NOVEL PROGRESSIVE LEARNING TECHNIQUE
FOR CLASSIFICATION PROBLEMS

RAJASEKAR VENKATESAN

School of Electrical and Electronics Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2017
Acknowledgements

This thesis is a major milestone in my journey of research and therefore, I am feeling very happy to thank all who have supported me for reaching it.

In the first place, I would like to express my heartfelt gratitude to Professor Er Meng Joo for his supervision, advice, guidance from the very early stage of this research and also for providing me with an excellent atmosphere for doing the research. His continuous encouragement and support in various ways inspired me and enriched my growth as a research student. It is his guidance and timely help that took me towards the completion of this thesis. It is indeed a great pleasure to work with a supervisor like Prof. Er Meng Joo who can always accommodate the students. Also, I’d like to extend my sincere gratitude to Professor Wang Dan Wei for his help, advice and support during the final stage of my candidature.

I would like to dedicate this thesis to my beloved parents and my lovely brother. Without my parents’ blessings and moral support, I don’t think I could have completed my PhD Studies. Thank you, mom and dad!!I love you and am forever indebted to you.

I would like to convey special thanks to my friends Narayanan Annamalai and Sathiya Narayanan Sekar for their continuous encouragement and support and useful technical discussions that helped me to shape my research in a better way.
I am very grateful to the School of Electrical and Electronics Engineering, Nanyang Technological University (NTU) for providing me with the research student scholarship as well as a conducive and comfortable environment to study. I would like to thank the Robotics II laboratory executive, Mr. Chia Chiu Kiat for his ever willingness to help.

My time at Singapore was made enjoyable in large part due to my roomies who became a part of my life. I am grateful to all my roomies (Satheesh, Rohit, Suresh, Vinoth, Prasanna and Jagadeesh) for their moral support in difficult times of my life.

Lastly, I would like to thank the Almighty for giving me wonderful parents, teachers and friends. They all made my life very pleasant and enjoyable.
# Contents

Acknowledgements

Contents

List of Figures

List of Tables

List of Abbreviations

Summary

1 **Introduction**

1.1 Classification

1.2 Learning Techniques

1.3 Thesis Overview

1.4 Organization of Thesis

2 **Background and Preliminaries**

2.1 Multi-label Classification

2.1.1 Batch Learning Methods
2.1.2 Online Methods

2.2 Extreme Learning Machines

2.3 Online Sequential Extreme Learning Machines

3 Batch Learning – Multi-label Classification

3.1 Proposed Algorithm

3.2 Experimentation

3.3 Results and Discussions

3.3.1 Consistency

3.3.2 Performance Metrics

3.3.3 Execution Speed

3.4 Summary

4 Online Learning – Multi-label Classification

4.1 Proposed OSML-ELM

4.2 Experimentation

4.3 Results and Discussions

4.3.1 Consistency

4.3.2 Performance Metrics

4.3.3 Comparison With the State-of-arts Techniques

4.3.4 Comparison of Execution Speed

4.3.5 Real-time Classification
4.3.6 Experimentation on Real-world Dataset 71

4.4 Summary 73

5 Online Learning – Universal Classifier 75

5.1 Proposed Approach 76

5.2 Experimentation 83

5.3 Results and Discussions 85

5.3.1 Consistency 86

5.3.2 Performance Comparison 88

5.3.3 Execution Speed 90

5.3.4 Experimentation on Real-world Dataset 92

5.4 Summary 93

6 Progressive Learning – Multi-class Classification 95

6.1 Progressive Learning Technique 96

6.2 Proposed Algorithm 98

6.3 Experimentation 107

6.4 Results and Discussions 110

6.4.1 Functionality 110

6.4.2 Consistency 115

6.4.3 Computational Reduction 117

6.4.4 Introduction of New Class at Different Time Instants 122
6.4.5 Multiple New Classes

6.4.6 Experimentation on Real-world Dataset

6.5 Summary

7 Progressive Learning – Multi-label Classification

7.1 Proposed Algorithm

7.2 Experimentation

7.3 Results and Discussions

7.3.1 Functionality

7.3.2 Consistency

7.3.3 Performance Metrics

7.3.4 Execution Speed

7.3.5 Experimentation on Real-world Dataset

7.4 Summary

8 Progressive Learning – Universal Classifier

8.1 Proposed Approach

8.2 Experimentation

8.3 Results and Discussions

8.3.1 Consistency

8.3.2 Performance

8.3.3 Execution Speed

125

134

136

138

139

148

150

150

153

155

156

157

158

159

161

171

173

173

174

176
### 8.4 Summary

### 9 Conclusions and Future Directions

#### 9.1 Conclusions

#### 9.2 Directions for Further Research

<table>
<thead>
<tr>
<th>Author’s Publications</th>
<th>182</th>
</tr>
</thead>
<tbody>
<tr>
<td>References</td>
<td>184</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Overview of Thesis 10

2.1 Overview of Multi-label Classification Techniques 17

2.2 Machine Learning Algorithms for Multi-label Problems 19

3.1 Hamming Loss Comparison 43

3.2 Accuracy Comparison 44

3.3 Precision Comparison 45

3.4 Recall Comparison 46

3.5 F1-measure Comparison 47

4.1 Hamming Loss Comparison 66

4.2 Accuracy Comparison 67

6.1 Flowchart of Algorithm 4 (Progressive Learning Technique for Multi-class Classification) 106

6.2 Testing Accuracy in Iris Dataset 112

6.3 Testing Accuracy in Waveform Dataset 114

6.4 Testing Accuracy in Balance-scale Dataset 115
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5</td>
<td></td>
<td>Comparison of Computational Reduction</td>
<td>119</td>
</tr>
<tr>
<td>6.6</td>
<td></td>
<td>Introduction of New Class in Training (a) Very Early, (b) In the Middle and (c) Towards the End</td>
<td>124</td>
</tr>
<tr>
<td>6.7</td>
<td></td>
<td>Sequential Learning of Two New Classes</td>
<td>127</td>
</tr>
<tr>
<td>6.8</td>
<td></td>
<td>Sequential Learning of Three New Classes</td>
<td>129</td>
</tr>
<tr>
<td>6.9</td>
<td></td>
<td>Individual and Overall Testing Accuracy – Sequential Introduction</td>
<td>130</td>
</tr>
<tr>
<td>6.10</td>
<td></td>
<td>Testing Accuracy for Simultaneous New Classes</td>
<td>132</td>
</tr>
<tr>
<td>6.11</td>
<td></td>
<td>Individual and Overall Testing Accuracy – Simultaneous New Classes</td>
<td>133</td>
</tr>
<tr>
<td>6.12</td>
<td></td>
<td>Evolution of Malware Families</td>
<td>135</td>
</tr>
<tr>
<td>7.1</td>
<td></td>
<td>Hamming Loss Plot for Scene (5+1) Dataset</td>
<td>151</td>
</tr>
<tr>
<td>7.2</td>
<td></td>
<td>Hamming Loss Plot for Medical (39+2+2+1+1) Dataset</td>
<td>152</td>
</tr>
<tr>
<td>7.3</td>
<td></td>
<td>Hamming Loss Plot for Corel5k (368+2+1+1+2) Dataset</td>
<td>153</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Dataset Specifications 37
3.2 Comparison Methods 38
3.3 Consistency Table – Cross Validation 40
3.4 Hamming Loss Comparison 43
3.5 Accuracy Comparison 44
3.6 Precision Comparison 45
3.7 Recall Comparison 46
3.8 F1-measure Comparison 47
3.9 Comparison of Training Time (in seconds) 48
3.10 Comparison of Testing Time (in seconds) 49
4.1 Dataset Specifications 61
4.2 Comparison Methods 63
4.3 Consistency Table – Cross Validation 64
4.4 Performance Metrics of OSML-ELM 65
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Training Time Comparison</td>
<td>68</td>
</tr>
<tr>
<td>4.6</td>
<td>Testing Time Comparison</td>
<td>68</td>
</tr>
<tr>
<td>4.7</td>
<td>Average Time per Epoch</td>
<td>70</td>
</tr>
<tr>
<td>4.8</td>
<td>Dataset Specifications for Aspect Detection</td>
<td>72</td>
</tr>
<tr>
<td>4.9</td>
<td>Experimental Results Comparison for Aspect Detection</td>
<td>73</td>
</tr>
<tr>
<td>5.1</td>
<td>Identification of Classification Type</td>
<td>80</td>
</tr>
<tr>
<td>5.2</td>
<td>Estimation of Number of Target Labels</td>
<td>81</td>
</tr>
<tr>
<td>5.3</td>
<td>Dataset Specifications</td>
<td>85</td>
</tr>
<tr>
<td>5.4</td>
<td>Consistency</td>
<td>87</td>
</tr>
<tr>
<td>5.5</td>
<td>Performance Comparison</td>
<td>88-89</td>
</tr>
<tr>
<td>5.6</td>
<td>Multi-label Performance Metrics</td>
<td>90</td>
</tr>
<tr>
<td>5.7</td>
<td>Execution Speed</td>
<td>91</td>
</tr>
<tr>
<td>5.8</td>
<td>Performance Comparison on AMD1 Dataset</td>
<td>93</td>
</tr>
<tr>
<td>5.9</td>
<td>Performance Comparison on AMD2 Dataset</td>
<td>93</td>
</tr>
<tr>
<td>6.1</td>
<td>Specifications of Multi-class Classification Datasets</td>
<td>109</td>
</tr>
<tr>
<td>6.2</td>
<td>Specifications of Iris Dataset</td>
<td>111</td>
</tr>
<tr>
<td>6.3</td>
<td>Specifications of Waveform Dataset</td>
<td>113</td>
</tr>
<tr>
<td>6.4</td>
<td>Specifications of Balance Scale Dataset</td>
<td>114</td>
</tr>
<tr>
<td>6.5</td>
<td>Consistency: Across Multiple Trials (10 Trials)</td>
<td>116</td>
</tr>
<tr>
<td>6.6</td>
<td>Consistency: 5-fold Cross Validation and 10-fold Cross Validation</td>
<td>117</td>
</tr>
</tbody>
</table>
6.7 Reduction in Number of Calculations by the Proposed Method
6.8 Comparison of Testing Accuracy (%)
6.9 Point of Introduction of New Class
6.10 Specifications of Character Dataset for 2 New Classes
6.11 Specifications of Character Dataset for 3 New Classes
6.12 Summary of Testing Accuracy for Sequential Introduction of Multiple New Classes
6.13 Specifications of Character Dataset for Simultaneous New Classes
6.14 Experimentation Results for AMD1 Dataset
6.15 Experimentation Results for AMD2 Dataset
7.1 Dataset Specifications
7.2 Consistency
7.3 Performance Metrics
7.4 Speed
7.5 Experimentation Results for Aspect Detection
8.1 Identification of Classification Type
8.2 Estimation of Number of Target Labels
8.3 Dataset Specifications
8.4 Consistency
8.5 Performance Measure
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.6</td>
<td>Performance Metrics for Multi-label Case</td>
<td>176</td>
</tr>
<tr>
<td>8.7</td>
<td>Execution Speed</td>
<td>177</td>
</tr>
</tbody>
</table>
List of Abbreviations

10-fcv 10-fold cross validation
5-fcv 5-fold cross validation
AA Algorithm Adaptation
BOMC Bayesian Online Multi-label Classification
BP Back Propagation
BR Binary Relevance
CC Classifier Chaining
CLR Calibrated Label Ranking
CT Classification Type
DT Decision Trees
ECC Ensemble of Classifier Chains
ELM Extreme Learning Machine
EN ENsemble
EOS-ELM Enhanced Online Sequential Extreme Learning Machine
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNN</td>
<td>Feedforward Neural Network</td>
</tr>
<tr>
<td>GP-ELM</td>
<td>Generalized Pruning Extreme Learning Machine</td>
</tr>
<tr>
<td>HOMER</td>
<td>Hierarchy Of Multi-label classifiERs</td>
</tr>
<tr>
<td>LC</td>
<td>Label Cardinality</td>
</tr>
<tr>
<td>LD</td>
<td>Label Density</td>
</tr>
<tr>
<td>ML-C4.5</td>
<td>Multi-Label C4.5</td>
</tr>
<tr>
<td>ML-kNN</td>
<td>Multi-Label k-Nearest Neighbours</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-Layer Perceptron</td>
</tr>
<tr>
<td>OS-ELM</td>
<td>Online Sequential Extreme Learning Machine</td>
</tr>
<tr>
<td>OSML-ELM</td>
<td>Online Sequential Multi-Label Extreme Learning Machine</td>
</tr>
<tr>
<td>PA</td>
<td>Passive Aggressive</td>
</tr>
<tr>
<td>PCT</td>
<td>Predictive Clustering Trees</td>
</tr>
<tr>
<td>PLT</td>
<td>Progressive Learning Technique</td>
</tr>
<tr>
<td>PT</td>
<td>Problem Transformation</td>
</tr>
<tr>
<td>QWML</td>
<td>QWeighted Multi-Label</td>
</tr>
<tr>
<td>RAKEL</td>
<td>RAndom k labEL sets</td>
</tr>
<tr>
<td>RB-ELM</td>
<td>Robust Bayesian Extreme Learning Machine</td>
</tr>
<tr>
<td>RDT</td>
<td>Random forest Decision Trees</td>
</tr>
<tr>
<td>RF</td>
<td>Random Forests</td>
</tr>
<tr>
<td>RFML-C4.5</td>
<td>Random Forest Multi-Label C4.5</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>RF-PCT</td>
<td>Random Forest – Predictive Clustering Trees</td>
</tr>
<tr>
<td>RLS</td>
<td>Recursive Least Square</td>
</tr>
<tr>
<td>ROS-ELM</td>
<td>Robust Online Sequential Extreme Learning Machine</td>
</tr>
<tr>
<td>SLFN</td>
<td>Single hidden Layer Feedforward Neural network</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machines</td>
</tr>
<tr>
<td>VOS-ELM</td>
<td>Voting based Online Sequential Extreme Learning Machine</td>
</tr>
</tbody>
</table>
Summary

The human brain is complex and is arguably the most sophisticated cognitive computer which can perform multifarious tasks rapidly and accurately. Developing a mental system to mimic the human brain and to enable machines to learn on their own without the need for extra programming each time is an active area of research and a feat yet to be achieved. Progressive learning is an effective learning model which is demonstrated by the human learning process. It is the process of learning continuously from direct experience. In machine learning, there are different categories of classification problems, namely (1) Single-label classification, which includes binary and multi-class classification and (2) Multi-label classification. Inspired by human learning, the research work has three key objectives. The first key objective of this research is the development of a label-independent generic classifier that is capable of performing binary, multi-class and multi-label classification. The second key objective of this research is developing an extreme learning machine based advanced learning technique capable of exhibiting progressive learning behaviour for classification problems. And finally, the third key objective is the integration of progressive learning technique to the label-independent classifier thereby resulting in a generic classifier capable of dynamic learning of new classes and also capable of addressing all the aforementioned types of classification problems.
The progressive learning technique based on the progressive learning paradigm exhibited by human learning process requires the classification technique to be independent of the number of class constraint and capable of learning several new classes on the go by retaining the knowledge of previous classes. It is realized by modifying the network structure by itself upon encountering a new class and updating the network parameters in such a way that it learns the new class by retaining the knowledge learnt thus far. The existing online sequential learning methods do not require retraining when a “new data sample” is received, but it fails when a “new class of data” which is unknown to the existing knowledge is encountered. Progressive learning technique overcomes this shortcoming by allowing the network to learn multiple new classes’ alien to existing knowledge, encountered at any point of time. In the sequence of batch and online learning paradigms, the next logical extension is progressive learning.

Machine learning classification can be categorized into single-label classification (binary and multi-class) and multi-label classification. Several machine learning classifiers have been developed and are available in the literature for each of the classification types. But the major limitation of all the classifiers in the literature is that, the classifiers are limited only to the particular type of classification problem for which it has been trained. There exists no classifier that is capable of universally addressing all the aforementioned types of classification problems. Inspired from human learning, a new label-independent/universal classifier which is capable of performing binary, multi-class and multi-label classification is developed.
The final outcome of the thesis is the development of human-learning-inspired progressively learning universally generic classifier. It is achieved by integrating the progressive learning feature onto the label-independent classifier. The newly developed classifier based on the extreme learning machine exploits its inherent high speed training and testing. Thus, the developed classifier can be used to address binary, multi-class and multi-label classification problems with dynamic class constraints accurately and efficiently.
Chapter 1

Introduction

Machine Learning is the training of a model from data that learns to generalize a decision against a performance measure. In other words, it’s the ability of a program to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, where its performance at tasks in $T$, as measured by $P$, improves with experience $E$. The machine learning tasks can be grouped into several categories such as classification, regression, clustering, dimensionality reduction, system modelling, control, pattern recognition, case reasoning, behavior identification etc. Among which, classification task is the problem of identifying which of the target categories a data sample belongs to.

1.1 Classification

Definition 1.1: Classification in machine learning is defined as “Given a set of training examples composed of pairs \( \{x_i, y_i\} \), find a function \( f(x) \) that maps each
attribute vector $x_i$ to its associated class $y_i$, $i = 1,2,\ldots,n$, where $n$ is the total number of training samples” [1].

This type of classification is the most common classification problem where the target classes are mutually exclusive. These classification problems are called as single-label classification. Single-label classification problems involve mapping each of the input vectors to its unique target class from a pool of target classes. However, there are several classification problems in which the target classes are not mutually exclusive and the input samples belong to more than one target class. These problems cannot be classified using the single-label classification thus resulting in the development of several multi-label classifiers to mitigate this limitation.

The single-label classification problem corresponds to training the network with data samples where each of the sample data is associated with a unique target class $\zeta_i$ from a pool of target class labels $L = \{\zeta_1, \zeta_2, \ldots, \zeta_M\}$, where $M > 1$. The single label classification problem can be further classified into binary classification and multi-class classification. Binary classification problems are the most trivial classification problems and have only two target classes ($M = 2$: $\zeta_1$ and $\zeta_2$). Medical diagnosis [2-4], biometric security [5], quality control, spam detection [6] and other similar applications are examples of binary classification. On the other hand, multi-class classification involves associating the data samples to any one of the target labels from a set of target labels ($M > 2$). In multi-class classification, each of the input samples corresponds to a unique class among a
pool of target class labels. Character recognition [7-9], biometric identification [10, 11], and other related applications are examples of multi-class classification. Several online machine learning classifiers for single-label classification are available in the literature [12, 13]. Evolving classifiers [14, 15] and fuzzy systems based classifiers [16-18] have also been developed for streaming data applications.

Unlike single-label classification, multi-label problems allow the input samples to be associated with more than one target labels. I.e. multi-label problems map the input data samples $x \in X$ to a set of labels $y \subseteq L$. The traditional binary and multi-class problems form a special case of multi-label problems. Thus, it can be stated that multi-label classifiers form the generalization of the classification problems. Due to its generality, the multi-label classification problems are more difficult and more complex when compared to single-label classification problems [19].

The need for multi-label classification is initially motivated by its need in text categorization. A single document can belong to more than one category of topics or labels, thus resulting in the need for a multi-label classifier. By the recent advancements in technology, the application areas of multi-label classifiers spread across various domains. The multi-label classification problems have gained much importance due to its rapidly increasing application areas. The application areas of multi-label classification include but are not limited to text categorization [20-24], bioinformatics [25, 26], medical diagnosis [27], image/scene and video categorization [28-30], genomics, map labeling [31], marketing, multimedia,
emotion, music categorization, etc. In recent years, the multi-label classification has drawn increasing research attention due to the realization of the omnipresence of multi-label prediction tasks in several areas [32, 33]. Due to the wide range of applications and increasing importance, several multi-label classification techniques have been developed and are available in the literature.

1.2 Learning Techniques

Soft computing refers to the computational techniques that use the inexact solutions to computationally hard tasks such as the solution of NP-complete problems. Soft computing techniques are tolerant to imprecision, uncertainty, partial truth and approximation. Machine learning, fuzzy logic, evolutionary computation and probabilistic reasoning are the four pillars of soft computing. Evolutionary computation includes several families of algorithms such as genetic algorithms, ant colony optimizations, particle swarm optimization etc. Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning allows computers to find insights in the data without being explicitly programmed on the characteristics and trends of the data. The essence of machine learning techniques is to automatically learn to make accurate predictions based on past observations. Machine learning classification is the problem of predicting the output label/class of an input sample based on the knowledge learnt from labelled training samples. Therefore, classification belongs to the category of supervised learning. Classification is used in a wide range of application domains such as text
categorization, fraud detection, optical character recognition, machine vision, natural language processing, market segmentation, bioinformatics etc. There are several machine learning algorithms available extensively in literature: neural networks, decision trees, support vector machines, naïve bayes, random forest etc. Inspired from biological neurons, neural networks are a subset of machine learning algorithms that are built around a model of interconnected layers of artificial neurons. There are two main categories of neural network based algorithms: feedforward and feedback neural networks. In a feedforward neural network, the interconnections are cascaded where the output of preceding layer is used as input of the following layer. It can be further categorized into single layer feedforward neural network and multi-layer feedforward neural network depending on the number of hidden layer neurons present in the network. In feedforward neural network the direction of information communication is single and forward: from preceding layer to following layer. A feedback neural network is a feedforward neural network with a feedback loop which feeds the outputs of a particular layer neurons back to the preceding layer(s). Hopfield neural network, bi-directional associative memory, recurrent neural networks are some of the examples of feedback neural networks.

The study on feedforward neural network (FNN) has gained prominence since the advent of back-propagation (BP) algorithm [34]. Several improved and optimized variants of the BP algorithm have then been developed and analyzed [35-40]. In the past two decades, single hidden layer feedforward neural networks (SLFNs) has gained significant importance due to its widespread applications in
recognition, classification and function approximation area [41-45]. Several
learning techniques have been proposed since then for effective training of the
SLFN [46]. Different classification techniques based on multi-layer perceptrons
(MLP), decision trees (DT), k-nearest neighbors (kNN), support vector machine
(SVM), extreme learning machine (ELM), naïve bayes classifier etc. have been
developed for each of the classification types and is available in the literature [47-
50]. The learning techniques can be grouped under two basic categories; Batch
learning and Online/Sequential learning [51].

In batch learning, all the training data are collected in prior, and the
parameters of the network are calculated by processing all the training data
concurrently. This poses a major limitation on the batch learning techniques as they
are unable to learn from streaming data. There are several batch learning
techniques available in the literature. One of the relatively new batch learning
scheme called Extreme Learning Machines (ELM) is proposed by Huang et al in
2004 [52]. The special nature of ELM is that the input weights and the hidden node
biases can be chosen at random [53]. A key feature of ELM is that it maintains the
universal approximation capability of SLFN. It has gained much attention to it and
several research works are further made in it due to its special nature of random
input weight initialization and its unique advantage of extreme learning speed [54].
The advantages of ELM over other traditional feedforward neural networks are
analyzed in the literature. Many new variants and developments are made to the
ELM and significant results are achieved in the approximation, classification and
regression areas [55, 56].
On the other hand, in online/sequential learning techniques the network parameters are updated iteratively with single-pass learning procedure [57-60]. Several books are available in the literature that comprehensively elaborates the data stream classification [61-64]. In many cases, online learning is preferred over batch learning as they can learn from data streams [65, 66] and do not require retraining whenever a new data sample is received [41, 59, 60]. Online-sequential learning method that combines ELM and Recursive Least Square (RLS) algorithm has been developed and is called Online-Sequential extreme learning machine (OS-ELM) [51]. Several variants of ELM and OS-ELM were developed and proposed in the literature [41, 42, 53, 55, 56, 67].

