\hat{N}}^T
\end{bmatrix}_{\hat{N} \times m}, \quad T = \begin{bmatrix}
t_1^T \\
\vdots \\
t_{\hat{N}}^T
\end{bmatrix}_{\hat{N} \times m}
\]

\( H \) is called the hidden layer output matrix of the neural network; the \( \hat{i}^{\text{th}} \) column of \( H \) is the \( \hat{i}^{\text{th}} \) hidden node output with respect to inputs \( x_1, x_2, \ldots, x_{\hat{N}} \).
7.5. Optimized Extreme Learning Machines as Base Estimators for Gradient Boosting Trees

To resolve the limitations of ELM, constructive and heuristic approaches have proposed in the literature. In recent years, regularization or penalty approach seems to be popular in resolving the ELM limitations. Since there is linear behaviour between hidden layer and output layer in ELM architecture, there is a problem associated with linear regression, and regularization helps to reduce the number of predictors in hidden layer by using sparse model.

7.5.1. RANSAC Multi Model Response Regularization

7.5.1.1. RANSAC Multi Model Response Regularization for Regression problems

To implement the RANSAC on regression problems, we proposed a RANSAC multi model response regularization which implements the sequential RANSAC on multiple models. We take out the outliers from data, which in our case, are the irrelevant hidden nodes as predictor variables, and $H$ is the hidden matrix as input from equation. In our case, the output weights follow a linear regression between hidden and output layer defined as

$$Y = mx + \epsilon \quad \quad (13)$$
Where Y is the output of instances of data, m is the predictor’s weights or slope and x is the input data and c is the constant. \( Y = \text{Output weights}\times H + \epsilon \)

\[
H = \begin{bmatrix}
g(w_1 \cdot x_1 + b_1) & \cdots & g(w_N \cdot x_1 + b_N) \\
\vdots & \ddots & \vdots \\
g(w_1 \cdot x_N + b_1) & \cdots & g(w_N \cdot x_N + b_N)
\end{bmatrix}_{N \times \hat{N}}
\]

Where

\[
\text{OutputWeights} = \text{RANSAC} - \text{Multi} \left[ \begin{array}{c}
g(w_1 \cdot x_1 + b_1) \\
\vdots \\
g(w_1 \cdot x_N + b_1) \\
\end{array} \right] \times \begin{array}{c}
g(w_N \cdot x_1 + b_N) \\
\vdots \\
g(w_N \cdot x_N + b_N)
\end{array}_{N \times \hat{N}}
\]

\[
\text{CS} = \text{RANSAC} \left( \sum_{w=1}^{m} \left( \sum_{j=1}^{n} D_{w_j} \right) \right)
\]

\( D_{xw_j} = \{x_{11}, x_{12}, \ldots, x_{wj}\} \) is the sets of data H with w\(^{th}\) observations in rows and j\(^{th}\) as hidden nodes predictors in columns of D matrix. For regression problems, sequential RANSAC implements on the set of all inliers, D1 that are generated by W different models where \( W_m = \text{rand}(D_M) \). The numbers of models are randomly generated using 20% of the input data.

To estimate the parameters of W models, each one is represented by k dimensional parameter vector \( \theta_w \) at each iteration iter. CS is estimated using MSS of each W model. The iteration run M times which is calculated before after removing the inliers from data D. the total number of inliers at iteration iter is less than total number of inliers at iteration iter-1. The complete formulation of multiple RANSAC response is defined as
The set of all inliers $D$ is generated by $W$ different models, which has cardinality $CS$ as

$$N_i = (N_{i,1} + N_{i,2} + \ldots\ldots N_{i,W})$$  \hspace{1cm} (17)

Let $\mathcal{M}_w(\theta_w)$ defines the manifold of dimension $k_w$ of all points with respect to parameter $\theta_w \in \mathbb{R}^{k_w}$ for the specified model for $1 \leq w \leq W$ with a subset $S_w$ from $D$ of $k_w$ elements at iteration $i$ called minimal sample set (MSS). To estimate the parameters of $W$ models, each one represented parameter vector $\theta_w$. At each $i$ iterations, MSS for each $W$ model is defined and $CS$ is estimated removing all outliers.