The issue with existing classification techniques is that, once they are trained to classify a specific number of classes, learning new classes is not possible. In order to learn the new class of data, it requires retraining all the classes anew again. But, as opposed to traditional machine learning algorithm’s training-testing cycle, human learning is a continuous process. The learning / training phase is never ending. Whenever human brain is stumbled upon with a new phenomenon, the learning resumes. The prime feature of human learning is that; the learning of new phenomenon does not affect the knowledge learnt. The new knowledge is learnt and is added along with existing knowledge. But the existing techniques in machine learning classification requires apriori information on the number of classes that will be present in the training dataset. The information on the number of classes is required to be either specified directly or is identified by analyzing the complete training data set. Based on this parameter, the network model will be
designed and only the parameters or the weights of the networks are updated depending on the input data through batch learning or online learning. This makes the existing techniques “static” with respect to the number of classes it can learn. While the existing techniques are suited for applications with a pre-known dataset, it might not be well suited for applications where the nature of training data is unknown and dynamic. For the real world and real time data, where the number of classes to be learnt is often unknown, the learning technique must be self-developing to meet the dynamic needs. The proposed progressive learning technique overcomes this shortcoming by allowing the network to learn multiple new classes’ alien to existing knowledge, encountered at any point of time. In the sequence of batch and online learning paradigms, the next logical extension is progressive learning.

1.3 Thesis Overview

The prime objectives of this research work are:

1. Development of a label-independent (single label or multi-label) classifier that is capable of performing binary, multi-class and multi-label classification.

2. Developing an extreme learning machine based advanced learning technique capable of exhibiting progressive learning behaviour for classification problems.

3. Integration of progressive learning technique to the label-independent (or universal) classifier thereby resulting in a generic classifier capable of
dynamic learning of new classes and also capable of addressing all the aforementioned types of classification problems.

In this line of objectives, the contribution of the thesis is the development, experimental analysis and evaluation of following neural network based techniques.

1. Batch learning classifier for multi-label classification
2. Online learning classifier for multi-label classification
3. Online learning label-independent classifier for binary, multi-class and multi-label classification
4. Progressive learning classifier for multi-class classification
5. Progressive learning classifier for multi-label classification
6. Progressive learning label-independent classifier for binary, multi-class and multi-label classification

The overview of the research works accomplished and the contribution of the thesis are summarized in Figure 1.1.
1.4 Organization of Thesis

In Chapter 2, a comprehensive summary of the theory of extreme learning machines and online sequential extreme learning machine are discussed. This chapter also reviews the various existing techniques available in the literature for each of the classification types.

In Chapter 3, a batch learning classifier for multi-label problems based on extreme learning machine is developed and discussed. This chapter also evaluates the developed method for various performance metrics. The proposed method is applied to 6 benchmark datasets of different domains and a wide range of label density and label cardinality. The results are compared with 9 state-of-the-arts multi-label classifiers. It can be seen from the results that the proposed method
surpasses all state-of-the-arts methods in terms of speed and remain one of the top techniques in terms of other performance metrics.

In Chapter 4, a high-speed online neural network classifier based on extreme learning machines for multi-label classification is developed and discussed. The developed technique serves as the base platform for the label-independent (or universal) classifier which is discussed in Chapter 5. The proposed method is experimented with six different datasets from different application domains such as multimedia, text, and biology. The hamming loss, accuracy, training time and testing time of the proposed technique is compared with nine different state-of-the-art methods introduced in chapter 3. Experimental studies show that the proposed technique outperforms the existing multi-label classifiers in terms of both performance and speed.

In Chapter 5, a single label-independent (or universal) classifier which is capable of classifying binary, multi-class and multi-label classification is developed. There are no such universal classifiers available in the literature thus far. The developed classifier is experimented with datasets from binary, multi-class and multi-label problems. The results obtained are compared with state-of-the-art techniques from each of the classification types.

In Chapter 6, the human learning inspired progressive learning technique (PLT) is introduced. The PLT is independent of the number of class constraints and it can learn several new classes on the go by retaining the knowledge of previous classes. In this chapter, the PLT is incorporated to multi-class classification.
consistency and the complexity of the PLT for multi-class classification are analyzed. Several standard datasets are used to evaluate the performance of the developed technique. A comparative study shows that the developed technique is superior over the existing techniques.

In Chapter 7, the progressive learning technique is incorporated with the online multi-label classification developed in chapter 4. As new labels are being introduced to the network, the network automatically restructures itself, updates the number of output neurons and the weights of interconnections between the neurons in such a way to facilitate learning of the newly introduced labels. From the experimental results, it can be seen that, as new labels are being introduced, the hamming loss of the network drops, which represents the new label being learned by the network. The performance of the proposed technique is evaluated using several multi-label datasets with different combinations of new labels being introduced at various time instances.

In Chapter 8, the final outcome of the thesis is achieved. The progressive learning technique is incorporated with the label-independent classifier and a new classifier is developed. The resulting new classifier can be used for any type of classification problem and any number of dynamic class constraints. The developed classifier is experimented with datasets from different classification types and the performance metrics are evaluated. The results are compared with the state-of-the-arts classifiers of the corresponding classification type.
Finally, the conclusions of the research findings are summarized in Chapter 9. It also discusses some of the promising future research directions and ideas from the current work.
Chapter 2

Background and Preliminaries

This chapter reviews the various state-of-the-arts classification techniques for multi-label classification available in the literature. The existing techniques are categorized and discussed. This chapter also provides a comprehensive overview of the ELM and OS-ELM methods. It summarizes the theory and the mathematical background of the ELM and OS-ELM technique.

2.1 Multi-label Classification

Multi-label classification has gained much importance in recent years due to its wide range of application domains. As opposed to single-label classification, each input sample is associated with a set of target labels in multi-label classification. The number of target labels corresponding to each input is not fixed and varies dynamically.
Definition 2.1 The definition for multi-label learning as given by [68] is, “Given a training set, \( S = (x_i, y_i), 1 \leq i \leq n \), consisting \( n \) training instances, \( (x_i \in X, y_i \in Y) \) drawn from an unknown distribution \( D \), the goal of the multi-label learning is to produce a multi-label classifier \( h : X \rightarrow Y \) that optimizes some specific evaluation function or loss function”

Let \( p_i \) be the probability that the input sample is assigned to \( i^{th} \) class from a pool of \( M \) target classes. For single label classification such as binary and multi-class classification conforms to the following equality condition.

\[
\sum p_i = 1
\]  

(1)

Multi-label problems does not hold to this equality as each sample may have more than one target class. Also, it can be seen that the binary classification problem, the multi-class problem and ordinal regression problems are specific instances of the multi-label problem with the number of labels corresponding to each data sample restricted to 1 [25].

The multi-label learning problem can be summarized as follows:

- There exists an input space that contains tuples (features or attributes) of size \( D \) of different data types such as Boolean, discrete or continuous. \( x_i \in X, x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \)

- A label space of tuple size \( M \) exists which is given as, \( L = \{\zeta_1, \zeta_2, \ldots, \zeta_M\} \)
- Each data sample is given as a pair of tuples (input space and label space respectively). \( \{(x_i, y_i) \mid x_i \in X, y_i \in Y, Y \subseteq L, 1 \leq i \leq N\} \) where \( N \) is the number of training samples.

- A training model that maps the input tuple to the output tuple with high speed, high accuracy and less complexity.

Several approaches for solving multi-label problem are available in the literature. Earlier categorization of the batch learning multi-label methods [69] classify the methods into two categories: Problem Transformation methods (PT) and Algorithm Adaptation methods (AA). This categorization is extended to include a third category of methods by Gjorgji Madjarov et al [70] called Ensemble methods (EN). Several review articles are available in literature that describes the various methods available for multi-label classification [1, 32, 68-70]. A brief summary of existing methods based on the mentioned review articles is discussed in this section. As adapted from [70], the overview of the multi-label methods available in the literature is given in Figure 2.1.
2.1.1 Batch Learning Methods

This section reviews the several batch learning techniques for multi-label classification available in the literature. As foreshadowed, the existing methods can be categorized into three categories; Problem transformation methods, Algorithm adaptation methods and Ensemble methods.

*Problem Transformation (PT) Methods*

PT methods are the methods that convert the multi-label problem into one or many single label problems and employ existing single label methods to solve the multi-label case. This type of method is also called as Algorithm independent approach because the transformation of the multi-label to single label of data is independent of the algorithm employed to decipher the problem. Some of the early PT methods used in literature are instance elimination, label elimination, label creation, label decomposition, etc. The recent PT methods can be classified into
three sub-categories: binary relevance methods, pairwise methods and label power-set methods. Binary relevance methods convert the M labelled, multi-label problem into M different binary problems. This follows the typical one-vs-all technique for classification. Binary relevance (BR) and classifier chaining (CC) are examples of this category. Pairwise methods utilize multiple classifiers that cover all possible combinations of label pairs. The end result of each of the classifier is then combined to provide the final result. Calibrated Label Ranking (CLR) employs voting based technique to combine the end result and QWML method utilizes Qweighted approach for result association. The label power-set methods combine the occurring label pairs into a new label and thereby converting the multi-label problem into a single label problem. HOMER is an example of such label power-set method.

Algorithm Adaptation (AA) Methods

AA methods are algorithm dependent methods in which the base algorithm itself is extended to adapt to the multi-label scenario. Based on the base algorithm on which the extension of multi-label has been made, the AA methods can be further classified. There are several such techniques available in the literature. Some of the key techniques developed are, predictive clustering trees (PCTs) are adapted from decision tree based classification. Multi-label k-nearest neighbors (ML-kNN) are extended from traditional k-nearest neighbors. Well known C4.5 algorithm is modified for multi-label C4.5 (ML-C4.5) to facilitate multi-label
problem solving. Other techniques like neural networks, SVM, Boosting are also extended to have multi-label variants.

*Ensemble (EN) Methods*

EN methods are developed over the AA and PT methods. EN methods employ an ensemble of learning methods available in AA or PT. Ensemble Classifier Chains (ECC) are an ensemble method that uses CC as base technique. Random forests (RF) are used to ensemble with algorithms such as PCTs, decision trees (DT) and ML-C4.5 thus resulting in RF-PCT, RDT and RFML-C4.5 ensemble multi-label techniques respectively. Random k label sets (RAkEL) method extracts m random subsets of labels with size k and uses label power-set for classifying each of the label sets.

Based on the machine learning algorithm used, the multi-label techniques can be categorized as shown in Figure 2.2. The figure is adapted from [70].

![Figure 2.2. Machine Learning Algorithms for Multi-label Problems](image-url)
Transforming the multi-label problem into multiple single label problem results in training more number of classifiers and thereby increasing the training time and testing time. The increase of execution time is directly proportional to the multi-labelness of the data set which is quantified by label density and label cardinality. Thus, there is a strong need for a method that can perform multi-label classification in a fast manner. Extreme learning machine (ELM) is a high speed learning network, which has been extensively studied in literature for various applications. Though there exists a large number of techniques for multi-label learning, it can be seen from the review that there exists no single multi-label learning algorithm which is based on extreme learning machine.

2.1.2 Online Methods

Due to the complicated nature of multi-label problems, very few works are available in online learning for multi-label classification. Some of the significant works are briefly reviewed. There are limited number of techniques available in the literature on multi-label classification for data streams [71]. A simpler approach is to use batch learning classifiers that trains on new batches of data streams by replacing the classifiers of previous batches. This type of learning is called batch-incremental learning. The first work on multi-label classifier for data streams is based on ensemble of classifiers which are trained on successive data chunks [72]. The paper by Read et. al [73] proposes multi-label stream classification by extending the heoffding tree [74] by using batch multi-label classifier in each node. Spyromitros-Xioufis [75] proposes binary relevance and kNN based multi-label
classifier for data streams. Microsoft [76] developed an Active Learning framework for multi-label classification as the result of the increase in demand for the need of multi-label classification in real world multi-media datasets. A Passive-Aggressive method is proposed by Crammer et. al [77] for multi-label classification. A Bayesian Online Multi-label Classification (BOMC) method is developed by Zhang et. al [78] for online multi-label classification. The Passive Aggressive and the Bayesian Online Multi-label Classification techniques are application specific and are implemented only for text categorization datasets. From the lack of mentioning of any online multi-label classification methods in any of the multi-label review articles thus far, it is evident that there are no generic online multi-label classification techniques that can be applied to a wide range of application domains. The PA and the BOMC techniques are implemented only for text categorization datasets and the Active Learning framework from Microsoft is application specific to multimedia datasets.

2.2 Extreme learning machines

ELM is a single-hidden layer feedforward neural network based learning technique [42, 79]. A key feature of ELM is that it maintains the universal approximation capability of single hidden-layer feedforward neural network. It has gained much attention due to its special nature of random input weight initialization and its unique advantage of extreme learning speed [46, 54]. In ELM, the initial weights and the hidden layer bias can be selected at random [80-84], and the network can be trained for the output weights to perform the classification [81-
84]. This results in a fast learning speed and generalization of performance. The theory and mathematical background of ELM has been extensively discussed in the literature [42, 46, 79]. There are other similar neural network based techniques [85, 86] which did not gain popularity and are largely forgotten.

A condensed overview of the batch learning ELM technique as proposed by Huang et. al. [52] is given below.

Consider there are N training samples represented as \{\( (x_j, t_j) \) \} where \( j \) varies from 1 to N, \( x_j \) denotes the input data vector: \( x_j = [x_{j1}, x_{j2}, \ldots x_{jn}]^T \in \mathbb{R}^n \) and \( t_j = [t_{j1}, t_{j2}, \ldots t_{jm}]^T \in \mathbb{R}^m \) denotes the target class labels. Let there be \( N' \) number of hidden layer neurons in the network, the output of the standard SLFN can be given as

\[
\mathbf{o}_j = \mathbf{f}(x_j) = \sum_{i=1}^{N'} \mathbf{\beta}_i \mathbf{g}_i(x_j) = \mathbf{h}(x_j) \mathbf{\beta}
\]  

(2)

For additive type activation function \( g(x) \), equation (2) can be written as,

\[
\sum_{i=1}^{N'} \mathbf{\beta}_i \mathbf{g}_i(x_j) = \sum_{i=1}^{N'} \mathbf{\beta}_i g(w_i \cdot x_j + b_i) = \mathbf{o}_j
\]  

(3)

where, \( j = 1, 2, \ldots N, \mathbf{w}_i = [w_{i1}, w_{i2}, \ldots w_{in}]^T \) denotes the weight vector from input nodes to \( i \)th hidden node, \( \mathbf{\beta}_i = [\beta_{i1}, \beta_{i2}, \ldots \beta_{in}]^T \) denotes the weight vector connecting \( i \)th hidden node to the output nodes and \( b_i \) is the hidden layer bias value.

For the standard SLFN mentioned in the equation above to perform as a classifier, the output of the network should be equal to the corresponding target
class of the input data given to the classifier. Therefore, the network is to minimize the training error as well as the norm of the output weights.

\[
\text{Minimize: } \| H\beta - T \|^2 \text{ and } \| \beta \|
\]

where, \( H \) is called the hidden layer output matrix of the neural network where each column of \( H \) gives corresponding output of the hidden layers for a given input \( x_i \). \( H, \beta \) and \( T \) are given as,

\[
H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix}_{N \times N' \prime} = \begin{bmatrix} h_1(x_1) & \cdots & h_{N'}(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_N) & \cdots & h_{N'}(x_N) \end{bmatrix}_{N \times N' \prime}
\]

(4)

\[
\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{N'}^T \end{bmatrix}_{N' \times m}
\]

(5)

\[
T = \begin{bmatrix} t_1^T \\ \vdots \\ t_{N'}^T \end{bmatrix}_{N \times m}
\]

(6)

For an additive type of activation function, the hidden layer output matrix \( H \) and the output of the network \( t \) can be given as,

\[
H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_{N'} \cdot x_1 + b_{N'}) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_{N'} \cdot x_N + b_{N'}) \end{bmatrix}_{N \times N' \prime}
\]

(7)

\[
\sum_{i=1}^{N'} \beta_i g(w_i \cdot x + b_i) = t_j
\]

(8)

where \( j = 1, 2, \ldots N \), and \( t_j \) denotes the target class corresponding to the input data vector \( x_j \). This equation can be written in compact form as
The mathematical framework and the training process are extensively described in the literature [42]. The key results are restated.

Lemma 1: [42] “Given a standard SLFN with $N$ hidden nodes and activation function $g: \mathbb{R} \rightarrow \mathbb{R}$ which is infinitely differentiable in any interval, for $N$ arbitrary distinct samples $(x_i, t_i)$, where $x_i \in \mathbb{R}^n$ and $t_i \in \mathbb{R}^m$, for any $w_i$ and $b_i$ randomly chosen from any intervals of $\mathbb{R}^n$ and $\mathbb{R}$, respectively, according to any continuous probability distribution, then with probability one, the hidden layer output matrix $H$ of the SLFN is invertible and \( \|H\beta - T\| = 0 \).”

Lemma 2: [42] “Given any small positive value $\epsilon > 0$ and activation function $g: \mathbb{R} \rightarrow \mathbb{R}$ which is infinitely differentiable in any interval, there exists $N' \leq N$ such that for $N$ arbitrary distinct samples $(x_i, t_i)$, where $x_i \in \mathbb{R}^n$ and $t_i \in \mathbb{R}^m$, for any $w_i$ and $b_i$ randomly chosen from any intervals of $\mathbb{R}^n$ and $\mathbb{R}$, respectively, according to any continuous probability distribution, then with probability one, \( \|H_{NN'} \beta_{N'Xm} - T_{NXm}\| < \epsilon \).”

Lemma 3: Given any bounded nonconstant piecewise continuous function $g: \mathbb{R} \rightarrow \mathbb{R}$ for additive nodes or any integrable piecewise continuous function $g: \mathbb{R} \rightarrow \mathbb{R}$ and \( \int_{\mathbb{R}} g(x)dx \neq 0 \) for RBF nodes, for any continuous target function $f$ and any randomly generated function sequence $\{g_n\}$, \( \lim_{n \to \infty} \|f - f_n\| = 0 \) holds with probability one if

\[
\beta_n = \frac{\langle e_{n-1}, g_n \rangle}{\|g_n\|^2}
\]
where, \( e_n = f - f_n \). Lemma 3 shows that, in the aspect of function approximation, the hidden nodes of the ELM are not required to be tuned. Furthermore, the universal approximation capability of ELM is the straightforward corollary of Lemma 3.

Thus it can be seen that for an ELM, the input weights \( w_i \), and the hidden layer neuron bias \( b_i \) can be randomly assigned. Training of the ELM involves estimating the output weights \( \beta \) such that the relation \( H\beta = T \) is true.

The output weight \( \beta \) for the ELM can be estimated using the Moore-Penrose generalized inverse as \( \beta = H^*T \), where \( H^* \) is the Moore-Penrose inverse of the hidden layer output matrix \( H \).

The overall batch learning algorithm of the ELM for training set of form \( \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, i = 1, \ldots, N\} \) with \( N' \) hidden layer neurons can be summarized as,

**STEP 1:** Random assignment of input weights \( w_i \) and hidden layer bias \( b_i \), \( i = 1, \ldots, N' \).

**STEP 2:** Computation of the hidden layer output matrix \( H \).

**STEP 3:** Estimation of output weights using \( \beta = H^*T \) where \( H^* \) is the Moore-Penrose inverse of \( H \) and \( T = [t_1, \ldots, t_N]^T \).
2.3 Online Sequential – Extreme Learning Machine

Based on the batch learning method of the ELM, sequential modification is performed and Online Sequential-ELM (OS-ELM) is proposed in literature [41]. OS-ELM operates on online data.

In the batch learning method ELM the output weight $\beta$ is estimated using the formula $\beta = H^+T$, where $H^+$ is the Moore-Penrose inverse of the hidden layer output matrix $H$. The $H^+$ can be written as,

$$H^+ = (H^TH)^{-1}H^T$$

(2)

As stated in [41], this solution gives the least square solution to $H\beta = T$.

The OS-ELM uses RLS algorithm to update the output weight matrix sequentially as the data arrives online. It has been well studied in the literature and the summary is given below.

Let $N_0$ be the number of samples in the initial block of data that is provided to the network.

Calculate $M_0 = (H_0^TH_0)^{-1}$ and $\beta_0 = M_0H_0^TT_0$.

For each of the subsequent sequentially arriving data, the output weights can be updated as

$$M_{k+1} = M_k - \frac{M_kh_{k+1}h_{k+1}^TM_k}{1 + h_{k+1}^TM_kh_{k+1}}$$

(3)
\[ \beta_{k+1} = \beta_k + M_{k+1}h_{k+1}(t_{k+1} - h_{k+1}'\beta_k) \]  

(4)

where \( k = 0,1,2,\ldots N-N_0-1 \).

The steps in the Online-Sequential ELM based on the RLS algorithm are summarized below.

**INITIALIZATION PHASE**

**STEP 1:** The input weights and the hidden layer bias are assigned in random.

**STEP 2:** For the initial block of \( N_0 \) samples of data, the hidden layer output matrix \( H_0 \) is calculated.

\[ H_0 = [h_1,\ldots,h_{N_0}]^T, \text{ where} \]

\[ h_i = [g(w_1x_i+b_1),\ldots,g(w_{N_0}x_i+b_{N_0})]^T, \quad i = 1,2,\ldots N_0. \]  

(5)

**STEP 3:** From the value of \( H_0 \), the initial values of \( M_0 \) and \( \beta_0 \) are estimated as

\[ M_0 = (H_0^TH_0)^{-1} \]  

(6)

\[ \beta_0 = M_0H_0^TT_0 \]  

(7)

**SEQUENTIAL LEARNING PHASE**

**STEP 4:** For each of the subsequent sequentially arriving data, the hidden layer output vector \( h_{k+1} \) is calculated.

**STEP 5:** The output weight is updated based on the RLS algorithm as,

\[ M_{k+1} = M_k - M_kh_{k+1}'(I + h_{k+1}M_kh_{k+1}^T)^{-1}h_{k+1}M_k \]  

(8)
\[ \beta_{k+1} = \beta_k + M_{k+1} h^T_{k+1}(t_{k+1} - h_{k+1} \beta_k) \] (9)

The theory and the formulation behind the operation of the OS-ELM and ELM have been discussed in detail in several literatures [41, 51]. The standard variants of activation function used in ELM [87] and other special mapping functions and their variants are discussed in detail in the literature [88-91]. The other variants of ELM includes ELM Kernel [92], ELM for imbalanced data [93], ELM for noisy data [94], Incremental ELM [95], ELM ensemble [96-98] and many other variants are summarized in [87].
Chapter 3

Batch Learning – Multi-label Classification

In this chapter, a high speed neural network classifier based on extreme learning machines for multi-label classification problem is developed and discussed. Multi-label classification is a superset of traditional binary and multi-class classification problems. The proposed work extends the extreme learning machine technique to adapt to the multi-label problems. As opposed to the single-label problem, both the number of labels the sample belongs to, and each of those target labels are to be identified for multi-label classification resulting in increased complexity. The proposed high speed multi-label classifier is applied to six benchmark datasets comprising of different application areas such as multimedia, text and biology. The training time and testing time of the classifier are compared with those of the state-of-the-arts methods. Experimental studies show that for all
the six datasets, our proposed technique performs well across all the datasets used for evaluation and have the fastest execution speed when compared to the existing multi-label classification methods.

By the recent advancements in technology, the application areas of multi-label classifiers spread across various domains. As discussed in Chapter 2, several classifiers are developed to address the multi-label problems and are available in the literature. The traditional binary and multi-class problems form a special case of multi-label problems. Thus, it can be stated that multi-label classifiers form the generalization of the classification problems. Also multi-label problems are more difficult to train compared to single label problems due to its generality. Due to the generality and the increased complexity in learning multi-label problems, in many of the existing methods, the time taken to train the classifier is generally high. As discussed in chapter 2, many of the existing multi-label classifiers belong to the problem transformation type where the multi-label problem is transformed into multiple single-label (binary or multi-class) problem and uses existing single-label techniques to address the multi-label problem. In this chapter, we propose a high speed multi-label classifier based on extreme learning machines (ELM) which performs uniformly well on datasets with wide range of label density and label cardinality and outperforms all the existing multi-label classifiers with respect to training time and testing time. The proposed method falls under the category of algorithm adaptation technique where in contrast to problem transformation type, the base algorithm itself is extended to adapt for multi-label classification problems.
3.1 Proposed Algorithm

The proposed algorithm aims at extending the extreme learning machine for multi-label classification. Therefore, it falls under the category of algorithm adaptation method. The various steps involved in the proposed multi-label ELM classifier are:

- Initialization of Parameters
- Processing of Inputs
- ELM Training
- ELM Testing
- Post-processing and Multi-label identification

A. Initialization of Parameters

Fundamental parameters such as the number of hidden layer neurons and the activation function are initialized.

B. Processing of Inputs

In traditional single label problems, the target class will be a single label associated with the input sample. But in multi-label each input sample can be associated with more than one class labels. Hence, each of the input samples will have the associated output label as a m-tuple with 0 or 1 representing the belongingness to each of the labels in the label space L. This is a key difference between the inputs available for single label and multi-label problems. As opposed to single label classification with a single target label, multi-label problems have a
target label set which is a subset of label space $L$. The label set denoting the belongingness for each of the labels is converted from unipolar representation to bipolar representation.

\section*{C. ELM Training}

The processed input is then provided to the basic batch learning ELM. Let $H$ be the hidden layer output matrix, $\beta$ be the output weights and $Y$ be the target label, the ELM can be represented in a compact form as $H\beta = Y$ where $Y \subseteq L$, $L = \{\zeta_1, \zeta_2, \ldots, \zeta_M\}$. In the training phase, the input weights and the hidden layer bias are randomly assigned and the output weights $\beta$ are estimated as $\beta = H^+Y$ where $H^+ = (H^TH)^{-1}H^T$ gives the Moore-Penrose generalized inverse of the hidden layer output matrix.

\section*{D. ELM Testing}

In the testing phase, the test data sample is evaluated using the values of $\beta$ obtained during the training phase. The input data which can be a combination of Boolean, discrete and continuous data type is given to the network. The network then predicts the target output using the equation $Y = H\beta$. The predicted output $Y$ obtained is set of real numbers of dimension equal to the number of labels.

\section*{E. Post-processing and Multi-label identification}

In traditional single label problems, the target label can be easily recognized from the raw predicted output. The index of the maximum value in the predicted
output gives the label to which the input sample belongs to. But this cannot be used in case of multi-label problems. The key challenge is that the input sample may belong to one or more than one of the target labels. Number of labels that the sample corresponds to is completely unknown. Hence a thresholding based label association is proposed. The L dimensioned raw predicted output is compared with a threshold value. The index values of the Y which are greater than the threshold fixed represents the belongingness of the input sample to the corresponding class. Setting the threshold value is of critical importance. Too high threshold will result in not assigning the labels to which the samples rightfully belong to and too low threshold value will result in wrongly assigning more labels to the input sample. Threshold setting has to be made in such a way that it maximizes the difference between the values of the label the data belongs to and label it does not. As a trivial case the threshold can be set as zero. In which case, the raw predicted output values will be passed as arguments to bipolar step function. The set of columns of the resulting matrix with values 1 gives the multi-label belongingness of the corresponding input. The summary of the proposed algorithm is given below.

**Algorithm 1: Proposed Multi-label ELM algorithm**

1. Initialization: The fundamental parameters of the network are initialized.
2. Pre-processing: The raw input data is processed for classification. Each input sample \( x(i) \) is associated with an m-tuple target labels \( (T(i)) \) with values representing the belongingness of the sample to the corresponding label.
3. For each input \( (x_i) \), the hidden layer output \( h_i \) is calculated
\[ h_i = [g(w_1 x_i + b_1), \ldots, g(w_N x_i + b_N)]^T \]

4. Hidden layer output matrix (training phase): \( H_{\text{train}} = [h_1, \ldots, h_N]^T \)

5. Output weight vector: \( \beta = H_{\text{train}}^T \) where \( H_{\text{train}}^T = (H_{\text{train}} H_{\text{train}}^T)^{-1} H_{\text{train}}^T \)

6. Prediction on training data: \( Y_{\text{train}} = H_{\text{train}} \beta \)

7. Threshold identification: From set of target labels \( T \) and the distribution of raw output values \( Y_{\text{train}} \), a threshold value \( (th) \) is identified.

8. Hidden layer output matrix (testing phase): \( H_{\text{test}} = [h_1, \ldots, h_N]^T \)

9. Prediction on testing data: \( Y_{\text{test}} = H_{\text{test}} \beta \)

10. Multi-label Classification: In multi-label classification, it is essential to identify both the number of labels and the subset of output labels to which each input belongs to.

11. Identification of number of labels: Number of labels corresponding to each input \( x(i) \) is identified by using the raw output values \( Y_{\text{test}}(i) \) and \( th \).

Number of labels = \( \sum \text{HS}(Y_{\text{test}}(i) - th) \)

where, \( \text{HS}(x) \) is the standard heaviside step function.

12. Identification of each of the output labels: The set of labels that corresponds to the indices value for which \( Y_{\text{test}}(i) > th \) is \textit{true} gives the output labels corresponding to the input \( x(i) \)

---

### 3.2 Experimentation

This section describes the different multi-label dataset metrics and gives the experimental design used to evaluate the proposed method.
Multi-label datasets have a unique property called the degree of multi-labelness. The number of labels, the number of samples having multiple labels, the average number of labels corresponding to a particular sample varies among different datasets. Not all datasets are equally multi-label. Two dataset metrics are available in the literature to quantitatively measure the multi-labelness of a dataset. They are Label Cardinality (LC) and Label Density (LD).

Consider there are \( N \) training samples and the dataset is of the form \( \{(x_i, y_i)\} \) where \( x_i \) in the input data and \( y_i \) is the target label set. The target label set is a subset of labels from the label space with \( M \) elements given as \( Y \subseteq L, \ L = \{\zeta_1, \zeta_2, \ldots, \zeta_M\} \).

**Definition 3.1** Label Cardinality of the dataset is the average number of labels of the examples in the dataset [69].

\[
\text{Label - Cardinality} = \frac{1}{N} \sum_{i=1}^{N} |Y_i| \tag{10}
\]

Label Cardinality is independent of the number of labels present in the dataset and signifies the average number of labels present in the dataset.

**Definition 3.2** Label Density of the dataset is the average number of labels of the examples in the dataset divided by \( |L| \) [69].

\[
\text{Label - Density} = \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i|}{|L|} \tag{11}
\]
Label density takes into consideration the number of labels present in the dataset. The properties of two datasets have same label cardinality, but different label density can vary significantly and may result in different behavior of the training algorithm [19]. The influence of label density and label cardinality on multi-label learning is analyzed by Bernardini et al in 2013 [99].

The proposed method is experimented with six benchmark datasets comprising of different application areas such as multimedia, text and biology. The performance of the proposed method is compared with that of nine existing methods and the results are discussed.

The datasets are chosen in such a way that they exhibit diverse nature of characteristics and the wide range of label density and label cardinality. The number of labels ranges from 6 labels to 374 labels and the number of features or attributes in the dataset ranges from 72 to 1449. The dataset metrics such as label cardinality varies from as low as 1.07 to as high as 4.24. Label cardinality of 1.07 represents that each of the input samples corresponds to 1.07 labels by average. Label cardinality of 4.24 signifies that each sample on an average corresponds to 4.24 labels. From the definition it can be seen that the label density takes into account the number of labels present in the dataset and is inversely proportional to it. Thus, lower the label density value indicates that only fewer samples correspond to a particular label. Hence the training method should be fast enough to learn the target label set within the limited samples. The datasets are obtained from KEEL.
multi-label dataset repository and the specifications of the dataset are given in Table 3.1.

Table 3.1. Dataset Specifications

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Domain</th>
<th>No. of Features</th>
<th>No. of Labels</th>
<th>LC</th>
<th>LD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>Multimedia</td>
<td>72</td>
<td>6</td>
<td>1.87</td>
<td>0.312</td>
</tr>
<tr>
<td>Yeast</td>
<td>Biology</td>
<td>103</td>
<td>14</td>
<td>4.24</td>
<td>0.303</td>
</tr>
<tr>
<td>Scene</td>
<td>Multimedia</td>
<td>294</td>
<td>6</td>
<td>1.07</td>
<td>0.178</td>
</tr>
<tr>
<td>Corel5k</td>
<td>Multimedia</td>
<td>499</td>
<td>374</td>
<td>3.52</td>
<td>0.009</td>
</tr>
<tr>
<td>Enron</td>
<td>Text</td>
<td>1001</td>
<td>53</td>
<td>3.38</td>
<td>0.064</td>
</tr>
<tr>
<td>Medical</td>
<td>Text</td>
<td>1449</td>
<td>45</td>
<td>1.25</td>
<td>0.027</td>
</tr>
</tbody>
</table>

The proposed method is experimented with the six benchmark datasets mentioned in Table 3.1. The training and testing time of the proposed method is compared with nine different multi-label techniques available in the literature. The nine techniques are chosen such that they are from PT, AA and EN methods. Also the chosen techniques belong to different learning paradigms such as SVM, decision trees and nearest neighbors. The details of the state of the art multi-label techniques used for result comparison are given in Table 3.2.
Table 3.2. Comparison Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Method</th>
<th>Machine Learning Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier Chain (CC)</td>
<td>PT</td>
<td>SVM</td>
</tr>
<tr>
<td>QWeighted approach for Multi-label Learning (QWML)</td>
<td>PT</td>
<td>SVM</td>
</tr>
<tr>
<td>Hierarchy Of Multi-label Classifiers (HOMER)</td>
<td>PT</td>
<td>SVM</td>
</tr>
<tr>
<td>Multi-Label C4.5 (ML-C4.5)</td>
<td>AA</td>
<td>Decision Trees</td>
</tr>
<tr>
<td>Predictive Clustering Trees (PCT)</td>
<td>AA</td>
<td>Decision Trees</td>
</tr>
<tr>
<td>Multi-Label k-Nearest Neighbors (ML-kNN)</td>
<td>AA</td>
<td>Nearest Neighbors</td>
</tr>
<tr>
<td>Ensemble of Classifier Chains (ECC)</td>
<td>EN</td>
<td>SVM</td>
</tr>
<tr>
<td>Random Forest Predictive Clustering Trees (RF-PCT)</td>
<td>EN</td>
<td>Decision Trees</td>
</tr>
<tr>
<td>Random Forest of ML-C4.5 (RFML-C4.5)</td>
<td>EN</td>
<td>Decision Trees</td>
</tr>
</tbody>
</table>
3.3 Results and Discussions

This section discusses the results obtained by the proposed method and compares it with the existing methods. The results obtained from the proposed method are evaluated for consistency, performance and speed.

3.3.1 Consistency

Consistency is a key feature that is essential for any new technique proposed. The proposed algorithm should provide consistent results with minimal variance. Being an ELM based algorithm, since the initial weights are assigned in random, it is critical to evaluate the consistency of the proposed technique. The unique feature of multi-label classification is the possibility of partial correctness of the classifier, i.e. one or more of the multiple labels to which the sample instance belongs and/or the number of labels the sample instance belongs can be identified partially correctly. Therefore, calculating the error rate for multi-label problems is not same as that of traditional binary or multi-class problems. In order to quantitatively measure the correctness of the classifier, the hamming loss performance metric is used. To evaluate the consistency of the proposed method, a 5-fold and a 10-fold cross validation of hamming loss metric is evaluated for each of the six datasets and is tabulated.
Table 3.3. Consistency Table – Cross Validation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Hamming Loss - 5-fcv</th>
<th>Hamming Loss - 10-fcv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>0.2492(±0.0058)</td>
<td>0.2509(±0.0050)</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.1906(±0.0025)</td>
<td>0.1911(±0.0031)</td>
</tr>
<tr>
<td>Scene</td>
<td>0.0854(±0.0029)</td>
<td>0.0851(±0.0033)</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.0086(±0.0005)</td>
<td>0.0090(±0.0006)</td>
</tr>
<tr>
<td>Enron</td>
<td>0.0474(±0.0022)</td>
<td>0.0472(±0.0015)</td>
</tr>
<tr>
<td>Medical</td>
<td>0.0108(±0.0008)</td>
<td>0.0109(±0.0009)</td>
</tr>
</tbody>
</table>

From the Table 3.3, it can be seen that the proposed technique is consistent in its performance over repeated executions and cross validations thus demonstrating the consistency of the technique.

3.3.2 Performance Metrics

As foreshadowed, the unique feature of multi-label classification is the possibility of partial correctness of the classifier. Therefore, a set of quantitative performance evaluation metrics is used to validate the performance of the multi-
label classifier. The performance metrics are hamming loss, accuracy, precision, recall and F1-measure.

**Hamming Loss.** The hamming loss is a measure of misclassification rate of the learning technique. The lower the hamming loss, the better is the classification accuracy. The correctness of the classification of the learning technique can be analyzed by comparing the hamming loss metric. Hamming loss gives the percentage of wrong labels to the total number of labels. It represents the number of times the sample-label pair is misclassified. The hamming loss for an ideal classifier is zero. The hamming loss is calculated using the following expression,

\[
\text{Hamming Loss} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{m} |MLC(x_i) \Delta Y_i|
\]  

where, \(MLC(x_i)\) denotes the predicted output of the multi-label classifier, and \(Y_i\) gives the target result to be achieved.

**Accuracy.** Accuracy of a classifier is defined as the ratio of the total number of correctly predicted labels to the total number of labels of that sample instance. The accuracy measure can be evaluated using the following expression,

\[
\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \frac{|MLC(x_i) \cap Y_i|}{|MLC(x_i) \cup Y_i|}
\]

**Precision.** Precision is the proportion of the predicted correct labels to the total number of actual labels averaged over all instances. In other words, it is the ratio of
true positives to the sum of true positives and false positives averaged over all instances. Precision can be computed as follows,

\[
Precision = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{|MLC(x_i) \cap Y_i|}{|MLC(x_i)|} \right)
\]

(14)

Recall. Recall is the proportion of the predicted correct labels to the total number of predicted labels averaged over all instances. In other words, it is the ratio of true positives to the sum of true positives and false negatives averaged over all instances. The expression for recall is given as follows:

\[
Recall = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{|MLC(x_i) \cap Y_i|}{|Y_i|} \right)
\]

(15)

F1 measure. F1 measure is given by the harmonic mean of Precision and Recall. The expression to evaluate F1 measure is given by,

\[
F1 - measure = \frac{1}{N} \sum_{i=1}^{N} \left( \frac{2 * |MLC(x_i) \cap Y_i|}{|MLC(x_i)| + |Y_i|} \right)
\]

(16)

A comparison of performance metrics such as hamming loss, precision, recall, accuracy and F1 measure of the proposed technique is shown in Tables 3.4-3.8. The performance of state-of-the-art techniques is adapted from [70]. From the tables, it is clear that the proposed method works uniformly well on all datasets and remains one of the top classifiers available for multi-label classification.
Table 3.4. Hamming Loss Comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CC</th>
<th>QWML</th>
<th>HOMER</th>
<th>ML- C4.5</th>
<th>PCT</th>
<th>ML- kNN</th>
<th>ECC</th>
<th>RFML- C4.5</th>
<th>RF- PCT</th>
<th>ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>0.256</td>
<td>0.254</td>
<td>0.361</td>
<td>0.247</td>
<td>0.267</td>
<td>0.294</td>
<td>0.281</td>
<td>0.198</td>
<td>0.189</td>
<td>0.251</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.193</td>
<td>0.191</td>
<td>0.207</td>
<td>0.234</td>
<td>0.219</td>
<td>0.198</td>
<td>0.207</td>
<td>0.205</td>
<td>0.197</td>
<td>0.191</td>
</tr>
<tr>
<td>Scene</td>
<td>0.082</td>
<td>0.081</td>
<td>0.082</td>
<td>0.141</td>
<td>0.129</td>
<td>0.099</td>
<td>0.085</td>
<td>0.116</td>
<td>0.094</td>
<td>0.085</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.017</td>
<td>0.012</td>
<td>0.012</td>
<td>0.01</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
</tr>
<tr>
<td>Enron</td>
<td>0.064</td>
<td>0.048</td>
<td>0.051</td>
<td>0.053</td>
<td>0.058</td>
<td>0.051</td>
<td>0.049</td>
<td>0.047</td>
<td>0.046</td>
<td>0.047</td>
</tr>
<tr>
<td>Medical</td>
<td>0.077</td>
<td>0.012</td>
<td>0.012</td>
<td>0.013</td>
<td>0.023</td>
<td>0.017</td>
<td>0.014</td>
<td>0.022</td>
<td>0.014</td>
<td>0.011</td>
</tr>
</tbody>
</table>

Figure 3.1. Hamming Loss Comparison
Table 3.5 Accuracy Comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CC</th>
<th>QW</th>
<th>HO</th>
<th>ML-ML</th>
<th>PCT</th>
<th>ML-kNN</th>
<th>ECC</th>
<th>RFML-C4.5</th>
<th>RF-PCT</th>
<th>ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>0.356</td>
<td>0.373</td>
<td>0.471</td>
<td>0.536</td>
<td>0.448</td>
<td>0.319</td>
<td>0.432</td>
<td>0.488</td>
<td>0.519</td>
<td>0.412</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.527</td>
<td>0.523</td>
<td>0.559</td>
<td>0.48</td>
<td>0.44</td>
<td>0.492</td>
<td>0.546</td>
<td>0.453</td>
<td>0.478</td>
<td>0.514</td>
</tr>
<tr>
<td>Scene</td>
<td>0.723</td>
<td>0.683</td>
<td>0.717</td>
<td>0.569</td>
<td>0.538</td>
<td>0.629</td>
<td>0.735</td>
<td>0.388</td>
<td>0.541</td>
<td>0.676</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.03</td>
<td>0.195</td>
<td>0.179</td>
<td>0.002</td>
<td>0</td>
<td>0.014</td>
<td>0.001</td>
<td>0.005</td>
<td>0.009</td>
<td>0.044</td>
</tr>
<tr>
<td>Enron</td>
<td>0.334</td>
<td>0.388</td>
<td>0.478</td>
<td>0.418</td>
<td>0.196</td>
<td>0.319</td>
<td>0.462</td>
<td>0.374</td>
<td>0.416</td>
<td>0.418</td>
</tr>
<tr>
<td>Medical</td>
<td>0.211</td>
<td>0.658</td>
<td>0.713</td>
<td>0.73</td>
<td>0.228</td>
<td>0.528</td>
<td>0.611</td>
<td>0.25</td>
<td>0.591</td>
<td>0.715</td>
</tr>
</tbody>
</table>

Figure 3.2. Accuracy Comparison
Table 3.6. Precision comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CC</th>
<th>QW</th>
<th>HO</th>
<th>ML-ML</th>
<th>PCT</th>
<th>ML-kNN</th>
<th>ECC</th>
<th>RF-ML-C4.5</th>
<th>RF-PCT</th>
<th>ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>0.551</td>
<td>0.548</td>
<td>0.509</td>
<td>0.606</td>
<td>0.577</td>
<td>0.502</td>
<td>0.58</td>
<td>0.625</td>
<td>0.644</td>
<td>0.548</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.727</td>
<td>0.718</td>
<td>0.663</td>
<td>0.62</td>
<td>0.705</td>
<td>0.732</td>
<td>0.667</td>
<td>0.738</td>
<td>0.744</td>
<td>0.718</td>
</tr>
<tr>
<td>Scene</td>
<td>0.758</td>
<td>0.711</td>
<td>0.746</td>
<td>0.592</td>
<td>0.565</td>
<td>0.661</td>
<td>0.77</td>
<td>0.403</td>
<td>0.565</td>
<td>0.685</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.042</td>
<td>0.326</td>
<td>0.317</td>
<td>0.005</td>
<td>0</td>
<td>0.035</td>
<td>0.002</td>
<td>0.018</td>
<td>0.03</td>
<td>0.144</td>
</tr>
<tr>
<td>Enron</td>
<td>0.464</td>
<td>0.624</td>
<td>0.616</td>
<td>0.623</td>
<td>0.415</td>
<td>0.587</td>
<td>0.652</td>
<td>0.69</td>
<td>0.709</td>
<td>0.668</td>
</tr>
<tr>
<td>Medical</td>
<td>0.217</td>
<td>0.697</td>
<td>0.762</td>
<td>0.797</td>
<td>0.285</td>
<td>0.575</td>
<td>0.662</td>
<td>0.284</td>
<td>0.635</td>
<td>0.774</td>
</tr>
</tbody>
</table>

Figure 3.3. Precision Comparison
Table 3.7. Recall comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CC</th>
<th>QWML</th>
<th>HOMER</th>
<th>ML-C4.5</th>
<th>PCT</th>
<th>ML-kNN</th>
<th>ECC</th>
<th>RFML-C4.5</th>
<th>RF-PCT</th>
<th>ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>0.397</td>
<td>0.429</td>
<td>0.775</td>
<td>0.703</td>
<td>0.534</td>
<td>0.377</td>
<td>0.533</td>
<td>0.545</td>
<td>0.582</td>
<td>0.491</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.6</td>
<td>0.6</td>
<td>0.714</td>
<td>0.608</td>
<td>0.49</td>
<td>0.549</td>
<td>0.673</td>
<td>0.491</td>
<td>0.523</td>
<td>0.608</td>
</tr>
<tr>
<td>Scene</td>
<td>0.726</td>
<td>0.709</td>
<td>0.744</td>
<td>0.582</td>
<td>0.539</td>
<td>0.655</td>
<td>0.771</td>
<td>0.388</td>
<td>0.541</td>
<td>0.709</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.056</td>
<td>0.264</td>
<td>0.25</td>
<td>0.002</td>
<td>0</td>
<td>0.014</td>
<td>0.001</td>
<td>0.005</td>
<td>0.009</td>
<td>0.043</td>
</tr>
<tr>
<td>Enron</td>
<td>0.507</td>
<td>0.453</td>
<td>0.61</td>
<td>0.487</td>
<td>0.229</td>
<td>0.358</td>
<td>0.56</td>
<td>0.398</td>
<td>0.452</td>
<td>0.508</td>
</tr>
<tr>
<td>Medical</td>
<td>0.754</td>
<td>0.801</td>
<td>0.76</td>
<td>0.74</td>
<td>0.227</td>
<td>0.547</td>
<td>0.642</td>
<td>0.251</td>
<td>0.599</td>
<td>0.744</td>
</tr>
</tbody>
</table>

Figure 3.4. Recall Comparison
Table 3.8. F1-measure comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CC</th>
<th>QWM</th>
<th>HOME</th>
<th>ML-C4.5</th>
<th>PCT</th>
<th>ML-kNN</th>
<th>ECC</th>
<th>RFML-C4.5</th>
<th>RF-PCT</th>
<th>ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>0.461</td>
<td>0.481</td>
<td>0.614</td>
<td>0.651</td>
<td>0.554</td>
<td>0.431</td>
<td>0.556</td>
<td>0.583</td>
<td>0.611</td>
<td>0.518</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.657</td>
<td>0.654</td>
<td>0.687</td>
<td>0.614</td>
<td>0.578</td>
<td>0.628</td>
<td>0.67</td>
<td>0.589</td>
<td>0.614</td>
<td>0.658</td>
</tr>
<tr>
<td>Scene</td>
<td>0.742</td>
<td>0.71</td>
<td>0.745</td>
<td>0.587</td>
<td>0.551</td>
<td>0.658</td>
<td>0.771</td>
<td>0.395</td>
<td>0.553</td>
<td>0.697</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.048</td>
<td>0.292</td>
<td>0.28</td>
<td>0.003</td>
<td>0</td>
<td>0.021</td>
<td>0.001</td>
<td>0.008</td>
<td>0.014</td>
<td>0.033</td>
</tr>
<tr>
<td>Enron</td>
<td>0.484</td>
<td>0.525</td>
<td>0.613</td>
<td>0.546</td>
<td>0.295</td>
<td>0.445</td>
<td>0.602</td>
<td>0.505</td>
<td>0.552</td>
<td>0.577</td>
</tr>
<tr>
<td>Medical</td>
<td>0.337</td>
<td>0.745</td>
<td>0.761</td>
<td>0.768</td>
<td>0.253</td>
<td>0.56</td>
<td>0.652</td>
<td>0.267</td>
<td>0.616</td>
<td>0.759</td>
</tr>
</tbody>
</table>

Figure 3.5. F1-measure Comparison
3.3.3 Execution Speed

The performance of the proposed method in terms of execution speed is evaluated by comparing the training time and the testing time of the algorithm used. The proposed method is applied to 6 datasets of different domains with a wide range of label density and label cardinality and the training time and the testing time are compared with other state-of-the-art techniques. The comparison table of training time and testing time is given in Table 3.9 and Table 3.10 respectively.