The proposed RANSAC multi model response regularization for binary and multiclass problems for ELM is implemented using one against all (OAA) method. As in OAA method, $j$ binary classifiers are constructed in which all the training examples will be used a teach time of training. The training data having the original class label $j_n = (1\ldots\ldots n)$ have each $j_n$ elements of positive one class and the remaining training data will be of zero class, creating $j_n$ models implementing proposed RANSAC multi model response regularization on $(j_n)$ binary classes. Finally, CS defined as $S'((\theta))$ of $j(n)$ classes is computed as

$$S'((\theta))_j = \sum_{j=1}^{n} S^{(i)}\theta_j \cup S^{(i-1)}\theta_j$$  \hspace{1cm} (18)
For this, consider the ELM for multi-class classification problem, formulated as $k$ binary ELM classification problem with the following form:

$$
H_{w_j} = y_1 \ldots H_{w_j} = y_j;
$$

Where for each $j$, $w_j$ is the output weights from the hidden layer to the output layer with output vector $y_j = (y_{i_1}, \ldots, y_{i_n})' \in \mathbb{R}_m$. Thus the output of the hidden layer as $H$ hidden matrix defines with respect to multiclass binary classifiers as

$$
H_j = \sum_{i=1}^{n} H * Y \left\{ \begin{array}{c} Y_j = 1 \\ \text{o} \end{array} \right\}
$$

(20)

where $H$ is the hidden layer output matrix and $Y$ is the $j$ binary classes with $m^{th}$ observations of training data and $n$ binary classes as columns vectors. Thus, we get $H_j$ hidden matrix where each $H_j$ belong to each binary class and RANSAC multi response regularization is implemented to acquire CS for each binary class as $S'(\theta)$. It can be concluded that RANSAC multi response regularization for binary and multiclass problems work in similar fashion as OAA-ELM with $j$ binary classes with a difference of $j^{th}$ label with positive class and rest other classes with -1 class.
7.5.1.2. Australian E-commerce OurDeal Database

Email personalization and user profiling model is implemented on customer database which is designed and based on an Australian e-commerce website ourdeal.com.au. Ourdeal website sells deals and promotions in different cities and states of Australia with respect to multiple categories like product, Travel etc. To access the deals on website, a member has to login with his personal account details which includes email address,, demographic information as gender, postcode etc. After login through his/her account, members can purchase vouchers straight away on ourdeal website and can claim by doing payment online. To promote those live deals on websites to broad range of loyal users and new users, daily mails are sent to user verified email address three times daily which includes customize selection of deals and promotions based on user online behaviour such as clicks, opens etc. and taste profiles designed for each user.

Thus the selection of deals from pools of deals to each customer with respect to their online usage behaviour on website and demographic information constitutes the email personalization strategy.

To implement and design the email personalization model, Figure 1 depicts the flowchart showing extracting data from data warehouse to provide customer plan for each member on daily basis.
As we can see in the Figure 1, member’s data is extracted from redshift data warehouse using sql queries and converted into python pandas data frame. As can be seen, we have a variety of members, split into historical transactional data for old and engaged members, and sparse data for new and non-engaged members. To make our model predictions better based on historical data of old and members with higher engagement levels, we exclude the new and non-engaged members to reduce sparse information. For those members, whose data samples were included, the complete demographic and personnel information was extracted for each member in the sample along with deal traffic data which includes member’s website usage as clicks, opens etc. Deals combinations are made for each members based on deals demographic information matched with member’s demography. These members deals combination are fed into MD data frame and website traffic data for each members along with calculated ratios based on customer events. We also penalize the customer events with respect to timeline so as to include the data based on latest 25 days interaction. This customer events information and penalizing the category along with calculated events affinity ratios are fed into MC data frame. Later, both MD and MC data frame are merged on primary key as memberid and dealid and feed into classification model based on gradient boost trees with base estimator as proposed RANSAC optimized Extreme Learning Machines. The classifier is training on 60% of training data and tested on 40% of testing data. The classifier is tuned with respect to
different parameters to get better classification accuracy and less mean square error.