Table 3.9. Comparison of Training Time (in seconds)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CC</th>
<th>QWML</th>
<th>HOMER</th>
<th>ML-C4.5</th>
<th>PCT</th>
<th>ML-kNN</th>
<th>ECC</th>
<th>RFML-C4.5</th>
<th>RF-PCT</th>
<th>ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>0.3</td>
<td>0.1</td>
<td>0.4</td>
<td>4.9</td>
<td>1.2</td>
<td>2.9</td>
<td>0.04</td>
</tr>
<tr>
<td>Yeast</td>
<td>206</td>
<td>672</td>
<td>101</td>
<td>14</td>
<td>1.5</td>
<td>8.2</td>
<td>497</td>
<td>19</td>
<td>25</td>
<td>0.2</td>
</tr>
<tr>
<td>Scene</td>
<td>99</td>
<td>195</td>
<td>68</td>
<td>8</td>
<td>2</td>
<td>14</td>
<td>319</td>
<td>10</td>
<td>23</td>
<td>0.12</td>
</tr>
<tr>
<td>Corel5k</td>
<td>1225</td>
<td>2388</td>
<td>771</td>
<td>369</td>
<td>30</td>
<td>389</td>
<td>10073</td>
<td>385</td>
<td>902</td>
<td>0.6</td>
</tr>
<tr>
<td>Enron</td>
<td>440</td>
<td>971</td>
<td>158</td>
<td>15</td>
<td>1.1</td>
<td>6</td>
<td>1467</td>
<td>25</td>
<td>47</td>
<td>0.26</td>
</tr>
<tr>
<td>Medical</td>
<td>28</td>
<td>40</td>
<td>16</td>
<td>3</td>
<td>0.6</td>
<td>1</td>
<td>103</td>
<td>7</td>
<td>27</td>
<td>0.11</td>
</tr>
</tbody>
</table>
Table 3.10. Comparison of Testing Time (in seconds)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CC</th>
<th>QWML</th>
<th>HOMER</th>
<th>ML-C4.5</th>
<th>ML-kNN</th>
<th>ML-C4.5</th>
<th>RF-C4.5</th>
<th>RF-PCT</th>
<th>ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emotion</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>6.6</td>
<td>0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>Yeast</td>
<td>25</td>
<td>64</td>
<td>17</td>
<td>0.1</td>
<td>0</td>
<td>5</td>
<td>158</td>
<td>0.5</td>
<td>0.2</td>
</tr>
<tr>
<td>Scene</td>
<td>25</td>
<td>40</td>
<td>21</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>168</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Corel5k</td>
<td>31</td>
<td>119</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>45</td>
<td>2077</td>
<td>1.8</td>
<td>2.5</td>
</tr>
<tr>
<td>Enron</td>
<td>53</td>
<td>174</td>
<td>22</td>
<td>0.2</td>
<td>0</td>
<td>3</td>
<td>696</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Medical</td>
<td>6</td>
<td>25</td>
<td>1.5</td>
<td>0.1</td>
<td>0</td>
<td>0.2</td>
<td>46</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

In summary, the proposed method outperforms all existing multi-label learning techniques in terms of training and testing time by several orders of magnitude. From the results, it can be seen that the proposed method is the fastest multi-label classifier when compared to the current state-of-the-arts techniques. The speed of the proposed classifier is many-fold greater than existing methods. Also, from the comparison results of other performance metrics such as hamming loss, accuracy, precision, recall and F1-measure, it can be seen that the proposed method remains one of the top positions in each case. Also, the F1-measure of the proposed approach outperforms the most recent method which uses canonical correlation analysis (CCA) with ELM for multi-label problems [27] in most cases.
The key advantage of the proposed method is that it performs uniformly well on all datasets and surpasses all existing state-of-the-arts methods in terms of speed.

3.4 Summary

The proposed high speed multi-label classifier executes with both fast speed and high accuracy. It is to be highlighted that there are no extreme-learning-machine-based multi-label classifiers existing in the literature thus far. The proposed method is applied to 6 benchmark datasets of different domains and a wide range of label density and label cardinality. The results are compared with 9 state-of-the-arts multi-label classifiers. It can be seen from the results that the proposed method performs uniformly well across all datasets and surpasses all state-of-the-arts methods in terms of speed. Thus, the proposed ELM-based multi-label classifier can be a better alternative for a wide range of multi-label classification techniques in order to achieve greater accuracy and very high speed.
Chapter 4

Online Learning – Multi-label Classification

In this chapter, a high-speed online neural network classifier based on extreme learning machines for multi-label classification is proposed. Multi-label classification problems are far more complex than binary and multi-class classification problems, as both the number of target labels and each of the target labels corresponding to each of the input samples are to be identified. The proposed work exploits the high-speed nature of the extreme learning machines to achieve real-time multi-label classification of streaming data. A new threshold-based online sequential learning algorithm is proposed for high speed and streaming data classification of multi-label problems. The proposed method is experimented with six different datasets from different application domains such as multimedia, text, and biology. The hamming loss, accuracy, training time and testing time of the
proposed technique is compared with nine different state-of-the-art methods. Experimental studies show that the proposed technique performs significantly well across various datasets in terms of performance and speed. In this chapter, an ELM based classifier that is capable of performing online multi-label classification on streaming data in real-time is developed. There are no online multi-label techniques available in the literature which can perform real-time multi-label classification. The proposed technique is experimented on six datasets from different application domains.

As discussed in Section 2.1, several approaches for solving multi-label problems are available in the literature. But most of the available approaches are based on batch learning techniques. Online techniques for multi-label classification are still greatly to be explored. The paper on streaming multi-label classification by Read and his team [107] list out the existing classifiers on multi-label classification for streaming applications. The existing techniques listed belongs to the category of problem transformation methods. In problem transformation methods, the multi-label classification problem is converted into multiple single-label classification problem and it uses existing single-label techniques for classification. The proposed method, on the other hand extends the base algorithm itself to adapt to the multi-label problems. Therefore, the proposed method differs significantly from the existing problem transformation based techniques. It is also to be highlighted that the proposed method is the first neural network based real-time online multi-label classifier. The proposed method employs a new threshold-based classification for multi-label problems. Unlike single-label classification, the number of target
labels differs for every data sample. Therefore, in multi-label classification, both the number of labels and the corresponding labels are unknown. Also, different multi-label datasets differ significantly from each other with respect to label density and label cardinality characteristics. A classifier that performs well in one dataset may not necessarily perform well in a different dataset. Due to the increased complexity of the multi-label classification caused by its generality, the time taken for training the classifier is high for most of the techniques. Also, the highly complex nature of the multi-label classification problems poses a considerable challenge in developing high-speed real-time online classifiers. Several real world applications require the need for multi-label classification. High-speed processing of streaming data for multi-label classification is highly essential for real-world real-time applications. The proposed work exploits the high-speed nature of extreme learning machines, and a novel online multi-label classifier is developed.

4.1 Proposed OSML-ELM

The various steps involved in the proposed method are briefly stated. The key novelty of the proposed method is that, there are no online multi-label classification techniques available in the literature that can perform classification in real-time on streaming data. The proposed method is the multi-label formulation of the online sequential extreme learning machine and hence called Online Sequential Multi-label ELM (OSML-ELM).
**Initialization of Parameters.** Fundamental parameters such as the number of hidden layer neurons and the activation function are initialized. Sigmoidal activation function is used for the experimentation. The problem of overfitting is tackled by using the early stopping technique. In early stopping technique, the point at which the training accuracy increases at the expense of generalization error is identified and further training is stopped. The number of hidden neurons are selected depending upon the nature and complexity of the dataset while preventing the overfitting of data.

**Processing of Inputs.** In traditional single-label problems, the target class will be a single-label associated with the input sample. But, in the multi-label case, each input sample can be associated with more than one class labels. Hence, each of the input samples will have the associated output label as an m-tuple with 0 or 1 representing the belongingness to each of the labels in the label space L. This is a key difference between the inputs available for single-label and multi-label problems. As opposed to single-label classification with a single target label, the multi-label problem has a target label set which is a subset of label space L. The label set denoting the belongingness for each of the labels is converted from unipolar representation to bipolar representation.

**ELM Training.** The processed input is then supplied to the online sequential variant of ELM technique. Let H be the hidden layer output matrix, β be the output weights and Y be the target label, the ELM can be represented in a compact form as Hβ = Y where Y⊆L, L = {ζ₁, ζ₂,..., ζₘ}. During the training phase, Let N₀ be
the number of samples in the initial block of data that is provided to the network. The initial output weight $\beta_0$ is calculated from equation 9 and 10.

$$\beta = H^+Y \text{ and } H^+ = (H^TH)^{-1}H^T,$$

Consider $M_0 = (H_0^TH_0)^{-1}$, therefore, $\beta_0 = M_0H_0^TY_0$.

For each of the subsequent sequentially arriving data, the output weights can be updated by incorporating the recursive least square algorithm with the ELM learning as

$$M_{k+1} = M_k - \frac{M_kh_{k+1}h_{k+1}^TM_k}{1 + h_{k+1}^TM_kh_{k+1}} \quad (17)$$

$$\beta_{k+1} = \beta_k + M_{k+1}h_{k+1}(Y_{k+1}^T - h_{k+1}^T\beta_k) \quad (18)$$

where $k = 0,1,2,\ldots N-N_0-1$.

The detailed mathematics and derivation behind the recursive least square based online ELM learning called online-sequential extreme learning machine is discussed in detail in several literatures.

ELM Testing. In the testing phase, the test data sample is evaluated using the values of $\beta$ obtained during the training phase. The input data that can be a combination of Boolean, discrete and continuous data type is given to the network. The network then computes $Y = H\beta$. The predicted output $Y$ obtained is a set of real numbers of dimension equal to the number of labels.
Post-processing and Multi-label Identification. The prime step in extending the ELM based technique for online multi-label problems is the post-processing and thresholding. In binary and multi-class classification, each of the input sample belongs to only one target label and, therefore, can be identified as the index of the maximum value in the predicted output. On contrary, in multi-label classification the number of labels each sample belongs to is not constant. Each input sample may belong to one or more than one of the target labels. Therefore, the classifier must predict both the number of labels and each of the corresponding labels for the input data sample. The number of labels corresponding to a data sample is completely unknown. Hence, in the proposed method, a thresholding-based label association is proposed. The threshold value is selected during the training phase such that it maximizes the separation between the family of labels the input belongs to and the family of labels the input does not belong to, based on the raw output values \( Y \). Setting up of the threshold value is of prime importance as it directly affects the performance of the classifier. The \( L \) dimensioned raw-predicted output is compared with a unique threshold value. The index values of the predicted output \( Y \) which are greater than the fixed threshold represent the belongingness of the input sample to the corresponding class.

Setting the threshold value is of critical importance. The threshold value is selected such that it maximizes the difference between the category of labels to which the sample belongs to and the category of labels to which the sample does not belong to with respect to the raw output values \( Y \) obtained during the training phase. The distribution of the raw output values of \( Y \) for categories of labels that
the input sample belongs to \((Y_A)\) and the categories of labels the input sample does not belong to \((Y_B)\) are identified. Based on the distribution of \(Y_A\) and \(Y_B\), a threshold value is identified using the formula,

\[
\text{Threshold value} = \frac{\min(Y_A) + \max(Y_B)}{2}
\]  \hspace{1cm} (19)

As a trivial case, the threshold can be set as zero. In which case, the raw predicted output values will be passed as arguments to a bipolar step function. The threshold value is compared with the raw output values of \(Y\) estimated by the classifier, and the number of target labels corresponding to the data sample is identified. Then, based on the threshold value, the subset of labels that corresponds to the input data sample is recognized. The threshold value is determined by analyzing the distribution of the raw predicted output values during the training phase. From the distribution, a particular value is chosen that maximizes the separation between the two categories of the labels. The proposed method belongs to the category of algorithm adaptation method, where the base algorithm is adapted to perform multi-label classification problems. It is to be highlighted that there are no ELM-based online multi-label classifiers in the literature thus far. The proposed method is the first to adapt the ELM for online multi-label problems and make extensive experimentation, results comparison and analysis with the state-of-the-art techniques. The overview of the proposed algorithm is summarized.

**Algorithm 2: Proposed OSML-ELM algorithm**

1. Initialization: The parameters of the network are initialized.
2. Pre-processing: Essential pre-processing of input data is performed. Each input
sample \( x(i) \) is associated with an \( m \)-tuple target labels \( (T(i)) \) with values representing the belongingness of the sample to the corresponding label.

3. For initial block of data with \( N_0 \) samples, \( M_0 = (H_0^T H_0)^{-1} \)

4. Output weight vector of initial block of data is calculated: \( \beta_0 = M_0 H_0^T Y_0 \)

5. For each of the sequentially arriving data, the model parameters are updated as

\[
\text{for } i = N_0 + 1 \text{ to } N
\]

\[
k = i - N_0
\]

\[
M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}}
\]

\[
\beta_{k+1} = \beta_k + M_{k+1} h_{k+1} (y_{k+1}^T - h_{k+1}^T \beta_k)
\]

6. Prediction on training data: \( Y_{\text{train}} = H_{\text{train}} \beta \)

7. Threshold identification: From set of target labels \( T \) and the distribution of raw output values \( Y_{\text{train}} \), a threshold value \( (th) \) is identified.

\[
\text{Threshold value, } \text{th} = (\min(Y_A) + \max(Y_B))/2
\]

8. Hidden layer output matrix (testing phase): \( H_{\text{test}} = [h_1, \ldots, h_{N'}]^T \)

9. Prediction on testing data: \( Y_{\text{test}} = H_{\text{test}} \beta \)

10. Multi-label classification: In multi-label classification both the number of labels and the subset of target label that correspond to the input are to be identified.

11. Identification of number of labels: Number of labels corresponding to each input \( x(i) \) is identified by using the raw output values \( Y_{\text{test}}(i) \) and \( th \).

\[
\text{Number of labels } = \sum HS(Y_{\text{test}}(i) - \text{th})
\]

12. where, \( HS(x) \) is the standard heaviside step function.

13. Identification of each of the output labels: The set of labels that corresponds to
the indices value for which $Y_{\text{test}}(i) > th$ is true gives the output labels corresponding to the input $x(i)$

4.2 Experimentation

This section describes the different multi-label dataset metrics and gives the experimental design used to evaluate the proposed method.

As discussed in Chapter 3, Multi-label datasets have a unique property called the degree of multi-labelness. In order to quantitatively measure the multi-labelness of a dataset, two dataset metrics are available in the literature. They are Label Cardinality (LC) and Label Density (LD). Not all datasets are equally multi-labelled. The number of labels, the number of samples having multiple labels, the average number of labels corresponding to a particular sample varies among different datasets resulting in a varied degree of multi-labelness to a dataset. It is to be noted that, two datasets with same label cardinality but different label density can significantly vary and may result in different behavior of the training algorithm. The influence of label density and label cardinality on multi-label learning is analyzed by.

The proposed method is experimented with five benchmark datasets comprising of different application areas such as multimedia, text, and biology. The datasets are chosen from different application domains and exhibit wide range
of label density and label cardinality. The number of target class labels ranges from 6 labels to 374 labels, and the number of features or attributes in the dataset ranges from 103 to 1449. The dataset metrics such as label cardinality varies from as low as 1.07 to as high as 4.24. Label cardinality of 1.07 represents that each of the input samples corresponds to 1.07 labels on average. Label cardinality of 4.24 signifies that each sample on an average corresponds to 4.24 labels. Since the label density is inversely proportional to the number of labels present in the dataset, lower the label density value indicates that only fewer samples correspond to a particular label, thus, posing a challenge for the multi-label techniques to train fast enough so as to learn the target label set within the limited samples. The specifications of the datasets are given in Table 4.1. The number of samples in each of the dataset used for training and testing phase and the feature dimension are included in the dataset specifications. The datasets are obtained from KEEL multi-label dataset repository.
Table 4.1. Dataset Specifications

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Domain</th>
<th>No. of Features</th>
<th>No. of Samples</th>
<th>#Train</th>
<th>#Test</th>
<th>No. of Labels</th>
<th>LC</th>
<th>LD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>Biology</td>
<td>103</td>
<td>2417</td>
<td>1600</td>
<td>817</td>
<td>4.24</td>
<td>4.24</td>
<td>0.303</td>
</tr>
<tr>
<td>Scene</td>
<td>Multimedia</td>
<td>294</td>
<td>2407</td>
<td>2000</td>
<td>407</td>
<td>1.07</td>
<td>1.07</td>
<td>0.178</td>
</tr>
<tr>
<td>Corel5k</td>
<td>Multimedia</td>
<td>499</td>
<td>5000</td>
<td>4500</td>
<td>500</td>
<td>3.52</td>
<td>3.52</td>
<td>0.009</td>
</tr>
<tr>
<td>Enron</td>
<td>Text</td>
<td>1001</td>
<td>1702</td>
<td>1200</td>
<td>502</td>
<td>3.38</td>
<td>3.38</td>
<td>0.064</td>
</tr>
<tr>
<td>Medical</td>
<td>Text</td>
<td>1449</td>
<td>978</td>
<td>700</td>
<td>278</td>
<td>1.25</td>
<td>1.25</td>
<td>0.027</td>
</tr>
</tbody>
</table>

The hamming loss, training and testing time of the proposed method are compared to 9 different multi-label techniques available in the literature. The 9 techniques are chosen such that they are from PT, AA and EN methods. The implementation procedure of all the 9 techniques are adapted from the extensive experimental comparison work on multi-label classifiers by Madjarov and team. Also, the chosen techniques belong to different learning paradigms such as SVM, decision trees, and nearest neighbors. The details of state-of-the-arts multi-label techniques used for result comparison are given in Table 4.2.
4.3 Results and Discussions

The proposed method is experimented with each of the datasets mentioned in Table 3 and is compared with 9 state-of-the-art multi-label classification techniques. Also, the performance of the proposed method is compared with the state-of-the-art online multi-label technique. This section discusses the results obtained by the proposed method and compares it with the existing methods. The results obtained from the proposed method are evaluated for consistency, performance, and speed.

4.3.1 Consistency

Consistency is a key feature that is essential for any new technique proposed. Any technique proposed should provide consistent results for multiple trials with minimal variance. The consistency of a technique can be identified using cross-validation procedure. Therefore, a 5-fold cross validation and a 10-fold cross validation is performed on the proposed technique for each of the 5 datasets. Since the initial weights are assigned randomly for an ELM based technique, it is critical to evaluate the consistency of the proposed technique.
Table 4.2. Comparison Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Method Category</th>
<th>Machine Learning Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifier Chain (CC)</td>
<td>PT</td>
<td>SVM</td>
</tr>
<tr>
<td>QWeighted approach for Multi-label Learning (QWML)</td>
<td>PT</td>
<td>SVM</td>
</tr>
<tr>
<td>Hierarchy Of Multi-label Classifiers (HOMER)</td>
<td>PT</td>
<td>SVM</td>
</tr>
<tr>
<td>Multi-Label C4.5 (ML-C4.5)</td>
<td>AA</td>
<td>Decision Trees</td>
</tr>
<tr>
<td>Predictive Clustering Trees (PCT)</td>
<td>AA</td>
<td>Decision Trees</td>
</tr>
<tr>
<td>Multi-Label k-Nearest Neighbors (ML-kNN)</td>
<td>AA</td>
<td>Nearest Neighbors</td>
</tr>
<tr>
<td>Ensemble of Classifier Chains (ECC)</td>
<td>EN</td>
<td>SVM</td>
</tr>
<tr>
<td>Random Forest Predictive Clustering Trees (RF-PCT)</td>
<td>EN</td>
<td>Decision Trees</td>
</tr>
<tr>
<td>Random Forest of ML-C4.5 (RFML-C4.5)</td>
<td>EN</td>
<td>Decision Trees</td>
</tr>
</tbody>
</table>

The unique feature of multi-label classification is the possibility of the partial correctness of the classifier. Therefore, calculating the error rate for multi-label problems is not same as that of traditional binary or multi-class problems. One or more of the multiple labels to which the sample instance belongs and/or the number of labels the sample instance belongs can be identified partially correctly resulting in the partial correctness of the classifier. Hence, the hamming loss
performance metric is used to quantitatively measure the correctness of the classifier. The hamming loss is a measure of the misclassification rate of the learning technique. The lower the hamming loss, the better is the classification accuracy. The definition and the equation for evaluation of the hamming loss has been discussed earlier in Chapter 3.

To evaluate the consistency of the proposed method, a 5-fold and a 10-fold cross validation of hamming loss metric is carried out for each of the 5 datasets and is tabulated.

Table 4.3. Consistency Table – Cross Validation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Hamming Loss - 5-fcv</th>
<th>Hamming Loss - 10-fcv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>0.206 ± 0.001</td>
<td>0.206 ± 0.002</td>
</tr>
<tr>
<td>Scene</td>
<td>0.098 ± 0.002</td>
<td>0.098 ± 0.002</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.009 ± 0.000</td>
<td>0.009 ± 0.000</td>
</tr>
<tr>
<td>Enron</td>
<td>0.049 ± 0.001</td>
<td>0.049 ± 0.001</td>
</tr>
<tr>
<td>Medical</td>
<td>0.011 ± 0.001</td>
<td>0.011 ± 0.001</td>
</tr>
</tbody>
</table>

From Table 4.3, it can be seen that the proposed technique is consistent in its performance over repeated executions and cross validations, thus, demonstrating the consistency of the technique.
4.3.2 Performance Metrics

Due to the possibility of the partial correctness of the classifier result, one specific metric will not be sufficient to quantitatively measure the performance of a technique. Therefore, a set of quantitative performance evaluation metrics is used to validate the performance of the multi-label classifier. The performance metrics used are hamming loss, accuracy, precision, recall and F1-measure. The definition and the equations for evaluating each of the performance metrics has been discussed and analyzed earlier in Chapter 3. The proposed method is experimented on five different datasets for the five different performance metrics, and the results are tabulated. From Table 4.4, it can be seen that, the proposed method has a very low hamming loss and better performance metric measures for a wide range of datasets irrespective of the label density and label cardinality values.

Table 4.4. Performance metrics of OSML-ELM

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Hamming Loss</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>0.206</td>
<td>0.493</td>
<td>0.693</td>
<td>0.580</td>
<td>0.632</td>
</tr>
<tr>
<td>Scene</td>
<td>0.098</td>
<td>0.610</td>
<td>0.630</td>
<td>0.645</td>
<td>0.637</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.009</td>
<td>0.060</td>
<td>0.175</td>
<td>0.063</td>
<td>0.093</td>
</tr>
<tr>
<td>Enron</td>
<td>0.049</td>
<td>0.404</td>
<td>0.640</td>
<td>0.461</td>
<td>0.536</td>
</tr>
<tr>
<td>Medical</td>
<td>0.011</td>
<td>0.713</td>
<td>0.760</td>
<td>0.740</td>
<td>0.750</td>
</tr>
</tbody>
</table>
4.3.3 Comparison with State-of-the-Arts Techniques

The performance of the proposed method is compared with nine state-of-the-art techniques as specified in Table 4.2. Hamming loss performance metric provides the percentage of wrong labels to the total number of labels. Accuracy performance metric provides the ratio of the total number of correctly predicted labels to the total number of labels of that sample instance. Therefore, hamming loss and accuracy are the key performance metrics for evaluating the performance of the proposed method. The hamming loss and accuracy metrics are used to compare the performance of the proposed technique with the state-of-the-art techniques. The comparison results are given in Figure 4.1 and Figure 4.2 respectively.

![Hamming Loss Comparison Graph](image_url)

Figure 4.1. Hamming Loss Comparison
Hamming loss is the measure of misclassification in the dataset. Lower the hamming loss, better the performance of the classifier. For an ideal classifier, the hamming loss is equal to zero. Accuracy is the ratio of number of correctly predicted labels to the total number of labels for a given sample. Higher the accuracy, better the performance of the classifier. It is evident from the figure that, among the 10 different multi-label classifiers, the proposed method performs uniformly well on all datasets and remains one of the best classifiers for solving wide range of multi-label classification problems.

4.3.4 Comparison of Execution Speed

The performance of the proposed method in terms of execution speed is evaluated by comparing the training time and the testing time of the algorithm used. The proposed method is applied to 5 different datasets from various
application domains and a wide range of label density and label cardinality values. The comparison of training time and testing time of the proposed method with existing state-of-the-art methods are tabulated in Tables 4.5 and 4.6.

Table 4.5. Training Time Comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>C</th>
<th>Q</th>
<th>W</th>
<th>ML</th>
<th>ML-C4.5</th>
<th>PC</th>
<th>ML-kNN</th>
<th>EC</th>
<th>RFML-C4.5</th>
<th>RF-PCT</th>
<th>OSML-ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>20</td>
<td>672</td>
<td>101</td>
<td>14</td>
<td>1.5</td>
<td>8.2</td>
<td>497</td>
<td>19</td>
<td>25</td>
<td>0.114</td>
<td></td>
</tr>
<tr>
<td>Scene</td>
<td>99</td>
<td>195</td>
<td>68</td>
<td>8</td>
<td>2</td>
<td>14</td>
<td>319</td>
<td>10</td>
<td>23</td>
<td>2.329</td>
<td></td>
</tr>
<tr>
<td>Corel5k</td>
<td>12</td>
<td>238</td>
<td>771</td>
<td>369</td>
<td>30</td>
<td>389</td>
<td>100</td>
<td>73</td>
<td>385</td>
<td>902</td>
<td>5.365</td>
</tr>
<tr>
<td>Enron</td>
<td>44</td>
<td>971</td>
<td>158</td>
<td>15</td>
<td>1.1</td>
<td>6</td>
<td>146</td>
<td>7</td>
<td>25</td>
<td>47</td>
<td>0.630</td>
</tr>
<tr>
<td>Medical</td>
<td>28</td>
<td>40</td>
<td>16</td>
<td>3</td>
<td>0.6</td>
<td>1</td>
<td>103</td>
<td>7</td>
<td>27</td>
<td>0.663</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6. Testing Time Comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>C</th>
<th>Q</th>
<th>W</th>
<th>ML</th>
<th>ML-C4.5</th>
<th>PC</th>
<th>ML-kNN</th>
<th>EC</th>
<th>RFML-C4.5</th>
<th>RF-PCT</th>
<th>OSML-ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>25</td>
<td>64</td>
<td>17</td>
<td>0.1</td>
<td>0</td>
<td>5</td>
<td>158</td>
<td>0.5</td>
<td>0.2</td>
<td>0.017</td>
<td></td>
</tr>
<tr>
<td>Scene</td>
<td>25</td>
<td>40</td>
<td>21</td>
<td>1</td>
<td>0</td>
<td>14</td>
<td>168</td>
<td>2</td>
<td>1</td>
<td>0.047</td>
<td></td>
</tr>
<tr>
<td>Corel5k</td>
<td>31</td>
<td>119</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>45</td>
<td>207</td>
<td>7</td>
<td>1.8</td>
<td>2.5</td>
<td>0.076</td>
</tr>
<tr>
<td>Enron</td>
<td>53</td>
<td>174</td>
<td>22</td>
<td>0.2</td>
<td>0</td>
<td>3</td>
<td>696</td>
<td>1</td>
<td>1</td>
<td>0.028</td>
<td></td>
</tr>
<tr>
<td>Medical</td>
<td>6</td>
<td>25</td>
<td>1.5</td>
<td>0.1</td>
<td>0</td>
<td>0.2</td>
<td>46</td>
<td>0.5</td>
<td>0.5</td>
<td>0.039</td>
<td></td>
</tr>
</tbody>
</table>
From Tables 4.5 and 4.6, it can be clearly seen that the proposed OSML-ELM outperforms all the existing techniques in terms of execution speed. Despite being an online learning algorithm, the speed of the proposed OSML-ELM is several folds faster than most of the existing batch learning techniques. This high speed nature of the OSML-ELM will enable it to perform real-time multi-label classification on streaming data. It is to be highlighted that there are no existing techniques in the literature that can perform real-time online multi-label classification.

4.3.5 Real-Time Classification

For an online classifier to perform classification in real-time, the time taken for executing a single block of data (epoch) should be very low. If the time taken for processing an epoch is more than the rate of arrival of the sequential data, real-time processing of the streaming data cannot be achieved. From the results obtained for the training time of the classifier, the average time required for the execution of a single block of data can be estimated. The number of epochs is identified by the number of times the sequential learning phase is executed while experimenting with the specific dataset. The average time of execution to process a single block of data for the five different datasets are tabulated.
Table 4.7. Average Time per Epoch

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training Time (s)</th>
<th>Number of epochs</th>
<th>Average time(s)/epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>0.114</td>
<td>51</td>
<td>0.00223529</td>
</tr>
<tr>
<td>Scene</td>
<td>2.329</td>
<td>48</td>
<td>0.04852083</td>
</tr>
<tr>
<td>Corel5k</td>
<td>5.365</td>
<td>93</td>
<td>0.05768817</td>
</tr>
<tr>
<td>Enron</td>
<td>0.63</td>
<td>48</td>
<td>0.013125</td>
</tr>
<tr>
<td>Medical</td>
<td>0.663</td>
<td>37</td>
<td>0.01791892</td>
</tr>
</tbody>
</table>

From Table 4.7, it can be seen that the proposed OSML-ELM can perform multi-label classification for streaming data applications with high accuracy and high speed. Also, the proposed method is compared with the state-of-the-art online multi-label technique by [76]. Since the active learning technique is multimedia specific, scene dataset is used to compare the performance. The paper lists the F1 score of the active learning method for scene dataset to be 0.5612. The proposed method achieves a F1 score of 0.6371 for the same scene dataset and is achieved in real-time streaming data. This shows that the proposed method has performed better in terms of speed and performance over the existing multi-label classifiers. The key advantage of the proposed method is that OSML-ELM is capable of performing multi-label classification in real time. It is to be highlighted that there are no online multi-label classifiers that can perform the multi-label classification in real time.
4.3.6 Experimentation on Real-world Dataset

The problem of multi-label classification has a wide range of application domains as discussed in Chapter 1. In order to evaluate the performance of the proposed method with real-world dataset, the problem of aspect detection in the domain of sentiment analysis is chosen. Aspect detection is the problem of identifying the “aspect terms” that are present in a sentence. Each sentence in a dataset will have multiple aspect terms and the number of aspect terms in each of the sentence can vary. Thus making aspect detection a typical multi-label problem. For experimentation, we have chosen two datasets belonging to two different domains from SemEval 2014 [102]. They are, Laptop review and Restaurant review datasets. Each of the sentences (samples) in the datasets are the customer reviews. The Laptop review dataset contains a total of 1048 labels resulting in the label cardinality of 0.7785 and label density of 0.0007. The restaurant review dataset contains a total of 1295 labels with label cardinality of 1.21 and label density of 0.0009. From this dataset, we remove the single-occurrence labels and only use the recurring labels for aspect detection. The specifications of the datasets used for experimental analysis are given in Table 4.8.
The proposed method is experimented with the laptop and restaurant review datasets for aspect detection. The performance of the proposed method is compared with other domain specific state of the art technique available for aspect detection by Poria et. al (2016) [103]. The results obtained by the proposed method is compared with the results obtained by Poria et. al’s method. Precision, Recall and F-measure are used as performance metrics for evaluation. The results obtained are tabulated in Table 4.9. From the table, it can be seen that, similar to the benchmark datasets, the proposed method performs well in real-world datasets also.
### Table 4.9: Experimental Results Comparison for Aspect Detection

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Poria et al [103]</th>
<th>OSML-ELM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Laptop</td>
<td>0.867</td>
<td>0.783</td>
</tr>
<tr>
<td>Restaurant</td>
<td>0.882</td>
<td>0.861</td>
</tr>
</tbody>
</table>

#### 4.4 Summary

The proposed OSML-ELM classifier performs uniformly well across various multi-label datasets in terms of speed and performance. The application areas of multi-label classification are rapidly increasing due to its generality and several real-world applications require the need for multi-label classification. Due to its wide variations in the characteristics of multi-label datasets based on label density and label cardinality, a classifier that performs well for one dataset might not perform well for a different dataset. Also, high-speed real-time classification of multi-label data is required for real-world applications. The OSML-ELM method is
a novel generic real-time multi-label classifier that performs uniformly well on datasets of a wide range of label density, label cardinality and application domains. The performance of the proposed method is compared with five datasets and nine different state-of-the-arts techniques. The proposed method is also experimented on real-world datasets for aspect detection. It can be seen from the results that the proposed OSML-ELM is a key progress in achieving real-time multi-label classification for streaming data applications. The proposed OSML-ELM can be extended further to learn new data labels progressively from the data stream by retaining the previously learnt knowledge without the need for retraining.
Chapter 5

Online Learning – Universal Classifier

Classification involves the learning of the mapping function that associates input samples to corresponding target label. There are two major categories of classification problems: Single-label classification and Multi-label classification. Traditional binary and multi-class classifications are sub-categories of single-label classification. Several classifiers are developed for binary, multi-class and multi-label classification problems, but there are no classifiers available in the literature capable of performing all three types of classification. In this chapter, a novel online label-independent/universal classifier capable of performing all the three types of classification is developed. Being a high speed online classifier, the proposed technique can be applied to streaming data applications. The performance of the developed classifier is evaluated using datasets from binary, multi-class and multi-label problems. The results obtained are compared with state-of-the-art techniques from each of the classification types.
Though there are several machine learning techniques available for binary, multi-class and multi-label classifications individually, there are no classifiers available in the literature that is capable of performing all three types of classification. In this chapter, an extreme learning machine based online universal classifier that is independent of classification type and can perform all three types of classification (binary, multi-class and multi-label) is developed. The developed classifier will identify both the classification type and the target labels associated with the input samples. The classifier is experimented with datasets corresponding to each of the three classification types and is evaluated for consistency, speed and performance. The results are compared with the state-of-the-art techniques of the individual classification type. The key novelty of the research work is the development of a label-independent/universal classifier capable of performing binary, multi-class and multi-label classification on streaming data. Through experiments involving datasets from all three classification types, different performance metrics are evaluated and compared with the classifiers of individual classification types.

5.1 Proposed Approach

An online label-independent/universal classifier capable of performing classification on binary, multi-class and multi-label datasets is developed. It is to be highlighted that there are no universal classifiers available in the literature that can classify all three classification types. Also, the proposed method is an online classifier and hence can be used for streaming data applications. Binary and multi-
class classification are sub categories of single-label classification. As foreshadowed, in single-label classification, each sample is associated with only one target label. Hence, the number of target labels associated with the input sample is 1 for binary and multi-class classification. In multi-label classification, each of the data sample corresponds to a subset of target labels. Therefore, both the number of target labels and the identity of the target labels are to be predicted for multi-label classification. This results in increased complexity in achieving universal classification technique. There are three key challenges to be addressed to achieve label-independent classifier (LIC).

1. Identification of classification type

2. Estimating the number of target labels corresponding to each input sample

3. Identifying each of the associated target labels.

The proposed approach is based on the online variant of ELM called online sequential extreme learning machine. The proposed approach falls under the category of an algorithm adaptation method in which the base algorithm is extended to adapt to the requirements of the universal classification. The various phases of the proposed algorithm are summarized.

**Initialization Phase.** Initialization Phase involves setting up the fundamental network parameters for the target classification problem. Being an ELM based technique, the input weights and the bias values are randomly initialized. The number of hidden layer neurons and the activation function are assigned. The
number of hidden layer neurons is to be selected such that the problem of overfitting is avoided.

**Data Pre-processing Phase.** The proposed algorithm needs to be capable of classifying both single-label and multi-label classification problems. The representation of data varies among each of the classification types. In binary and multi-class classification, the output is represented as a single value which identifies the unique target class that is associated with the input sample. On the other hand, in multi-label classification, since each input can have multiple labels, the output is represented as a vector with dimensions equal to the total number of output labels. Thus, proper pre-processing of data is essential in achieving LIC. In the proposed approach, the target label of all three classification types is represented as a vector with dimension equal to the number of output labels. Each element in the vector signifies the belongingness of the input to the corresponding label.

**Online Training Phase.** During the training phase, the data samples and the target labels are provided as the input and the output weight values are estimated iteratively by online training. The proposed method is based on the online variant of ELM. The online training phase has two steps.

*Initial Block Step:* Let $N_0$ be the number of training samples in the initial block of data, the initial output weight values are calculated using equations

$$M_0 = (H_0^T H_0)^{-1} \quad (20)$$
\[ \beta_0 = M_0 H_0^T Y \]  

(21)

**Sequential Training Step:** Upon completion of the initial block step, the subsequent data samples arriving sequentially are processed in the sequential training step. The output weight is updated iteratively with sequentially arriving data blocks using the recursive least square technique [41, 51]. The sequential update of output weight is given by the equations

\[
M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}}
\]  

(22)

\[
\beta_{k+1} = \beta_k + M_{k+1} h_{k+1} (Y_{k+1}^T - h_{k+1}^T \beta_k)
\]  

(23)

By the end of the training phase, the values of \( \beta \) are estimated.

**Testing Phase.** In the testing phase, the target output of the input samples is predicted using the values of \( \beta \) estimated from the training phase and the input data samples. The raw output values of the network are evaluated using the relation \( Y = H \beta \). The raw output value obtained from the testing phase is then processed to address the three challenges of the label-independent classifier (LIC).

**Classification Phase.** In the classification phase, the raw output values \( Y \) obtained from the training phase is used to predict the classification type, number of associated target labels and identifying each of the target labels corresponding to each input sample.
Identifying the Classification Type: The classification type of binary, multi-class or multi-label is identified using the classification type (CT) value and dimension of output vector ‘l’. The CT value is evaluated using the equation.

\[ CT = | HS(Y) | \] (24)

where Y is the raw output vector and HS(x) is the heaviside function. Identification of classification type based on all possible valid combinations of CT and L is given in Table 5.1.

<table>
<thead>
<tr>
<th>CT = 1</th>
<th>L = 2</th>
<th>Binary Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT = 1</td>
<td>L &gt; 2</td>
<td>Multi-class Classification</td>
</tr>
<tr>
<td>CT &gt; 1</td>
<td>L &gt; 2</td>
<td>Multi-label Classification</td>
</tr>
</tbody>
</table>

Estimating the Number of Target Labels: Upon establishing the classification type, the number of target labels is then estimated as given in Table 5.2. For binary and multi-class classification, the number of target labels is one, since each input belongs to unique target labels. For multi-label classification, the CT value corresponds to the number of target labels associated with the input data sample.
Table 5.2: Estimation of Number of Target Labels

<table>
<thead>
<tr>
<th>Classification Type</th>
<th>Number of Target Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Binary Classification</em></td>
<td>1</td>
</tr>
<tr>
<td><em>Multi-class Classification</em></td>
<td>1</td>
</tr>
</tbody>
</table>
| *Multi-label Classification*| \[
\sum_{i=1}^{L} HS(Y_i) \]

**Algorithm 3: Proposed Label-independent (Universal) Classifier Algorithm**

1. Initialization of parameters

2. Formatting input to uniform representation

3. Initial block training

   Input: Initial N0 samples of data in the form \{ (x_i, y_i) \}

   Output: \( \beta_0 \)

   Evaluation:

   \[
   M_0 = (H_0^T H_0)^{-1}
   \]

   \[
   \beta_0 = M_0 H_0^T Y_0
   \]

4. Sequential training

   Input: Sequentially arriving data blocks in the form \{ (x_i, y_i) \}
Output: $\beta_k$

Evaluation:

$$M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}}$$

$$\beta_{k+1} = \beta_k + M_{k+1} h_{k+1} (Y^T_{k+1} - h_{k+1}^T \beta_k)$$

5. Evaluating raw output value $Y$

Input: Data sample $x_i$

Output: $Y$

Evaluation:

$$Y = H\beta$$

6. Evaluating CT value: $CT = |H(Y)|$

7. Identifying classification type based on CT and L

8. Evaluating number of associated target labels

9. Calculating the belongingness vector: $B = H(Y)$

10. Identifying the associated target labels

11. Evaluation of performance metrics corresponding to classification type

Identifying the Target Labels: The target labels are identified using the belongingness vector. The belongingness vector $B$ is given as,
where $Y$ is the raw output value and $HS(x)$ is Heaviside function. Each element of the vector $B$ denotes the belongingness of the input to the corresponding label. Thus, the label index of the non-zero entries of the $B$ gives the target labels associated with the input samples. Upon estimating the target labels, the performance metrics of the classifier are evaluated. Thus, the proposed technique is capable of classifying all three types of classification problems. The overview of the proposed approach is summarized.

5.2 Experimentation

The experimental design, dataset specifications and the comparison methods used to evaluate the proposed method are discussed in this section. Since the proposed method is capable of classifying all the classification types, the datasets from both single-label and multi-label classification problems are chosen for experimentation. In multi-label problems, different datasets have different degree of multi-labelness. Therefore, two metrics, Label cardinality and label density are used to quantitatively measure the degree of multi-labelness. These measures can also be used for single-label classification. Label cardinality gives the average number of labels corresponding to the input data. Therefore, for binary and multi-class classification, label cardinality will always be 1. Label density on the other hand also considers the number of labels in evaluation. Label cardinality and label density are very important metrics in the dataset specification for multi-label data. For example, label cardinality of 4.24 signifies that each of the input samples
corresponds to more than 4 labels on an average. Two datasets having same label cardinality, but different label density can significantly vary the performance of the classifier. The specifications of the dataset used for experimentation are given in the Table 5.3.

The proposed method is evaluated using 5 single-label (2 binary and 3 multi-class) datasets and 4 multi-label datasets. The datasets cover a wide range of feature dimension, number of labels, label density and label cardinality. The performance results of the proposed method on these datasets are compared with state-of-the-art techniques in the specific classification type. The performance of the proposed method on single-label classification datasets are compared with Support Vector Machine (SVM), kNN (k-Nearest Neighbor), MLP (Multi-layer Perceptron) and ELM (Extreme Learning Machine) based techniques. The results of the multi-label classification datasets are compared with the state-of-the-art methods based on SVM, kNN, DT (Decision Trees) and RF (Random Forest).
Table 5.3: Datasets Specifications

<table>
<thead>
<tr>
<th>Classification type</th>
<th>Dataset</th>
<th>Number of labels</th>
<th>Feature dimension</th>
<th>Number of Samples</th>
<th>Label Cardinality</th>
<th>Label Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-label</td>
<td>Diabetes</td>
<td>2</td>
<td>8</td>
<td>768</td>
<td>1.00</td>
<td>0.500</td>
</tr>
<tr>
<td></td>
<td>Ionosphere</td>
<td>2</td>
<td>34</td>
<td>351</td>
<td>1.00</td>
<td>0.500</td>
</tr>
<tr>
<td>Multi-class</td>
<td>Iris</td>
<td>3</td>
<td>4</td>
<td>150</td>
<td>1.00</td>
<td>0.333</td>
</tr>
<tr>
<td></td>
<td>Waveform</td>
<td>3</td>
<td>21</td>
<td>5000</td>
<td>1.00</td>
<td>0.333</td>
</tr>
<tr>
<td></td>
<td>Balance-scale</td>
<td>3</td>
<td>4</td>
<td>625</td>
<td>1.00</td>
<td>0.333</td>
</tr>
<tr>
<td>Multi-label</td>
<td>Scene</td>
<td>6</td>
<td>294</td>
<td>2407</td>
<td>1.07</td>
<td>0.178</td>
</tr>
<tr>
<td></td>
<td>Yeast</td>
<td>14</td>
<td>103</td>
<td>2417</td>
<td>4.24</td>
<td>0.303</td>
</tr>
<tr>
<td></td>
<td>Corel5k</td>
<td>374</td>
<td>499</td>
<td>5000</td>
<td>3.53</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>Enron</td>
<td>53</td>
<td>1001</td>
<td>1702</td>
<td>3.38</td>
<td>0.064</td>
</tr>
</tbody>
</table>

5.3 Results and Discussions

This section summarizes the experimental results of the proposed universal classifier on the datasets specified in table. Being an online method, the proposed algorithm can be used for streaming data applications.
5.3.1 Consistency

Consistency is one of the key virtues of any new technique developed. An algorithm that is inconsistent with results on different trials is unreliable. Cross-validation is one of the effective ways to evaluate the consistency of the method. A 10-fold cross validation is performed for each of the datasets. In single-label classification problems, since each of the sample belongs to only one output, the performance of the classifier can be evaluated using the percentage of accuracy. However, multi-label classification poses a unique problem of partial correctness of the results. Since each sample corresponds to multiple target labels and the number of associated target labels also varies, it results in obtaining partially correct results. Therefore, a different set of performance metrics is used to evaluate the performance of the proposed method for multi-label classification datasets.

Hamming loss is one of the key performance metric for multi-label classification. It is the quantitative measure of the number of times the sample-label pair is misclassified. Lower the hamming loss, better the performance of the classifier. Hamming loss is evaluated as the summation of misclassified sample-label pair averaged over the total number of samples and labels.

Different sets of performance metrics are used for single-label and multi-label classifiers. The performance of the binary and multi-class classifier is evaluated using percentage of accuracy and the performance of multi-label classifier is evaluated using hamming loss for the consistency evaluation. The results obtained are tabulated in Table 5.4. From the table, it can be seen that the
proposed LIC is highly consistent for all datasets from binary, multi-class and multi-label classification.