The set of parameters are selected using cross fold validation upto 10 folds and best model is picked up with defined set of parameters with higher ROC. Based on the trained classifier, top high rank deals are selected and for each member personalized customer plan is generated with set of deals based on customer needs. Selected deals in the customer plans are sent as personalized emails and the model is updated weekly to fulfill the dynamic personalization for each customers. In this study, the proposed model run on 1 million members database with 45 attributes as features such as gender, distance, clicks, opens etc. Comparative analysis is done among different classifiers on the database with the proposed model in terms of better accuracy rate, true positive and false positive rate and confusion matrix.
The model also predicts the probability estimates of each deals with respect to members about chances of clicking the deal or not. Estimated probability estimates are compared with observed probability estimates using different classifiers. The model is written in python using Spyder as IDE with numpy, pandas and sklearn libraries. The model is tested on production box with configuration of intel i5 processor with 32 GB of RAM using windows server 2008. Python scripts are integrated with redshift data warehouse using postgresql queries to access and extract data from data warehouse.
warehouse. After the customer plan is generated, different APIs are used to define jobs and designing templates for personalized emails to selected members.

In later sections, distribution of data and experimental results are explained with comparative analysis with benchmark high performance classifiers.

7.6 Experimental Results

To implement the email personalization, firstly data is extracted from data warehouse and fed into pandas Dataframe in python. Sql queries are integrated in python for accessing the data from redshift data warehouse and uploading the final plan for email personalization. Figure 2 represent the distribution of member’s data with respect to subscribers and non-subscribers in 1 year tenure scale. Subscribers and non-subscribers in the figure represent members who are subscribers for receiving emails from their starting tenure until this experiment. Non subscriber are the members which unsubscribes during this experimental study from their starting tenure. Global parameters are initialized along with exception handling foe extracting data from datawarehouse. As we can see in figure 2, distribution of data seems to be normal distribution with skewed towards right. Figure 3 depicts the member’s data density distribution with respect to subscriber and non-subscribers in different states of Australia as Sydney, Melbourne ranked in scale of 1-10. As we can see in all figures, there is a very thin
line to differentiate members behaviour in terms of whether they will keep subscribe the emails or unsubscribe it.

Figure 7.2: Density Distribution of member’s data with respect to subscribers and non-subscribers in 1 year tenure scale

Figure 7.3: Density Distribution of member’s data with respect to subscribers and non-subscribers in 1-10 location scale
Figure 7.4: Distribution of deviance for training and testing data for 0-300 iterations. With regularization and using optimized extreme learning machine as base estimators for gradient boosting trees.

Figure 7.5: Distribution of deviance for training and testing data for 0-300 iterations with no regularization and no base estimators for gradient boosting trees.
Figure 7.6: Feature importance of multiple attributes as features for member’s data fed into grad boost classifier

Figure 4 depicts the comparison of training data and testing data variance for 300 iterations using regularization and proposed optimized extreme learning machines as base estimators for gradient boosting trees. As we can see in the Figure 4, there is less variance in training and testing data using optimized extreme learning machines as base estimator for grad boost classifiers. Compared to Figure 4, where we do not use regularization and base estimators, grad boost classifiers by itself doesn’t perform well in terms of low deviation and the model is over fitted creating huge difference of deviation between training and testing data. In such case, regularization with optimized RANSAC extreme learning machines performs significantly well with grad boost trees in terms of less deviation among training and testing data.

Figure 7.5 depicts the feature importance of multiple attributes as features for member’s data fed into grad boost classifier. Often features do not
contribute equally to predict the target response; in many situations the majority of the features are in fact irrelevant. Individual decision trees intrinsically perform feature selection by selecting appropriate split points. Gradboost trees also provide the feature importance among multiple attributes with respect to target which help in taking out the non-relevant features.

Table 7.1: Comparative analysis using grad boost classifier with proposed base estimator and non-base estimator

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Mean Square Error (MSE)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Grad-Boost</td>
<td>Grad-boost with ORELM</td>
</tr>
<tr>
<td>100</td>
<td>0.75</td>
<td>0.23</td>
</tr>
<tr>
<td>200</td>
<td>0.56</td>
<td>0.19</td>
</tr>
<tr>
<td>300</td>
<td>0.45</td>
<td>0.11</td>
</tr>
</tbody>
</table>

Table 7.1 depicts the comparative analysis of grad boost classifier on ourdeal database using base estimators and non-base estimators. As we can see from Table, as iterations increases from 100 to 300, Grad boost classifier with optimized RANSAC pruned extreme learning machines (OPRELM) perform better than grad boost with linear regression (LR) as base estimators. The ROC curve using base estimator as OPRELM is estimated as 0.96 and confusion matrix shows less false positive and false negative rate. Later, deals recommended by the classifiers are ranked with respect to click rate and send to each member with personalized deals chosen by classifier.
7.7 Conclusions

Email personalization is the process of customizing the content and structure of email according to member’s specific and individual needs, taking advantage of member’s navigational behaviour. In this Chapter, we present the proposed architectural design for email personalization using ourdeal database based on grad boost with optimized pruned extreme learning machines as base estimators. We also conducted a depth dive in data analysis to find each member’s behaviour and important attributes which plays a significant role in increasing clicks rates in personalized emails.