Table 5.4: Consistency

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy % (10-fcv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>78.2 ± 3.2</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>96.4 ± 2.6</td>
</tr>
<tr>
<td>Iris</td>
<td>99.2 ± 0.6</td>
</tr>
<tr>
<td>Waveform</td>
<td>85.3 ± 1.8</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>90.7 ± 3.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Hamming loss (10-fcv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene</td>
<td>0.096 ± 0.002</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.201 ± 0.001</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.009 ± 0.000</td>
</tr>
<tr>
<td>Enron</td>
<td>0.047 ± 0.001</td>
</tr>
</tbody>
</table>
5.3.2 Performance Comparison

There are no universal classifier available in the literature to perform direct comparison with the proposed method. Therefore, the performance of the proposed classifier is compared with the state-of-the-art techniques in each of the classification type. For single-label classification datasets, the performance of the proposed method is compared with similar binary and multi-class techniques based on SVM, kNN, MLP and ELM.

Table 5.5: Performance Comparison

<table>
<thead>
<tr>
<th>Single-label Classification</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM</td>
</tr>
<tr>
<td>Diabetes</td>
<td>77.5</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>94.9</td>
</tr>
<tr>
<td>Iris</td>
<td>98.7</td>
</tr>
<tr>
<td>Waveform</td>
<td>85.7</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>91.2</td>
</tr>
</tbody>
</table>
For multi-label classification problems, the performance of the proposed classifier is compared with SVM, kNN, DT and RF based techniques. The results are given in the Table 5.5. Presence of the possibility of partial correctness in multi-label classification results in the need for evaluation of other parameters such as accuracy and F1 measure for multi-label classification. The results are tabulated in Table 5.6.

<table>
<thead>
<tr>
<th>Dataset</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SVM</td>
<td>kNN</td>
<td>RF-PCT</td>
<td>RFML-C4.5</td>
<td>LIC</td>
</tr>
<tr>
<td>Scene</td>
<td>0.082</td>
<td>0.099</td>
<td>0.094</td>
<td>0.116</td>
<td>0.096</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.193</td>
<td>0.198</td>
<td>0.197</td>
<td>0.205</td>
<td>0.201</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.012</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
<td>0.009</td>
</tr>
<tr>
<td>Enron</td>
<td>0.048</td>
<td>0.051</td>
<td>0.046</td>
<td>0.047</td>
<td>0.047</td>
</tr>
</tbody>
</table>
Table 5.6: Multi-label Performance Metrics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene</td>
<td>0.615</td>
<td>0.634</td>
<td>0.642</td>
<td>0.638</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.498</td>
<td>0.697</td>
<td>0.582</td>
<td>0.634</td>
</tr>
<tr>
<td>Corel5k</td>
<td>0.062</td>
<td>0.179</td>
<td>0.061</td>
<td>0.091</td>
</tr>
<tr>
<td>Enron</td>
<td>0.408</td>
<td>0.645</td>
<td>0.464</td>
<td>0.540</td>
</tr>
</tbody>
</table>

From the comparison table it is evident that the proposed LIC performs uniformly well in datasets of all classification types.

5.3.3 Execution Speed

The execution speed of the proposed classifier is evaluated in terms of training time and testing time. Execution speed of the classifier plays a vital role for streaming data applications. In order to perform real-time streaming data classification, the execution speed of the classifier should be less than the arrival rate of the streaming data. Therefore, a high speed classifier is essential for real-time streaming data applications. The proposed LIC/universal classifier exploits the inherent high-speed nature of the ELM. The training time and the testing time of the proposed universal algorithm for each dataset is given in Table 5.7.
### Table 5.7: Execution Speed

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training Time (s)</th>
<th>Testing Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-label Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diabetes</td>
<td>0.005</td>
<td>0</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.007</td>
<td>0</td>
</tr>
<tr>
<td>Iris</td>
<td>0.002</td>
<td>0</td>
</tr>
<tr>
<td>Waveform</td>
<td>0.012</td>
<td>0.001</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>0.009</td>
<td>0</td>
</tr>
<tr>
<td><strong>Multi-label Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scene</td>
<td>2.546</td>
<td>0.053</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.134</td>
<td>0.021</td>
</tr>
<tr>
<td>Corel5k</td>
<td>5.521</td>
<td>0.079</td>
</tr>
<tr>
<td>Enron</td>
<td>0.652</td>
<td>0.043</td>
</tr>
</tbody>
</table>

From the table, it is evident that the proposed classifier is capable of performing classification of all types with high speed, thus facilitating its application for real-time streaming data.
5.3.4 Experimentation on Real-world Dataset

Apart from the benchmark datasets, the proposed online universal classifier is also experimented with real world datasets. The problem of android malware family classification is chosen for experimentation. Two open sourced android malware datasets (we name it AMD1 and AMD2) are chosen for experimentation. The datasets contain the android application apks (samples) and the family (class) of malware it belongs to. The AMD1 [104] dataset contains a total of 5560 samples and 174 classes. The AMD2 [105] dataset contains a total of 1200 samples and 49 classes. This is a typical multi-class classification problem. Two popular techniques for android malware detection (Drebin [104] and CSBD [106]) are used for performance comparison. Drebin technique extracts the features from the apk files and use SVM based classifier for malware detection. Similarly, CSBD technique uses an RF based classifier after feature extraction to detect malware. In experimentation, we use the features extracted by Drebin and CSBD for each of the AMD1 and AMD2 datasets and use the proposed universal classifier (LIC_{DREBIN}, LIC_{CSBD}) to identify the malware families. The experimentation results are given in Tables 5.8 and 5.9. Precision, Accuracy, Recall and F-measure are used as performance metrics for results comparison. From the table, it is clear that, the proposed universal classifier with features from both Drebin and CSBD, outperformed the existing state of the art techniques which uses SVM and RF based classifier. It is evident from the experiment that, the proposed online universal classifier is highly suitable for real-world datasets with streaming data applications.
Table 5.8: Performance Comparison on AMD1 Dataset

<table>
<thead>
<tr>
<th></th>
<th>$\text{SVM}^{\text{DREBIN}}$</th>
<th>$\text{RF}^{\text{CSBD}}$</th>
<th>$\text{LIC}^{\text{DREBIN}}$</th>
<th>$\text{LIC}^{\text{CSBD}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.931</td>
<td>0.953</td>
<td>0.944</td>
<td>0.959</td>
</tr>
<tr>
<td>Recall</td>
<td>0.872</td>
<td>0.879</td>
<td>0.896</td>
<td>0.884</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.901</td>
<td>0.914</td>
<td>0.919</td>
<td>0.920</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.894</td>
<td>0.911</td>
<td>0.902</td>
<td>0.918</td>
</tr>
</tbody>
</table>

Table 5.9: Performance Comparison on AMD2 Dataset

<table>
<thead>
<tr>
<th></th>
<th>$\text{SVM}^{\text{DREBIN}}$</th>
<th>$\text{RF}^{\text{CSBD}}$</th>
<th>$\text{LIC}^{\text{DREBIN}}$</th>
<th>$\text{LIC}^{\text{CSBD}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.974</td>
<td>0.981</td>
<td>0.977</td>
<td>0.990</td>
</tr>
<tr>
<td>Recall</td>
<td>0.962</td>
<td>0.984</td>
<td>0.966</td>
<td>0.989</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.968</td>
<td>0.982</td>
<td>0.971</td>
<td>0.989</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.956</td>
<td>0.972</td>
<td>0.961</td>
<td>0.983</td>
</tr>
</tbody>
</table>

5.4 Summary

A novel online label-independent/universal classifier based on extreme learning machine is developed. It is to be highlighted that there are no classifiers available in the literature that can classify binary, multi-class and multi-label classification. The proposed online universal classifier is experimented with nine
different datasets of different classification types and the results are compared with state-of-the-art techniques in each type of classification problem. The proposed classifier is evaluated in terms of consistency, performance and speed. The high speed nature of the proposed classifier makes it suitable for real-time streaming data applications.
Chapter 6

Progressive Learning – Multi-class Classification

In this chapter, a progressive learning technique for multi-class classification is developed. The proposed progressive learning algorithm is adapted from the natural learning process exhibited by the children. Peter Jarvis in his book [100] has described in detail the nature of the human learning process. As opposed to traditional machine learning algorithm’s training-testing cycle, human learning is a continuous process. The learning / training phase is never ending. Whenever human brain is stumbled upon with a new phenomenon, the learning resumes [100]. The key feature of human learning is that, the learning of new phenomenon does not affect the knowledge learnt. The new knowledge is learnt and is added along with existing knowledge. Though there are several online and sequential learning methods, the information of number of classes is fixed during
initialization. This restricts the possibility of learning newer classes on the run. Therefore, we propose a novel learning technique called progressive learning which can learn new classes/labels on the run.

6.1 Progressive Learning Technique:

The learning process in machine learning can be categorized into two broad categories: (i) batch learning and (ii) sequential learning (online learning). Batch learning algorithms require pre-collection of training data. The collected data set is then used for training the neural network. The network parameters are calculated and updated by processing all the training data together. There are several batch learning algorithms in the literature. Batch learning involves processing of the complete data set concurrently for updating the weights. This technique is limited to its applications as batch learning techniques are more time consuming and requires the complete data set prior to training. On the other hand, in online/sequential learning algorithms, the network parameters are updated as and when a new training data arrives. To overcome the shortcomings of the batch learning techniques, several sequential and/or online learning algorithms are developed. In many cases, sequential learning algorithms are preferred over batch learning algorithms as they do not require retraining whenever a new data sample is received.

The issue with existing machine learning techniques for classification is that, once they are trained to classify a specific number of classes, learning of new classes/labels is not possible. In order to learn new class of data it requires
retraining all the classes anew again. The existing techniques require a priori information on the number of classes that will be present in the training dataset. The information on the number of classes is required to be either specified directly or is identified by analyzing the complete training data set. Based on this parameter, the network model will be designed and only the parameters or the weights of the networks are updated depending on the sequentially arriving input data. This makes the existing techniques “static” with respect to the number of classes it can learn.

While the existing techniques are suited for applications with pre-known dataset, it might not be well suited for applications such as cognitive robotics or those involving real-time data where the nature of training data is unknown. For such real-world and real-time data, where the number of classes to be learnt is often unknown, the learning technique must be self-developing to meet the dynamic needs. To overcome this shortcoming, a novel learning paradigm is proposed, called the “progressive learning”.

Progressive learning is the next stage of advancement to the online learning methods. Existing online sequential techniques only learn to classify data among a fixed set of classes which are initialized during the initialization phase of the algorithm. They fail to dynamically adapt when introduced to new class/classes on the run. The progressive learning technique is independent of the number of class constraint and it can learn several new classes on the go by retaining the knowledge of previous classes. This is achieved by modifying the network structure by itself
upon encountering a new class and updating the network parameters in such a way that it learns the new class and retains the knowledge learnt thus far.

The existing online sequential learning methods do not require retraining when a “new data sample” is received. But it fails when a “new class of data” which is unknown to the existing knowledge is encountered. Progressive learning technique overcomes this shortcoming by allowing the network to learn multiple new classes’ alien to existing knowledge, encountered at any point of time.

In this chapter, we propose progressive learning technique based multi-class classifier. The consistency and the complexity of the proposed progressive learning classifier are analyzed. Several standard datasets are used to evaluate the performance of the developed technique. A comparative study shows that the developed technique is superior to existing techniques.

6.2 Proposed Algorithm

The key objective of the progressive learning technique is that it can dynamically learn new classes on the run. Suppose the network is initially trained to classify ‘m’ number of classes. Consider the network encounters ‘c’ number of new classes which are alien to the previously learnt class, the Progressive Learning Technique (PLT) will adapt automatically and starts to learn the new class by maintaining the knowledge of previously learnt classes.

The introduction of new class(es) to the network, results in changes in the dimension of the output vector and the output weight matrix. Also the newly
formed matrices with increased dimension should be evaluated in such a way that it still retains the knowledge learnt thus far and also facilitates the learning of the newly introduced class(es). The method of increasing the dimension of the matrix, the weight update and matrix recalibration methods of the proposed algorithm are significantly different from the class-incremental extreme learning machine [39]. The proposed algorithm can not only learn sequential introduction of single new class, but also simultaneous and sequential introduction of multiple new classes. The proposed algorithm is also independent of the time of introduction of the new class(es).

Consider there are N’ hidden layer neurons, and the training data is of the form \((x_i, t_i)\), the steps of the PLT algorithm are:

**INITIALIZATION PHASE**

**STEP 1:** The input weights and the hidden layer bias are assigned at random.

**STEP 2:** For the initial block of \(N_0\) samples of data, the hidden layer output matrix \(H_0\) is calculated.

\[
H_0 = [h_1, \ldots, h_{N'}]^T, \quad h_i = [g(w_1 x_i + b_1), \ldots, g(w_{N'} x_i + b_{N'})]^T, \quad i = 1, 2, \ldots, N_0.
\]  

**STEP 3:** From the value of \(H_0\), the initial values of \(M_0\) and \(\beta_0\) are estimated as

\[
M_0 = (H_0^T H_0)^{-1}
\]  

\[
(27)
\]
\[ \beta_0 = M_0 H_0^T T_0 \] (28)

**SEQUENTIAL LEARNING PHASE**

The subsequent data that arrives to the network can be trained either on one-by-one or chunk-by-chunk basis. Let ‘b’ be the chunk size. Unity value for b results in training the network on one-by-one basis.

When a new data sample/chunk of data is arrived, it can fall into either of the two categories.

i) Absence of new class of data

ii) Presence of new class / classes of data

If there are no new classes in the current set of data, the PLT is similar to OS-ELM and the usual process of calculating and updating the output weights is performed. The subsequent algorithm steps for the case of no new classes in current chunk of data are as follows.

**STEP 4:** The hidden layer output vector \( h_{k+1} \) is calculated.

**STEP 5:** The output weight is updated based on the RLS algorithm as,

\[
M_{k+1} = M_k - M_k h_{k+1}^T (I + h_{k+1} M_k h_{k+1}^T)^{-1} h_{k+1} M_k
\] (29)

\[
\beta_{k+1} = \beta_k + M_{k+1} h_{k+1}^T (t_{k+1} - h_{k+1} \beta_k)
\] (30)
If there is a new class(es) in the chunk of data arrived, a novel progressive learning technique is used to recalibrate the network to accommodate new class by retaining old knowledge.

The algorithm maintains the classes learnt thus far in a separate set. When a new data sample/block of data arrives, the data are analyzed for the class it belongs to. If the target class of new data block is equal to or a subset of existing classes, no new classification has been encountered. When the new data block’s target class set is not a subset of existing classes, it means that the system has encountered new classification and a special recalibrate routine is initiated.

In the recalibration routine, the number of new classes encountered is determined and class labels are identified. Let ‘c’ be the number of new classes encountered. Upon identifying the number of new classes introduced, the set containing the classes learnt thus far is updated accordingly.

The neural network is redesigned with the number of output neurons increased accordingly and the interconnections redone. The weights of the new network are determined from the current and the previous weights of the old network. The weight update is made such that the knowledge learnt by the old network is retained and the knowledge of new classes is included along with it.

Consider there are N’ hidden layer neurons in the network, m classes of data are currently learnt by the network and b be the chunk size of the sequential
learning. The introduction of ‘c’ new classes at any instant k+1, will modify the dimensions of the output weight matrix \( \beta \) from \( \beta_{N'Xm} \) to \( \beta_{N'Xm+c} \).

The output weight matrix \( \beta \) is of critical importance in ELM based networks. Since the input weights and the hidden layer bias are randomly assigned, the values in the \( \beta \) matrix control the number of classes learnt and the accuracy of each class. The algorithm steps are continued as follows.

**STEP 4:** The values of \( \beta_{N'Xm+c} \) are calculated based on the current values of \( \beta_{N'Xm} \), \( (h_k)_{hXN'} \) and \( (M_k)_{N'XN'} \).

The current \( \beta \) matrix is of the dimension \( (\beta_k)_{N'Xm} \) and ‘c’ new classes are introduced. Therefore, to accommodate the output weight matrix for the increased number of output layer neurons, the \( \beta \) matrix is transformed to \( \tilde{\beta}_k \) as given in equation.

\[
\tilde{\beta}_k = (\beta_k)_{N'Xm} I_{mXm+c}
\]

where \( I_{mXm+c} \) is a rectangular identity matrix of dimension m X m+c.

\[
\tilde{\beta}_{kN'Xm+c} = (\beta_k)_{N'Xm} \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \end{bmatrix}_{mXm+c}
\]

\[
\tilde{\beta}_{kN'Xm+c} = [(\beta_k)_{N'Xm} \quad O_{N'Xc}]_{N'Xm+c}
\]

where \( O_{N'Xc} \) is zero matrix.
Upon extending the weight matrix to accommodate the increased number of output neurons, the learning learnt thus far has to be incorporated in the newly upgraded weight matrix. Appending zero matrix is a trivial way to increase the dimensions. The matrix values have to be updated such that the network retains the knowledge of existing classes and can learn new classes as if they were available from the beginning of the training phase.

From equation 18, it can be seen that, the error difference between the target class \( t_{k+1} \) and the predicted class \( h_{k+1} \beta_k \) is scaled by a learning factor and is added to \( \beta_k \). Since ‘c’ new classes are introduced only at the \( k+1 \)th time instant, for the initial \( k \) data samples, the target class label value corresponding to the new class is -1. Therefore, the \( k \)-learning step update for the ‘c’ new classes \( (\Delta \beta_k)^{N\times C} \) can be written as,

\[
(\Delta \beta_k)^{N\times C} = (M_k)^{N\times N_t} (h_k^T)^{N\times b} [-1 \ \ldots \ -1]_{b\times c} \tag{34}
\]

\[
(\Delta \beta_k)^{N\times C} = -(M_k)^{N\times N_t} (h_k^T)^{N\times b} J_{b\times c} \tag{35}
\]

where \( J_{b\times c} \) is an all-ones matrix.

\[
J_{b\times c} = \begin{bmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 1 \end{bmatrix}_{b\times c} \tag{36}
\]

The \( k \)-learning step update for the new classes is then incorporated with the \( \tilde{\beta}_{k_{m\times m+c}} \) to provide the upgraded \( (\beta_k)^{N'\times (m+c)} \) matrix which is recalibrated to adapt learning ‘c’ new classes.
\[(\Delta \beta_k)_{N'Xm+c} = [O_{N'Xm} - (M_k)_{N'XN} (h_k^T)_{N'Xb} f_{bXc}] \]  

(37)

The recalibrated output weight matrix \((\beta_k)_{N'X(N'+c)}\) is calculated as,

\[(\beta_k)_{N'X(m+c)} = \overline{\beta}_{k_{N'Xm+c}} + (\Delta \beta_k)_{N'Xm+c} \]  

(38)

Upon simplification, \((\beta_k)_{N'X(N'+c)}\) can be expressed as,

\[(\beta_k)_{N'X(m+c)} = [ (\beta_k)_{N'Xm} \ (\Delta \beta_k)_{N'Xc} ] \]  

(39)

\((\beta_k)_{N'Xm}\) represents the knowledge previously learnt. The dimension of \(\beta\) is increased from \(m\) to \(m+c\). As opposed to populating the increased dimension with identity matrix values, the new entries \((\Delta \beta_k)_{N'Xc}\) are calculated in such a way that the newly introduced classes will appear to the neural network as if they are present from the beginning of the training procedure and the training data samples thus far does not belong to the newly introduced class.

The network is recalibrated such that the \((\Delta \beta_k)_{N'Xc}\) matrix represents the learning of the new class from the beginning of the training phase to the current data sample considering that none of the previous data samples belong to the newly introduced class. i.e. The \((\Delta \beta_k)_{N'Xc}\) is computed which is equivalent to the \(k\)-learning step equivalent of the ‘c’ new classes from the beginning of the training phase.

Therefore the updated \((\beta_k)_{N'X(m+c)}\) matrix represents the network with \((m+c)\) classes with ‘m’ previously existing classes and ‘c’ new classes.

\textit{STEP 5}: The hidden layer output vector \(h_{k+1}\) is calculated.
**STEP 6:** The output weight matrix of increased dimension to facilitate learning of new class is updated based on the RLS algorithm as,

\[
M_{k+1} = M_k - M_k h_{k+1}^T (I + h_{k+1} M_k h_{k+1}^T)^{-1} h_{k+1} M_k
\]  

\[
\beta_{k+1} = \beta_k + M_k h_{k+1}^T (t_{k+1} - h_{k+1} \beta_k)
\]

(40)  

(41)

Whenever a new class(es) are encountered, the training resume learning the new class/classes by retaining the existing knowledge. The algorithm also supports recalibration with multiple new classes introduced simultaneously and sequentially. Also, the new classes can be introduced at any instant of time and any number of times to the network.

The algorithm of the progressive learning technique (PLT) is summarized in flow chart given in Figure 6.1.
Figure 6.1: Flowchart of Algorithm 4 (Progressive Learning Technique for Multi-class Classification)
6.3 Experimentation

Proposed progressive learning algorithm exhibit “dynamic” learning of new class of data. Current multiclass classification algorithms fails to adapt when encountered with new class and hence the accuracy drops when introduced with one or more new classes. The proposed algorithm redesigns itself to adapt to new classifications and still retaining the knowledge learnt thus far.

The proposed progressive learning algorithm is tested with several real world and standard datasets. The standard datasets are in general uniformly distributed. But to test the performance of progressive learning effectively, it should be presented with conditions where new classes are introduced in a non–uniform manner at different time instants. Hence the standard datasets cannot be used directly to test the progressive learning algorithm efficiently. The datasets should be in such a way that only a subset of classes is available for training initially and new classes should be introduced at arbitrary time instances during the latter part of training. Thus, some of the standard datasets are modified and used for testing the proposed algorithm.

By default, classification problems involve two classes: 1. Presence of class and 2. Absence of class. These are the two trivial classes that are available in any of the classification problem. Since the minimum number of classes in a classification is two, learning of new classes is absent in bivariate datasets. For binary classification datasets, since there are only two classes and no new classes are introduced, the proposed algorithm performs similar to the existing online
sequential algorithm. The unique feature of progressive learning is clearly evident only in multiclass classification.

Thus the proposed algorithm is tested with multiclass classification datasets such as iris, balance scale, waveform and character datasets. The specifications of the datasets are shown in Table 6.1.

The proposed technique is experimented with both balanced and unbalanced datasets. Balanced dataset is one in which each of the class has equal or almost equal number of training data. Unbalanced dataset is a skewed dataset where a subset of classes has a high number of training samples and other classes have fewer training samples.

The proposed algorithm also works for introduction of multiple new classes. The number of classes can be increased from 2 to 3, and then from 3 to 4 and 4 to 5 and so on. For testing multiple new classes, the proposed method is tested with character recognition dataset which is described in the latter part of this section. The introduction of multiple classes both sequentially and simultaneously at multiple time instances are experimented and verified.
Table 6.1. Specifications of Multi-class Classification Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of classes</th>
<th>Number of features/attributes</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris dataset</td>
<td>3</td>
<td>4</td>
<td>Basic benchmark dataset</td>
</tr>
<tr>
<td>Balance scale dataset</td>
<td>3</td>
<td>4</td>
<td>Benchmark dataset for unbalanced data</td>
</tr>
<tr>
<td>Waveform dataset</td>
<td>3</td>
<td>21</td>
<td>Basic benchmark dataset</td>
</tr>
<tr>
<td>Wine dataset</td>
<td>3</td>
<td>13</td>
<td>Basic benchmark dataset</td>
</tr>
<tr>
<td>Satellite image dataset</td>
<td>6</td>
<td>36</td>
<td>Basic benchmark dataset</td>
</tr>
<tr>
<td>Digit dataset</td>
<td>10</td>
<td>64</td>
<td>Basic benchmark dataset</td>
</tr>
<tr>
<td>Character dataset 1</td>
<td>4</td>
<td>17</td>
<td>Dataset for sequential introduction of two new classes</td>
</tr>
<tr>
<td>Character dataset 2</td>
<td>5</td>
<td>17</td>
<td>Dataset for sequential introduction of three new classes</td>
</tr>
<tr>
<td>Character dataset 3</td>
<td>5</td>
<td>17</td>
<td>Dataset for simultaneous introduction of new classes</td>
</tr>
</tbody>
</table>
6.4 Results and Discussions

The functionality of the technique, consistency and complexity are the three key features to be tested for any new technique. The functional testing is used to validate that the proposed algorithm is functional and results in its expected behavior. The functionality of the technique is tested using iris, waveform and balance-scale datasets. The operational working of the concept of progressive learning in the proposed algorithm is tested in the functionality test. Consistency is another key feature that is essential for any new technique. The proposed algorithm should provide consistent results for multiple trials with minimal variance. Being an ELM based algorithm, the consistency of the proposed method across several trials of same dataset and also the consistency across 10-fold cross validation are tested. Complexity analysis is essential for a new technique. The number of operations performed and calculations involved in the proposed method is computed and is compared against the existing method. Also the performance of the proposed algorithm is evaluated by introducing new classes at different time instances (1. Very early during training, 2. In the middle of training and 3. Towards the end of training) are evaluated. Both sequential and simultaneous introduction of new classes are experimented and results are analyzed and discussed.

6.4.1 Functionality

The proposed technique is experimented with iris, waveform and balance scale datasets to verify the basic intended functionality of the technique. The iris dataset consists of three classes which are uniformly distributed over the 150 instances. To
facilitate testing of progressive learning, the dataset is redistributed such that first 50 samples consists of only two classes (sentosa, versicolor) and the third class (virginica) is introduced only after the 51st sample. This type of redistribution closely emulates the real time scenario of encountering a new class on the run. The ability of the proposed algorithm to recognize, adapt and learn the new class can be verified by this testing. The distribution details of the dataset used is given in Table 6.2.

Table 6.2. Specifications of Iris Dataset

<table>
<thead>
<tr>
<th>Data range</th>
<th>Number of classes</th>
<th>New class added</th>
<th>Point of introduction of new class</th>
<th>Class labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–50</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>Sentosa, Versicolor</td>
</tr>
<tr>
<td>50–150</td>
<td>3</td>
<td>1</td>
<td>51</td>
<td>Sentosa, Versicolor, Virginica</td>
</tr>
</tbody>
</table>
The Progressive Learning algorithm is tested with the specified iris dataset and the learning curve or the testing accuracy graph is plotted. The result obtained is shown in Figure 6.2. The testing accuracy is continuously calculated with the test data set for every new training data. It can be seen from the graph that until the sample index 50, the testing accuracy is only 66.6%. This implies that the system has learnt only two of the three classes thus far. When the third new class is introduced in the 51st sample, the system recognizes the introduction of a new class and recalibrates itself by sufficiently increasing the number of neurons in the output layer. The weight matrix is suitably increased in dimension and the weights are updated based on the special recalibration technique proposed. A new network structure and weight parameters are formed from the current network parameters.
and the data obtained from the new class. In the forthcoming iterations, the system then trains for recognition of the new class in addition to previously learnt classes and reaches a steady state testing accuracy. This process results in the sudden rise in the testing accuracy of the network, which then settles at a final testing accuracy value. This sudden increase in the testing accuracy is due to the fact that the network can now recognize the newly encountered class of data.

The same procedure is repeated for waveform and balance scale dataset. The dataset specifications of the waveform and balance scale dataset are shown in Table 6.3-6.4. The result obtained by the progressive learning method is shown in Figures 6.3-6.4 respectively.

Table 6.3. Specifications of Waveform Dataset

<table>
<thead>
<tr>
<th>Data range</th>
<th>Number of classes</th>
<th>New class added</th>
<th>Point of introduction of new class</th>
<th>Class labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–1500</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>Waveform 1, Waveform 2</td>
</tr>
<tr>
<td>1501–3000</td>
<td>3</td>
<td>1</td>
<td>1501</td>
<td>Waveform 1, Waveform 2, Waveform 3</td>
</tr>
</tbody>
</table>
Figure 6.3. Testing Accuracy in Waveform Dataset

Table 6.4 Specifications of Balance Scale Dataset

<table>
<thead>
<tr>
<th>Data range</th>
<th>Number of classes</th>
<th>New class added</th>
<th>Point of introduction of new class</th>
<th>Class labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–350</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>Left, Right</td>
</tr>
<tr>
<td>351–1100</td>
<td>3</td>
<td>1</td>
<td>351</td>
<td>Left, Right, Balanced</td>
</tr>
</tbody>
</table>
From the test results, the expected behavior of the progressive learning method is verified. The result shows that the algorithm is able to learn new classes dynamically on the run and the learning of new class does not significantly affect the accuracy of the classes previously learnt. The consistency and performance of the proposed method is evaluated using six benchmark datasets.

6.4.2 Consistency

Consistency is a critical characteristic to be tested for any new technique. The proposed technique is verified for its consistency in its results. Consistency is a
key virtue that any technique should exhibit. The learning technique which provides inconsistent results is not reliable for practical applications. Being an ELM based technique, the input weights and the hidden layer bias values are initialized at random. Hence multiple executions of the same dataset and same specification results in different results. Therefore, the same dataset with same specification is executed multiple times to determine the consistency across multiple executions. The consistency results of repeated multiple executions of the three datasets are shown in Table 6.5.

Table 6.5 Consistency: Across Multiple Trials (10 Trials)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Testing Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Dataset</td>
<td>99.4 ± 0.9660</td>
</tr>
<tr>
<td>Waveform Dataset</td>
<td>83.9 ± 1.2589</td>
</tr>
<tr>
<td>Balance Scale Dataset</td>
<td>91.6 ± 1.0557</td>
</tr>
<tr>
<td>Wine Dataset</td>
<td>97.9 ± 0.9285</td>
</tr>
<tr>
<td>Satellite Image Dataset</td>
<td>89.6 ± 1.1640</td>
</tr>
<tr>
<td>Digit Dataset</td>
<td>97.1 ± 0.7854</td>
</tr>
</tbody>
</table>

Cross validation is the most common method to evaluate the consistency of any given technique. The proposed algorithm is tested with each of the datasets for 5-fold cross validation (5-fcv) and 10-fold cross validation (10-fcv) and the
resulting testing accuracy is tabulated. TABLE 6 gives the consistency of the proposed algorithm for cross validation performance. It can be seen from the table that the proposed algorithm is consistently accurate in each of the attempts. The deviation of the testing accuracy is in order of about 1% from the mean value which is nominal. Thus, the results show that the proposed method gives consistent and reliable testing accuracy for both balanced and unbalanced datasets.

Table 6.6 Consistency: 5-Fold Cross Validation and 10-Fold Cross Validation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>5-fcv</th>
<th>10-fcv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Dataset</td>
<td>99 ± 1.0954</td>
<td>99.4 ± 1.0544</td>
</tr>
<tr>
<td>Waveform Dataset</td>
<td>84.1 ± 1.2589</td>
<td>83.6 ± 1.3658</td>
</tr>
<tr>
<td>Balance Scale Dataset</td>
<td>91.8 ± 1.0557</td>
<td>91.2 ± 1.3847</td>
</tr>
<tr>
<td>Wine Dataset</td>
<td>97.5 ± 1.5376</td>
<td>97.9 ± 1.4625</td>
</tr>
<tr>
<td>Satellite Image Dataset</td>
<td>89.4 ± 1.4618</td>
<td>89.8 ± 1.5537</td>
</tr>
<tr>
<td>Digit Dataset</td>
<td>97.2 ± 1.0441</td>
<td>96.9 ± 1.0683</td>
</tr>
</tbody>
</table>

6.4.3 Computational Reduction

The number of computations required for the proposed progressive learning technique is analyzed and compared with the existing OS-ELM method. Though learning of new classes dynamically on the run causes overhead to the
computations and seemingly increases the complexity of the technique, the actual computational complexity of the proposed technique is lesser than the OS-ELM method. The decrease in complexity is due to two reasons.

1. The overhead computations responsible for increasing the number of output neurons, creating new interconnections and recalibration of weights occur only during the samples when a new class is introduced. Thus, the recalibration routine is invoked only when there is a new class, henceforth causing minimal increase in the computation complexity. For example, when only one new class is introduced, the recalibration procedure is invoked only once.

2. The progressive learning method also provides another distinct advantage. Since the new classes are learnt dynamically, it results in lesser number of weight calculations when compared with other static online sequential training techniques like OS-ELM.

For example, in the iris dataset considered, the traditional algorithm needs to update the weight for all three output neurons for the entire 150 instances of the training set. But in the proposed method, there are only two output neurons till the occurrence of the third class. The third output neuron is introduced only during the recalibration stage triggered by the introduction of a new class. Thus, the number of weight calculations is effectively reduced. This effectively reduces the number of computations performed and thereby reducing the computational complexity. The reduction in number of weight calculations is shown in Table 6.7. The number
of computations in the OS-ELM is normalized to 100 and the computational complexity of OS-ELM and the progressive learning method are compared in Figure 6.5.

![Figure 6.5. Comparison of Computational Reduction](image)

**Figure 6.5. Comparison of Computational Reduction**
Table 6.7 Reduction in Number of Calculations by the Proposed Method

<table>
<thead>
<tr>
<th>Dataset</th>
<th>No. of weight calculations in OS-ELM (*nHidden)</th>
<th>Point of introduction of new class</th>
<th>No. of weight calculations in proposed method (*nHidden)</th>
<th>% of calculations saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris dataset</td>
<td>150 * 3</td>
<td>51</td>
<td>(50 * 2) + (100 * 3)</td>
<td>11.11 %</td>
</tr>
<tr>
<td>Waveform dataset</td>
<td>3000 * 3</td>
<td>1501</td>
<td>(1500 * 2) + (1500 * 3)</td>
<td>16.67 %</td>
</tr>
<tr>
<td>Balance scale dataset</td>
<td>1100 * 3</td>
<td>351</td>
<td>(350 * 2) + (750*3)</td>
<td>10.61 %</td>
</tr>
<tr>
<td>Wine dataset</td>
<td>120*3</td>
<td>71</td>
<td>(70 * 2) + (50 * 3)</td>
<td>19.44 %</td>
</tr>
<tr>
<td>Satellite image dataset</td>
<td>4500 * 6</td>
<td>3001</td>
<td>(3000 * 5) + (1500 * 6)</td>
<td>11.11%</td>
</tr>
<tr>
<td>Digit dataset</td>
<td>4000 * 10</td>
<td>3001</td>
<td>(3000 * 9) + (1000 * 10)</td>
<td>7.5 %</td>
</tr>
</tbody>
</table>

Though the new classes are learnt only from halfway through the datasets, the testing accuracy of the algorithm is nearly maintained or even improved when compared to algorithms with a static number of classes. The reason for the change in accuracy is due to the fact that new classes are learnt on the run after the learning of previous classes. If the previously learnt classes and the new class are fairly distinctive the learning accuracy will be improved. On other hand, in some cases due to the feature set of the learnt class and new class, the learning of the new class
will affect the existing knowledge but only to a little extent thereby marginally reducing the overall accuracy.

The testing accuracy of the proposed algorithm is compared with the existing OS-ELM and its variants such as voting based OS-ELM (VOS-ELM), enhanced OS-ELM (EOS-ELM), robust OS-ELM (ROS-ELM), robust bayesian ELM (RB-ELM) and generalized pruning ELM (GP-ELM) and is tabulated as shown in TABLE 8. It can be seen from the table that despite learning the new classes dynamically at a later stage of training, the testing accuracy is either improved or maintained nearly equal to the testing accuracy of the OS-ELM based methods. But the proposed method provides two key advantages over the existing methods.

1. Reduction in computational complexity.
2. Flexibility to learn new classes at any instant of time.

From the results obtained thus far, it is evident that the proposed progressive learning algorithm learns new class of data in a dynamic way.
Table 6.8 Comparison of Testing Accuracy (%)

<table>
<thead>
<tr>
<th></th>
<th>OS-ELM</th>
<th>VOS-ELM</th>
<th>EOS-ELM</th>
<th>ROS-ELM</th>
<th>RB-ELM</th>
<th>GP-ELM</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRIS dataset</td>
<td>98</td>
<td>99.2</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Waveform dataset</td>
<td>84.2</td>
<td>84.8</td>
<td>84.6</td>
<td>85.1</td>
<td>84.3</td>
<td>84.7</td>
<td>83.9</td>
</tr>
<tr>
<td>Balance scale dataset</td>
<td>90.7</td>
<td>91.1</td>
<td>90.8</td>
<td>91.4</td>
<td>90.9</td>
<td>91.3</td>
<td>91.6</td>
</tr>
<tr>
<td>Wine dataset</td>
<td>97.2</td>
<td>97.5</td>
<td>97.4</td>
<td>98.0</td>
<td>97.1</td>
<td>97.6</td>
<td>97.9</td>
</tr>
<tr>
<td>Satellite image dataset</td>
<td>88.9</td>
<td>89.2</td>
<td>89.0</td>
<td>89.1</td>
<td>89.5</td>
<td>89.8</td>
<td>89.6</td>
</tr>
<tr>
<td>Digit dataset</td>
<td>96.6</td>
<td>96.8</td>
<td>96.5</td>
<td>96.9</td>
<td>97.1</td>
<td>97.3</td>
<td>97.1</td>
</tr>
</tbody>
</table>

6.4.4 Introduction of New Class at Different Time Instants

The new class is introduced at different stages of the training period and its effect on learning rate is analyzed. In order to analyze the response, three different test cases are experimented and performance is measured. The new class is
introduced at three different time instances. 1. Very early during training, 2. In the middle of training, 3. Towards the end of training. The testing accuracy curve in each of the test case is plotted and the results are evaluated and compared. The point of introduction of a new class for each of the test case is tabulated and is given in Table 6.9. The performance of the proposed network for each of the test cases is given in Figure 6.6. It can be seen from the figure that, independent of the point of introduction of a new class to the system, the network is capable of learning the new class and the final steady state testing accuracy is the same across the test cases.

Table 6.9. Point of Introduction of New Class

<table>
<thead>
<tr>
<th>Test cases</th>
<th>Point of introduction of new class (Total number of samples = 150)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Early</td>
<td>6</td>
</tr>
<tr>
<td>In the Middle</td>
<td>71</td>
</tr>
<tr>
<td>Towards the End</td>
<td>131</td>
</tr>
</tbody>
</table>
Figure 6.6: Introduction of New Class in Training (a) Very Early, (b) In the Middle and (c) Towards the End
6.4.5 Multiple New Classes

The performance of the proposed technique when introduced with multiple new classes both sequentially and simultaneously is discussed in this section. Learning of multiple new classes by the proposed algorithm is tested by using the Character recognition dataset. Several combinations of tests are made such as

1. Sequential introduction of 2 new classes (4 classes)
2. Sequential introduction of 3 new classes (5 classes)
3. Simultaneous introduction of 2 new classes along with one new class sequentially (5 classes)

The performance of the proposed algorithm on each of the test case is observed.

*Sequential Introduction of 2 new classes*

Character dataset with 4 classes (A, B, C and D) is used to test the sequential introduction of two new classes in the proposed algorithm. The dataset is redistributed to meet the testing requirements for progressive learning. The specifications of the dataset are given in Table 6.10.
Table 6.10 Specifications of Character Dataset for 2 New Classes

<table>
<thead>
<tr>
<th>Data range</th>
<th>Number of classes</th>
<th>New class added</th>
<th>Class labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 800</td>
<td>2</td>
<td>-</td>
<td>A and B</td>
</tr>
<tr>
<td>801 – 1600</td>
<td>3</td>
<td>C</td>
<td>A, B and C</td>
</tr>
<tr>
<td>1601 – 3096</td>
<td>4</td>
<td>D</td>
<td>A, B, C and D</td>
</tr>
</tbody>
</table>

Initially the network is sequentially trained with only two classes A and B up to 800 samples. A new class ‘C’ is introduced to the training data in the 801st sample and a fourth class ‘D’ is introduced as 1601st sample. The proposed algorithm identifies both the new classes and recalibrates itself each time and continues learning. This results in two sudden rise in the learning curve of the network. The first rise corresponding occurring at 801st sample corresponds to the learning of class ‘C’ and the second rise occurring at 1601st sample corresponds to learning of class ‘D’. The learning curve graph is shown in Fig. 7.
Figure 6.7. Sequential Learning of Two New Classes

*Sequential introduction of 3 new classes*

Character dataset with 5 classes (A, B, C, D and E) is used for testing sequential introduction of 3 new classes. The network is initially trained to recognize only two classes. Three new classes (C, D and E) are introduced one after another after the initial training of two classes. The specifications of the dataset are shown in Table 6.11.
Table 6.11 Specifications of Character Dataset for 3 New Classes

<table>
<thead>
<tr>
<th>Data range</th>
<th>Number of classes</th>
<th>New class added</th>
<th>Class labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 800</td>
<td>2</td>
<td>-</td>
<td>A and B</td>
</tr>
<tr>
<td>801 – 1600</td>
<td>3</td>
<td>C</td>
<td>A, B and C</td>
</tr>
<tr>
<td>1601 – 2000</td>
<td>4</td>
<td>D</td>
<td>A, B, C and D</td>
</tr>
<tr>
<td>2001 – 3850</td>
<td>5</td>
<td>E</td>
<td>A, B, C, D and E</td>
</tr>
</tbody>
</table>

Each of the new classes is introduced sequentially at later time instants and the algorithm adapts to new class each time and also maintains the testing accuracy at the same level. The testing accuracy curve is shown in Figure 6.8.

To verify that learning of each new class is independent of previously learnt classes, the overall testing accuracy is broken down into individual testing accuracy of each of the classes and is shown in Fig. 9. It can be seen that, the testing accuracy of each of the classes remains over 90%. Also, whenever a new class is introduced, a new learning curve is formed which contributes towards the overall accuracy along with the existing classes.
The network is initially trained with two classes A and B. Third class C is introduced after 800 samples and the learning curve of the class C is shown in black line. Another new class ‘D’ who’s testing accuracy as shown in red is introduced after the 1600 sample. A fifth class, ‘E’ is introduced in the 2001\textsuperscript{st} sample and its learning curve is shown in light blue.

From the graph it can be seen that each class introduced is learnt anew without affecting much the existing knowledge. The learning accuracy of each of the classes is collectively responsible for the overall accuracy of the network. Further, it can be seen that the testing accuracy of each of the classes is over 90% and the overall accuracy of 94% is achieved.

Figure 6.8. Sequential Learning of Three New Classes
The testing accuracy obtained by introducing one, two and three new classes is summarized in Table 6.12.

From the table it can be observed that learning of multiple new classes does not affect the testing accuracy of previously learnt class. Hence this method can be used to learn a large number of multiple new classes in a progressive manner without affecting the testing accuracy of previously learnt classes.
Table 6.12 Summary of Testing Accuracy for Sequential Introduction of Multiple New Classes

<table>
<thead>
<tr>
<th>Number of classes introduced sequentially</th>
<th>Testing Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two base class + One new class</td>
<td>93.8 %</td>
</tr>
<tr>
<td>Two base class + Two new classes</td>
<td>93.7 %</td>
</tr>
<tr>
<td>Two base class + Three new classes</td>
<td>94</td>
</tr>
</tbody>
</table>

Simultaneous introduction of new classes

To verify that the proposed algorithm performs effectively when multiple classes are introduced simultaneously (introduced in the same block), character dataset with specifications as shown in Table 6.13 is used. Here, the two classes C and D are introduced together and the new class E at a later stage. The testing accuracy is shown in Figure 6.10.

Table 6.13 Specifications of Character Dataset for Simultaneous New Classes

<table>
<thead>
<tr>
<th>Data range</th>
<th>Number of classes</th>
<th>New class added</th>
<th>Class labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 800</td>
<td>2</td>
<td>-</td>
<td>A and B</td>
</tr>
<tr>
<td>801 – 2000</td>
<td>4</td>
<td>C,D</td>
<td>A, B, C and D</td>
</tr>
<tr>
<td>2001 – 3850</td>
<td>5</td>
<td>E</td>
<td>A,B,C,D and E</td>
</tr>
</tbody>
</table>
The first rise observed at the sample instant of 800 in the testing accuracy curve corresponds to the introduction of two new classes (characters C and D). The algorithm identifies both the new classes and recalibrates to facilitate multiple class addition. The second rise in the curve corresponds to the introduction of the third class (character E). In order to show that the previous knowledge is retained and new knowledge is added along with the existing, the testing accuracy is split up for each of the five alphabets and is shown in Fig. 11. It can be seen that, two new learning curves corresponding to each new class C and D is introduced in the 800th sample. Both of the newly introduced classes are learnt simultaneously along with the existing classes A and B. The learning curve at 1600th sample index corresponds to the introduction of class E. Also, from the graph it is clear that the learning of additional classes does not significantly affect the testing accuracy of the classes previously learnt. Thus, enabling the proposed algorithm to learn multiple new classes both sequentially and simultaneously in a progressive manner.

Figure 6.10. Testing Accuracy for Simultaneous New Classes
The proposed algorithm introduces new neurons in the output layer and recalibrates the network by itself to facilitate learning of new classes. Since only the output layer neurons are increased and the number of hidden layer neurons is the same, the learning of new classes that can be progressively learnt is limited by the number of classes that can be learnt by the given number of hidden layer neurons. Further, the proposed algorithm can be extended such that both the output neurons and hidden layer neurons are increased such that any number of new classes can be learnt progressively.
6.4.6 Experimentation on Real-world Dataset

In several real-world applications, the number of classes to be identified is often unknown and it grows with time. Traditional online learning techniques need to be retrained whenever a new class is being introduced. Therefore, progressive learning technique, which can learn new classes continuously is a most suitable alternate such problems. An example of one such continuously growing number of classes in real-world is android malware family classification. In android malware family classification, new families of malware are being introduced in real-time. As mentioned in Chapter 5, two open sourced android malware datasets (AMD1 [104] and AMD2 [105]) are chosen for experimentation. The evolution of malware families over time is analyzed and is shown in figure 6.12. The x axis in the figure gives the timeline in YYYYMM format and the left y axis gives the number of newly introduced malware family for that particular month and the right y axis gives the cumulative number of malware families so far. It is clearly evident that, new families of malware are being introduced continuously in the market. Therefore, it is highly required for a classifier to continuously grow and adapt to those new malware families to perform classification. The specifications of the two datasets are discussed earlier in Chapter 5. We use two popular android malware detection techniques (Drebin [104] and CSBD [106]) for performance evaluation. As foreshadowed, Drebin technique uses a SVM based classifier (SVM\textsuperscript{DREBIN}) and CSBD uses an RF based classifier (RF\textsuperscript{CSBD}) for malware detection. We use the features extracted by Drebin and CSBD and replace the classifier with the proposed progressive learning multi-class classifier (PL-MCC\textsuperscript{DREBIN} and PL-MCC\textsuperscript{CSBD}) and
the results are obtained. The proposed PL-MCC adapts and remodels itself whenever new families of malware are introduced thus enabling the classifier to learn the new classes. The performance metrics used for comparison are: Precision, Recall, F-measure and Accuracy. The results are summarized in Tables 6.14 and 6.15. From the tables, it is clear that, the progressive learning technique clearly outperforms the existing state-of-the-art classifiers.