After data is extracted using step up process according to defined architectural design, we treat the data with grad boost classifier using optimized pruned extreme learning machine and linear regression as based estimator. Experimental results showed that as the iterations increased from 100 to 300, mean square error is much lower compare to grad boost with linear regression as base estimators. We can concluded that our proposed method works much better in predicting customers behaviour on deals send in personalized emails. Higher accurate model helps us to find out better customer behaviour and design the customer plan according to accurate customer’s needs.

Currently this model is implemented in ourdeal, and every morning set of deals are selected using proposed method for personalized emails and
defining user profiles. In future work, we will deploy this strategy of email personalization and user profiles using multi-attributes telecommunication databases at OPTUS. Further study will be extended by deploying email personalization and defining user profiles at OPTUS using multiple data sources as telecommunication data, demographic, social media to provide services at personalized level.
CHAPTER VIII

Conclusions and Future Work

Feed-forward neural networks have been extensively used in many fields due to their ability: (1) to approximate complex nonlinear mappings directly from the input samples; and (2) to provide models for a large class of natural and artificial phenomena that are difficult to handle using classical parametric techniques. On the other hand, there is a lack of faster learning algorithms for neural networks. The traditional learning algorithms are usually far slower than required. It is not surprising to see that it may take several hours, several days, and even more time to train neural networks by using traditional methods.

The accuracy and performance of machine learning and statistical models are still based on tuning certain parameters and optimization for generating better predictive models of learning that is based on the training data. Larger datasets and samples are also problematic, due to increase in computational times, complexity and bad generalization due to outliers.

In this thesis, using the motivation from extreme learning machine (ELM), we propose a novel algorithm, the RANSAC multi model response regularization for multiple models to prune the large number of hidden nodes to acquire better optimality, generalization and classification accuracy of the network in ELM. Experimental evaluation of the proposed
algorithm was done on different benchmark datasets and real time problems such as Protein Folding Problem, Early diagnosis of brain abnormalities in MRI images and user profiling and personalization using e-commerce data.

First three chapters of the thesis discuss the introductory section about machine learning, detailed description of Artificial Neural Networks and its current hybrid variants, with the third chapter covering past work done on hybrid ANNs such as cascade correlation neural network, constructive neural network, and basic ELM model as literature review.

In fourth chapter, we propose an annular ELM based on RANSAC multi model response regularization to optimally prune the hidden nodes in a network and improve better generalization and classification accuracy. Optimally pruned annular extreme leaning machine (OP-AELM) algorithm using RANSAC multi model response regularization, is an extension of original ELM algorithm with pruning of neurons to design optimal neural architecture removing irrelevant hidden nodes to acquire optimal tuned parameters. Regularized version of least squares regression with several penalties on coefficient vector are used to remove the irrelevant or low relevance hidden weights to achieve compact neural networks. Experimental evaluation was conducted using an extensive comparative study with the proposed RANSAC multi model response regularization
based annular ELM network on different benchmark datasets for binary and multiclass classification and regression problems.

It can be concluded from experimental results that proposed RANSAC multi model response regularized based annular ELM performs significantly well in terms of higher classification accuracy with optimally pruned hidden units. Further, the proposed algorithm is faster compared to other algorithms in the study, since it implements the ELM with less pruned hidden units without sacrificing the higher generalization capability of ELM network. Future work can be extended by stacking the OP-AELM using deep machine learning convolutional networks and autoencoders as a semi supervised machine learning algorithm to optimise the cost function with less relative mean square error on large datasets using GPU hardware. There is also a potential opportunity to extend this algorithm in designing graphical user interface (GUI) open source platform so as to integrate with python and R plugins to use it in production datasets, for example – in telecommunications, image processing areas.

In fifth chapter, we used the annular optimized pruned extreme learning machines on protein folding problem which is one of the challenging and most important problems in the area of bioinformatics. In contrast to protein folds prediction, it’s very hard to classify its various folds with its different amino acids attributes due to the limited training data availability. In this study, we compared various classifiers and proposed an improved
algorithm and highly accurate hybrid ensemble classifiers based on Support Vector Machines (SVM and pruned annular Extreme Learning Machines for protein folding recognition). Our proposed classifier involves dimensionality reduction using PCA and LDA prior to classification which results in significant improvement in recognition accuracy. The combined PCA-SVM and LDA-SVM classifiers results in an accuracy of (65.32%) and (68.12%), as compared to SVM (56.70%) and other above mentioned classifiers. The similar PCA/LDA feature selection for annular Extreme Learning Machines also results in improvement in recognition accuracy up to (77.67%) and (82.45%). We also compare our proposed classifiers using multiple kernels approach resulting in promising results. These results seem to be positive for classification for protein sequences as compared to results reported in the previous work. Additionally, all our experiments were done on the original protein features developed by Ding and Dubchak. Further work can be done using OP-AELM using deep machine learning convolutional networks and autoencoders as a semi supervised machine learning algorithm on other high dimensional protein sequences to reduce the computational power, more accuracy and higher recognition rate.

In sixth chapter, we used the pruned annular extreme learning machines for identifying the brain abnormalities from Magnetic Resonance (MRI) structural images. Fully automatic normal and diseased human brain
classification from magnetic resonance images (MRI) is of great importance for research and clinical studies. Out of several debilitating ageing related health conditions, white matter lesions (WMLs) are commonly detected in elders and in patients with multiple brain abnormalities like Alzheimer’s disease, Huntington’s disease and other neurological disorders. Therefore, segmentation and quantification of white matter lesions via texture analysis is very important in understanding the impact of aging and diagnosis of various brain abnormalities. Manual segmentation of WM lesions, which is still used in clinical practices, shows the limitation to differentiate brain abnormalities using human visual abilities. Such methods can produce a high risk of misinterpretation and can also contribute to variation in correct classification. The signal of homogeneity and heterogeneity of abnormal areas in Region of Interest (ROI) in white matter lesions of brain in T2-MRI images can be quantified by texture analysis algorithms. The ability to measure small differences in MRI images is essential and important to reduce the diagnosis errors of brain abnormalities. In this study, we present the study on magnetic resonance imaging (MRI) of various states of brain, by extracting the most significant features for classification into normal and abnormal brain images. We propose a novel method based on the wavelet transform to initially decompose the images, and then use various feature selection algorithms to extract the most significant features of brain from the MRI images. By using different classifiers to detect the abnormality of brain
images from a publicly available neuroimaging dataset, we found that a principled approach involving wavelet-based feature extraction, followed by selection of the most significant features using the proposed Optimised Pruned Extreme Learning Machines, and classification using learning-based classifiers results in a significant improvement in accuracy as compared with previously reported studies. The classification accuracy of more than 93% in case of deep machine learning and 94% in case of pruned extreme machine learning demonstrates the utility of the proposed method. The training and testing time for the complete study used by deep and extreme machine learning is much less as compared to SVM and other traditional classifiers reported in the literature. In this study, we have applied this method only to axial T2-weighted images at a particular depth inside the brain. Future work can be extended by using the same method can be employed for T1-weighted, proton density and other types of MR images. With the help of above approaches, one can develop software for a diagnostic system for the detection of brain disorders like Alzheimer’s, Huntington’s, Parkinson’s diseases etc.

In seventh chapter, a novel approach has been used for Email Personalization and User Profiling using RANSAC Multi Model Response Regression based Optimized Pruning Extreme Learning Machines and Gradient Boosting Trees on e-commerce dataset. Personalization is a key component for adapting a standardized product or service to an individual
customer’s needs. Application of personalization fits nicely into notions of Internet to provide a rich environment for well suited interaction and segmentation. Therefore, the requirement for predicting user needs in order to improve the usability and user retention of a Web site can be addressed by personalizing it. User profiling is the process of creating taste profiles to specific customer either explicitly or implicitly. Taste profiles of users created by using demographic information, their interests and their behaviour from web traffic data. Using user’s taste profiles, customization and content of emails marketing and website can be exploited with customer’s specific needs.

In this study, we present the proposed architectural design for email personalization using ourdeal database based on grad boost with optimized pruned extreme learning machines as base estimators. We also conducted a depth dive in data analysis to find each members behaviour and important attributes which plays a significant role in increasing clicks rates in personalized emails.

After data is extracted using step up process according to defined architectural design, we treat the data with grad boost classifier using optimized pruned extreme learning machine and linear regression as based estimator. Experimental results showed that as the iterations increased from 100 to 300, mean square error is much lower compare to grad boost with linear regression as base estimators. We can conclude that our proposed
method works much better in predicting customer’s behaviour on deals send in personalized emails. Higher accurate model helps us to find out better customer behaviour and design the customer plan according to accurate customer’s needs.

Currently this model is implemented in ourdeal, an Australian ecommerce database, with every morning set of deals selected using proposed method for personalized emails and defining user profiles. In future work, we will deploy this strategy of email personalization and user profiles using multi-attributes telecommunication databases at OPTUS. Further study will be extended by deploying email personalization and defining user profiles at OPTUS using multiple data sources as telecommunication data, demographic, social media to provide services at personalized level.
Appendices

Databases

Chapter IV Experimental database

The data sets were collected from the University of California at Irvine (UCI) Machine Learning Repository and their different attributes for the data sets for classification and regression.

Chapter V Experimental Database

In this study, we used datasets derived from the structural classification of proteins (SCOP) database (Berman et al., 2007). Details of these protein sets are described and presented in Table 5.1.

Chapter VI Experimental Database

The input dataset consists of axial, T2-weighted, 256 X 256 pixel MR brain images (Figure 6.1). These images were downloaded from the (Harvard Medical School website (http://med.harvard.edu/AANLIB/). Only those sections of the brain in which lateral ventricles are clearly seen are considered in our study. The number of MR brain images in the input dataset is 60 of which 6 are of normal brain and 54 are of abnormal brain. The abnormal brain image set consists of images of brain affected by Alzheimer’s and other diseases.
Chapter VII Experimental Database

Ourdeal database for this study is collected from customer’s response on e-commerce website ourdeal.com.au into Amazon redshift server using Google Analytics and responsys platform for collection of transactional data. Every transactional data of 1 million customers are tracked and recorded when they click or visit any page of ourdeal website. The database also have customer data record of clicking and opening mails send to customers based on proposed recommendation systems engines designed in chapter VII. In this experimental study, transactional customer’s responses data is used from Jan 2014-Dec 2014. Ourdeal database is currently private database having personal and transaction information of customers and cannot be used for any other experimental study.
REFERENCES

- Balasundaram, S., Gupta, D., & Kapil. 1-Norm extreme learning machine for regression and multiclass classification using Newton method. Neurocomputing(0). doi: http://dx.doi.org/10.1016/j.neucom.2013.03.051
- Balázs, G. Cascade-Correlation Neural Networks: A Survey.
novel neural network algorithm. Paper presented at the GCC Conference and Exhibition (GCC), 2011 IEEE.

  


  doi: http://dx.doi.org/10.1016/j.neucom.2012.02.042


  doi: http://dx.doi.org/10.1016/j.ins.2011.09.015
• doi: http://dx.doi.org/10.1016/j.suscom.2012.10.002
• Ebrahimzadeh, A., & Ranaee, V. (2010). Control chart pattern recognition using an optimized neural network and efficient features. ISA Transactions, 49(3), 387-393. doi: http://dx.doi.org/10.1016/j.isatra.2010.03.007
Neurocomputing, 74(16), 2502-2510. doi: http://dx.doi.org/10.1016/j.neucom.2010.11.032


- Figueiredo, E. M. N., & Ludermir, T. B. Investigating the use of alternative topologies on performance of the PSO-ELM. Neurocomputing(0). doi: http://dx.doi.org/10.1016/j.neucom.2013.05.047


- doi: http://dx.doi.org/10.1016/j.neucom.2008.11.004
  doi: 10.1007/s10706-008-9229-9
  doi: http://dx.doi.org/10.1016/j.neucom.2013.02.047
• Guo, L., Hao, J.-h., & Liu, M. An incremental extreme learning machine for online sequential learning problems. Neurocomputing(0).
  doi: http://dx.doi.org/10.1016/j.neucom.2013.03.055
  doi: 10.1162/089976602760128090
  doi: http://dx.doi.org/10.1016/j.neunet.2013.01.015
  doi: http://dx.doi.org/10.1016/j.neucom.2012.01.040

258
• He, Y.L., Geng, Z.Q., Xu, Y., & Zhu, Q.X. A hierarchical structure of extreme learning machine (HELM) for high-dimensional datasets with noise. Neurocomputing(0). doi: http://dx.doi.org/10.1016/j.neucom.2013.08.024


  doi: 10.1109/TSMCB.2002.804368

• Horata, P., Chiewchanwattana, S., & Sunat, K. (2013). Robust extreme learning machine. Neurocomputing, 102(0), 31-44.


  doi: http://dx.doi.org/10.1016/j.neucom.2007.07.025

  doi: 10.1007/s13042-011-0019-y

• doi: 10.1109/TSMCB.2008.2008724
• doi: http://dx.doi.org/10.1016/j.neucom.2013.07.021
• doi: http://dx.doi.org/10.1016/j.asoc.2009.12.025
• Khan, A., Yang, J., & Wu, W. Double parallel feedforward neural network based on extreme learning machine with L1/2 regularizer. Neurocomputing(0).
• doi: http://dx.doi.org/10.1016/j.neucom.2013.03.053
• doi: http://dx.doi.org/10.1016/j.eswa.2011.08.116


  doi: http://dx.doi.org/10.1016/j.eswa.2007.09.003


  doi: http://dx.doi.org/10.1016/j.neucom.2009.02.013

  doi: http://dx.doi.org/10.1016/j.neucom.2010.05.022

doi: http://dx.doi.org/10.1016/j.supflu.2012.10.013


doi: http://dx.doi.org/10.1016/j.neucom.2012.03.010


doi: http://dx.doi.org/10.1016/j.neucom.2010.09.019


doi: http://dx.doi.org/10.1016/j.neucom.2013.05.051
• Li, X., Mao, W., & Jiang, W. Fast sparse approximation of extreme learning machine. Neurocomputing(0). doi: http://dx.doi.org/10.1016/j.neucom.2013.01.064

• Lim, J. (2013). Partitioned online sequential extreme learning machine for large ordered system modeling. Neurocomputing, 102(0), 59-64. doi: http://dx.doi.org/10.1016/j.neucom.2011.12.049


  doi: 10.1007/BF02478259
  doi: http://dx.doi.org/10.1016/j.neucom.2010.12.042
  doi: 10.1109/TNN.2011.2162341
  doi: http://dx.doi.org/10.1016/j.patcog.2011.03.013


  doi: http://dx.doi.org/10.1016/j.neucom.2007.08.026


  doi: http://dx.doi.org/10.1016/j.neucom.2012.01.041

  doi: http://dx.doi.org/10.1016/j.cmpb.2013.03.005


  doi: http://dx.doi.org/10.1016/j.fss.2010.08.007
• Olatunji, S. O., Selamat, A., & Abdulraheem, A. (2014). A hybrid model through the fusion of type-2 fuzzy logic systems and extreme learning machines for modelling permeability prediction. Information Fusion, 16(0), 29-45. doi: http://dx.doi.org/10.1016/j.inffus.2012.06.001


• doi: http://dx.doi.org/10.1016/j.knosys.2010.11.001


• Sun, Y., Yuan, Y., & Wang, G. Extreme learning machine for classification over uncertain data. Neurocomputing(0).

• doi: http://dx.doi.org/10.1016/j.neucom.2013.08.011


• doi: http://dx.doi.org/10.1016/j.neucom.2009.03.016


  doi: http://dx.doi.org/10.1016/j.neucom.2010.12.035

  doi: http://dx.doi.org/10.1016/j.neucom.2013.02.053

  doi: http://dx.doi.org/10.1016/j.neucom.2010.11.030


  doi: http://dx.doi.org/10.1016/j.neucom.2012.02.003
  doi: http://dx.doi.org/10.1016/j.matdes.2008.06.006
  doi: http://dx.doi.org/10.1016/j.neucom.2010.12.041
  doi: http://dx.doi.org/10.1016/j.neucom.2012.08.010