Figure 6.12: Evolution of Malware Families
### Table 6.14: Experimentation Results for AMD1 dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>SVM</th>
<th>RF</th>
<th>PL-MCC</th>
<th>PL-MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>0.931</td>
<td>0.953</td>
<td>0.957</td>
<td>0.970</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>0.872</td>
<td>0.879</td>
<td>0.898</td>
<td>0.901</td>
</tr>
<tr>
<td><strong>F-measure</strong></td>
<td>0.901</td>
<td>0.914</td>
<td>0.926</td>
<td>0.934</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>0.894</td>
<td>0.911</td>
<td>0.915</td>
<td>0.924</td>
</tr>
</tbody>
</table>

### Table 6.15: Experimentation Results for AMD2 Dataset

<table>
<thead>
<tr>
<th>Method</th>
<th>SVM</th>
<th>RF</th>
<th>PL-MCC</th>
<th>PL-MCC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precision</strong></td>
<td>0.974</td>
<td>0.981</td>
<td>0.984</td>
<td>0.992</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>0.962</td>
<td>0.984</td>
<td>0.971</td>
<td>0.991</td>
</tr>
<tr>
<td><strong>F-measure</strong></td>
<td>0.968</td>
<td>0.982</td>
<td>0.977</td>
<td>0.991</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>0.956</td>
<td>0.972</td>
<td>0.969</td>
<td>0.987</td>
</tr>
</tbody>
</table>

### 6.5 Summary

In this chapter, a novel learning technique of progressive learning for multi-class classification is developed. Progressive learning enables the network to learn multiple new classes dynamically on the run. The new classes can be learnt in both sequential and simultaneous manner. Hence this technique is much suited for applications where the number of classes to be learned is unknown. Progressive
learning enables the network to recalibrate and adapt when encountered with a new class of data. The proposed progressive learning technique will perform effectively in applications such as cognitive robotics where the system is trained by real time experienced based data.
Chapter 7

Progressive Learning – Multi-label Classification

In this chapter, the progressive learning technique discussed in Chapter 6 is integrated to online multi-label classifier discussed in Chapter 4. As discussed earlier, the key objective of the progressive learning technique is to dynamically learn new classes on the run. The resulting progressive learning technique based online classifier can learn new labels while retaining the knowledge of previous labels for multi-label problems. Whenever new labels are being introduced, new output neurons are automatically added and the neural network connections and parameters are automatically re-structured. The automatic re-structuring of the neural network and update of neural network parameters are made in such a way to facilitate the learning of the new labels while retaining the knowledge previously learnt. It is to be highlighted that there are no similar algorithms available in the
literature for multi-label classification. The developed classifier enables learning of multiple new classes being introduced at multiple time instances. The developed technique is evaluated with different multi-label datasets for consistency, performance and speed. Though multiple new labels are introduced on the go, the performance of the proposed technique is superior to several state-of-the-art online learning techniques.

### 7.1 Proposed Algorithm

Progressive learning technique for multi-label classification is an extension of the previous works on online multi-label classifier and progressive learning technique for multi-class classification. In multi-label classification, each of the input samples is associated with a subset of target labels. Therefore, it is essential to determine both the number of labels and the identity of labels corresponding to an input sample. The various steps of the proposed algorithm are:

*Initialization of Parameters*

Fundamental parameters such as the number of hidden layer neurons and the activation function are initialized. Sigmoidal activation function is used for the experimentation. The problem of overfitting is tackled by using the early stopping technique. In early stopping technique, the point at which the training accuracy increases at the expense of generalization error is identified and further training is stopped. The number of hidden neurons are selected depending upon the nature and complexity of the dataset while preventing the overfitting of data.
Processing of Inputs

As discussed earlier, in traditional single-label problems, the target class will be a single-label associated with the input sample. But, in the multi-label case, each input sample can be associated with more than one class labels. Hence, each of the input samples will have the associated output label as an m-tuple with 0 or 1 representing the belongingness to each of the labels in the label space L. This is a key difference between the inputs available for single-label and multi-label problems. As opposed to single-label classification with a single target label, the multi-label problem has a target label set which is a subset of label space L. The label set denoting the belongingness for each of the labels is converted from unipolar representation to bipolar representation.

ELM Training

The processed input is then supplied to the online sequential variant of ELM technique. Let H be the hidden layer output matrix, β be the output weights and Y be the target label, the ELM can be represented in a compact form as $H\beta = Y$ where $Y \subseteq L$, $L = \{\zeta_1, \zeta_2, \ldots, \zeta_m\}$. During the training phase, Let $N_0$ be the number of samples in the initial block of data that is provided to the network. The initial output weight $\beta_0$ is calculated from equation 9 and 10.

$$\beta = H^*Y$$ and $$H^* = (H^TH)^{-1}H^T,$$
Consider $M_0 = (H_0^TH_0)^{-1}$, therefore, $\beta_0 = M_0H_0^TY_0$.

The subsequent data that arrives to the network can be trained either on one-by-one or chunk-by-chunk basis. Let ‘b’ be the chunk size. Unity value for b results in training the network on one-by-one basis.

When a new data sample/chunk of data is arrived, it can fall into either of the two categories.

i. Absence of new class of data

ii. Presence of new class / classes of data

If there are no new labels in the current set of data, the PLT for multi-label classification is similar to OSML-ELM and the usual process of calculating and updating the output weights is performed. The subsequent algorithm steps for the case of no new classes in current chunk of data are as follows.

The hidden layer output vector $h_{k+1}$ is calculated.

The output weight is updated based on the RLS algorithm as discussed in earlier chapter. If there are new labels in the chunk of data arrived, the progressive learning technique is used to recalibrate the network to accommodate new labels by retaining the previously learnt knowledge.

Similar to progressive learning for multi-class classification, whenever the network encounters new labels, the special recalibrate routine is initiated. In the recalibration routine, the number of new labels introduced is determined. Let ‘c’ be
the number of new labels being introduced. The neural network is redesigned with the number of output neurons increased accordingly and the interconnections redone. The weights of the new network are determined from the current and the previous weights of the old network.

Consider there are N’ hidden layer neurons in the network, m labels of data are currently learnt by the network and b be the chunk size of the sequential learning. The introduction of ‘c’ new classes at any instant k+1, will modify the dimensions of the output weight matrix $\beta$ from $\beta^{N'Xm}$ to $\beta^{N'Xm+c}$.

The output weight matrix $\beta$ is of critical importance in ELM based networks. Since the input weights and the hidden layer bias are randomly assigned, the values in the $\beta$ matrix control the number of classes learnt and the accuracy of each class. The ELM Training continues as follows.

The values of $\beta^{N'Xm+c}$ are calculated based on the current values of $\beta^{N'Xm}$, $(h_k)^{bXN'}$ and $(M_k)^{N'XN'}$.

The current $\beta$ matrix is of the dimension $(\beta_k)^{N'Xm}$ and ‘c’ new labels are introduced. Therefore, to accommodate the output weight matrix for the increased number of output layer neurons, the $\beta$ matrix is transformed to $\tilde{\beta}_k$ as given in equation.

$$\tilde{\beta}_k = (\beta_k)^{N'Xm} I_{mXm+c} \quad (42)$$

where $I_{mXm+c}$ is a rectangular identity matrix of dimension m X m+c.
\[
\bar{\beta}_{N'Xm+c} = (\beta_k)_{N'Xm} \begin{bmatrix}
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
0 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0
\end{bmatrix}_{mXm+c}
\] (43)

\[
\bar{\beta}_{N'Xm+c} = [(\beta_k)_{N'Xm} \quad O_{N'Xc}]_{N'Xm+c}
\] (44)

where \(O_{N'Xc}\) is zero matrix.

From equation 18, it can be seen that, the error difference between the target class \(t_{k+1}\) and the predicted class \(h_{k+1}\beta_k\) is scaled by a learning factor and is added to \(\beta_k\). Since ‘c’ new labels are introduced only at the k+1th time instant, for the initial k data samples, the target label value corresponding to the new labels are -1. Therefore, the k-learning step update for the ‘c’ new labels \((\Delta \beta_k)_{N'Xc}\) can be written as,

\[
(\Delta \beta_k)_{N'Xc} = (M_k)_{N'XN'} (h_k^T)_{N'Xb} \begin{bmatrix}
-1 & \ldots & -1 \\
\vdots & \ddots & \vdots \\
-1 & \ldots & -1
\end{bmatrix}_{bXc}
\] (45)

\[
(\Delta \beta_k)_{N'Xc} = -(M_k)_{N'XN'} (h_k^T)_{N'Xb} J_{bXc}
\] (46)

where \(J_{bXc}\) is an all-ones matrix.

\[
J_{bXc} = \begin{bmatrix}
1 & \ldots & 1 \\
\vdots & \ddots & \vdots \\
1 & \ldots & 1
\end{bmatrix}_{bXc}
\] (47)

The k-learning step update for the new labels is then incorporated with the \(\bar{\beta}_{kmXm+c}\) to provide the upgraded \((\beta_k)_{N'X(m+c)}\) matrix which is recalibrated to adapt learning ‘c’ new labels.

\[
(\Delta \bar{\beta}_k)_{N'Xm+c} = [O_{N'Xm} \quad -(M_k)_{N'XN'} (h_k^T)_{N'Xb} J_{bXc}]
\] (48)
The recalibrated output weight matrix \((\beta_k)_{N'X(N'+c)}\) is calculated as,

\[
(\beta_k)_{N'X(m+c)} = \overline{\beta}_{kN'Xm+c} + (\Delta \bar{\beta}_k)_{N'Xm+c}
\]

Upon simplification, \((\beta_k)_{N'X(N'+c)}\) can be expressed as,

\[
(\beta_k)_{N'X(m+c)} = \begin{bmatrix} (\beta_k)_{N'Xm} & (\Delta \beta_k)_{N'Xc} \end{bmatrix}
\]

\((\beta_k)_{N'Xm}\) represents the knowledge previously learnt. The dimension of \(\beta\) is increased from \(m\) to \(m+c\). As opposed to populating the increased dimension with identity matrix values, the new entries \((\Delta \beta_k)_{N'Xc}\) are calculated in such a way that the newly introduced labels will appear to the neural network as if they are present from the beginning of the training procedure and the training data samples thus far does not belong to the newly introduced labels.

The network is recalibrated such that the \((\Delta \beta_k)_{N'Xc}\) matrix represents the learning of the new labels from the beginning of the training phase to the current data sample considering that none of the previous data samples belong to the newly introduced class. i.e. The \((\Delta \beta_k)_{N'Xc}\) is computed which is equivalent to the k-learning step equivalent of the ‘c’ new labels from the beginning of the training phase.

Therefore the updated \((\beta_k)_{N'X(m+c)}\) matrix represents the network with \((m+c)\) classes with ‘m’ previously existing classes and ‘c’ new labels.

The hidden layer output vector \(h_{k+1}\) is then calculated.
The output weight matrix of increased dimension to facilitate learning of new labels is updated based on the RLS algorithm as discussed earlier.

**ELM Testing**

In the testing phase, the test data sample is evaluated using the values of $\beta$ obtained during the training phase. The input data that can be a combination of Boolean, discrete and continuous data type is given to the network. The network then computes $Y = H\beta$. The predicted output $Y$ obtained is a set of real numbers of dimension equal to the number of labels.

**Post-processing and Multi-label Identification**

The prime step in extending the progressive learning technique for multi-label problems is the post-processing and thresholding. As discussed earlier, in binary and multi-class classification, each of the input sample belongs to only one target label and, therefore, can be identified as the index of the maximum value in the predicted output. On contrary, in multi-label classification the number of labels each sample belongs to is not constant. Each input sample may belong to one or more than one of the target labels. Therefore, the classifier must predict both the number of labels and each of the corresponding labels for the input data sample. The number of labels corresponding to a data sample is completely unknown. Hence, in the proposed method, a thresholding-based label association is proposed. The threshold value is selected during the training phase such that it maximizes the separation between the family of labels the input belongs to and the family of
labels the input does not belong to, based on the raw output values \( Y \). Setting up of the threshold value is of prime importance as it directly affects the performance of the classifier. The \( L \) dimensioned raw-predicted output is compared with a unique threshold value. The index values of the predicted output \( Y \) which are greater than the fixed threshold represent the belongingness of the input sample to the corresponding class.

Setting the threshold value is of critical importance. The threshold value is selected such that it maximizes the difference between the category of labels to which the sample belongs to and the category of labels to which the sample does not belong to with respect to the raw output values \( Y \) obtained during the training phase. The distribution of the raw output values of \( Y \) for categories of labels that the input sample belongs to \( (Y_A) \) and the categories of labels the input sample does not belong to \( (Y_B) \) are identified. Based on the distribution of \( Y_A \) and \( Y_B \), a threshold value is identified using the formula,

\[
\text{Threshold value} = \frac{\min(Y_A) + \max(Y_B)}{2}
\]  

As a trivial case, the threshold can be set as zero. In which case, the raw predicted output values will be passed as arguments to a bipolar step function. The threshold value is compared with the raw output values of \( Y \) estimated by the classifier, and the number of target labels corresponding to the data sample is identified. Then, based on the threshold value, the subset of labels that corresponds to the input data sample is recognized. The threshold value is determined by analyzing the distribution of the raw predicted output values during the training.
phase. From the distribution, a particular value is chosen that maximizes the separation between the two categories of the labels. The proposed method belongs to the category of algorithm adaptation method, where the base algorithm is adapted to perform multi-label classification problems. It is to be highlighted that there are no ELM-based online multi-label classifiers in the literature thus far. The proposed method is the first to adapt the ELM for online multi-label problems and make extensive experimentation, results comparison and analysis with the state-of-the-art techniques. The overview of the proposed algorithm is given below.

Algorithm 5: Proposed Progressive Multi-label Classification Algorithm

13. The parameters of the network are initialized

14. The raw input data is processed for classification

15. ELM Training – Initial phase

Processing of initial block of data

\[
M_0 = (H_0^TH_0)^{-1}
\]

\[
\beta_0 = M_0H_0^TY_0
\]

16. ELM Training – Sequential phase

Case 1: No new labels are introduced:

\[
M_{k+1} = M_k - \frac{M_kh_{k+1}h_{k+1}^TM_k}{1 + h_{k+1}^TM_kh_{k+1}}
\]

\[
\beta_{k+1} = \beta_k + M_{k+1}h_{k+1}(Y_{k+1}^T - h_{k+1}^T\beta_k)
\]
Case 2: ‘c’ new labels are introduced:

\[
(\beta_k)_{N' \times (m+c)} = \begin{bmatrix} (\beta_k)_{N' \times m} & (\Delta \beta_k)_{N' \times c} \end{bmatrix}
\]

\[
(\Delta \beta_k)_{N' \times c} = (M_k)_{N' \times N} (h_k^T)_{N' \times b} \begin{bmatrix} -1 & \cdots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \cdots & -1 \end{bmatrix}_{b \times c}
\]

\[
M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}}
\]

\[
\beta_{k+1} = \beta_k + M_{k+1} h_{k+1} (Y_{k+1}^T - h_{k+1}^T \beta_k)
\]

17. ELM Testing

Estimation of raw output values using \( Y = H\beta \)

18. Post-processing and Multi-label identification

Applying the threshold value based on separation between two categories of labels (\( Y_A \) and \( Y_B \)). Threshold value = \( (\text{min}(Y_A) + \text{max}(Y_B))/2 \)

Identifying the number of labels corresponding to input data sample

Identifying the target class labels for the input data sample

7.2 Experimentation

As discussed in Chapters 3 and 4, Multi-label datasets have a unique property called the degree of multi-labelness. Not all datasets are equally multi-
labelled. Therefore, we use LC and LD to quantitatively measure the degree of multilabelness for each dataset. The proposed method is experimented with four benchmark datasets comprising of different application areas such as multimedia, text, and biology. The specifications of the datasets are given in table. Since, the datasets are discussed in detail in the previous chapters, they are not restated. Multiple new labels are introduced in each datasets at multiple time instances.

Table 7.1. Dataset Specifications

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Domain</th>
<th>No. of Features</th>
<th>No. of Samples</th>
<th>No. of Labels</th>
<th>Label Introduction Pattern</th>
<th>LC</th>
<th>LD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>Biology</td>
<td>103</td>
<td>2417</td>
<td>14</td>
<td>12 + 2</td>
<td>4.24</td>
<td>0.303</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10 + 1 + 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scene</td>
<td>Multimedia</td>
<td>294</td>
<td>2407</td>
<td>6</td>
<td>5 + 1</td>
<td>1.07</td>
<td>0.178</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 + 1 + 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corel5k</td>
<td>Multimedia</td>
<td>499</td>
<td>5000</td>
<td>374</td>
<td>373 + 1</td>
<td>3.52</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>368 + 2 + 1 + 1 + 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medical</td>
<td>Text</td>
<td>1449</td>
<td>978</td>
<td>45</td>
<td>44 + 1</td>
<td>1.25</td>
<td>0.027</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>39 + 2 + 2 + 1 + 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>38 + 3 + 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.3 Results and Discussion

The developed technique is evaluated for functionality, consistency, performance metrics and speed. The functionality check is performed in order to verify the progressive learning feature of the technique. The hamming loss metric is continuously monitored to evaluate the functionality of the technique. Any newly developed technique should provide consistent results across multiple trials. Therefore 5 fold and 10 fold cross validation is used to determine the consistency of the proposed method. Also, various performance metrics and the training time and testing time of the developed technique are evaluated and analysed.

7.3.1 Functionality

The proposed technique is experimented with scene, corel5k and medical datasets to verify the progressive learning functionality. The dataset is redistributed so that one or multiple new labels are introduced at multiple time instances. Hamming loss is a key performance metric to evaluate the testing accuracy of multi-label classification. It gives the fraction of wrong labels to the right labels. Therefore, by continuously monitoring the hamming loss value, the functionality of the progressive learning technique can be verified. Whenever a new label is introduced to the network, learning resumes and the newly introduced labels are learnt by retaining the knowledge of previously learnt labels. This results in a significant “dip” in the hamming loss value.
The proposed technique is first experimented with scene dataset. The dataset has a total of 6 labels. The samples in the dataset are redistributed such that new labels are introduced at later stage. The dataset with the label introduction pattern (5 + 1) is used to evaluate the progressive learning functionality of the proposed technique. The hamming loss measure is monitored continuously and is plotted. The individual hamming loss of the newly introduced class is evaluated separately and plotted overlapping to the overall hamming loss as shown in the Figure 7.1.

![Figure 7.1: Hamming Loss Plot for Scene (5+1) Dataset](image)

From the figure it can be seen that, a significant dip occurs in the hamming loss metric whenever a new label is introduced to the network. This is the direct consequence of the network restructuring itself automatically and learning the newly introduced label thus verifying the progressive learning feature of the proposed method. The same procedure is repeated for medical dataset. Medical
dataset has 45 labels and the dataset is redistributed to have the label introduction pattern (39+2+2+1+1). The hamming loss curve is plotted as shown in Figure 7.2.

![Hamming Loss Plot](chart.png)

Figure 7.2: Hamming Loss Plot for Medical (39+2+2+1+1) Dataset

From the figure it is evident that, the hamming loss significantly decreases every time a new label is introduced. This proves that the proposed technique learns multiple new labels being introduced at multiple time instances on the run. Finally, the proposed technique is experimented with corel5k dataset with label introduction pattern of (368+2+1+1+2). The results obtained are shown in Figure 7.3.
From the test results, the expected behaviour of the progressive learning method is verified. The result proves that the developed algorithm is capable of learning multiple new labels dynamically on the run by retaining the knowledge of previously learnt labels.

7.3.2 Consistency

Analyzing the consistency of the proposed approach is of prime importance. The proposed technique should be consistent across multiple trials without much variance. In order to evaluate the consistency, we have used cross-validation technique. The goal of cross-validation is to identify the performance measures of the model on a data set that is independent of the data that were used for the training. K-fold cross-validation technique is used to measure the consistence. In
particular, a 5-fold cross validation and a 10-fold cross validation is performed on the proposed technique for different combinations of label introduction pattern on different datasets. The hamming loss performance metric is used to quantitatively measure the correctness of the classifier. The results are tabulated in Table 7.2.

Table 7.2: Consistency

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label introduction pattern</th>
<th>Hamming loss 5-fcv</th>
<th>Hamming loss 10-fcv</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>12 + 2</td>
<td>0.196 ± 0.001</td>
<td>0.195 ± 0.002</td>
</tr>
<tr>
<td>Yeast</td>
<td>10 + 1 + 3</td>
<td>0.198 ± 0.001</td>
<td>0.199 ± 0.001</td>
</tr>
<tr>
<td>Scene</td>
<td>5+1</td>
<td>0.104 ± 0.002</td>
<td>0.106 ± 0.001</td>
</tr>
<tr>
<td>Scene</td>
<td>4+1+1</td>
<td>0.139 ± 0.002</td>
<td>0.140 ± 0.002</td>
</tr>
<tr>
<td>Medical</td>
<td>44+1</td>
<td>0.012 ± 0.001</td>
<td>0.011 ± 0.001</td>
</tr>
<tr>
<td>Medical</td>
<td>39+2+2+1+1</td>
<td>0.016 ± 0.001</td>
<td>0.016 ± 0.001</td>
</tr>
<tr>
<td>Medical</td>
<td>38+3+4</td>
<td>0.023 ± 0.001</td>
<td>0.022 ± 0.002</td>
</tr>
<tr>
<td>Corel5k</td>
<td>373+1</td>
<td>0.010 ± 0.000</td>
<td>0.009 ± 0.000</td>
</tr>
<tr>
<td>Corel5k</td>
<td>368+2+1+1+2</td>
<td>0.010 ± 0.002</td>
<td>0.010 ± 0.001</td>
</tr>
</tbody>
</table>
7.3.3 Performance Metrics

Due to the possibility of the partial correctness of the classifier result, one specific metric will not be sufficient to quantitatively measure the performance of a technique. Therefore, a set of quantitative performance evaluation metrics is used to validate the performance of the multi-label classifier. The performance metrics used are hamming loss, accuracy, precision, recall and F₁-measure. The performance metrics of the proposed technique for each of the dataset and label introduction pattern are evaluated and shown in Table 7.3.

Table 7.3: Performance Metrics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label introduction pattern</th>
<th>Hamming loss</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast 12+2</td>
<td>0.196</td>
<td>0.495</td>
<td>0.685</td>
<td>0.583</td>
<td>0.634</td>
<td></td>
</tr>
<tr>
<td>Yeast 10+1+3</td>
<td>0.198</td>
<td>0.492</td>
<td>0.679</td>
<td>0.578</td>
<td>0.629</td>
<td></td>
</tr>
<tr>
<td>Scene 5+1</td>
<td>0.104</td>
<td>0.609</td>
<td>0.627</td>
<td>0.659</td>
<td>0.643</td>
<td></td>
</tr>
<tr>
<td>Scene 4+1+1</td>
<td>0.139</td>
<td>0.569</td>
<td>0.584</td>
<td>0.699</td>
<td>0.636</td>
<td></td>
</tr>
<tr>
<td>Medical 44+1</td>
<td>0.012</td>
<td>0.693</td>
<td>0.740</td>
<td>0.729</td>
<td>0.734</td>
<td></td>
</tr>
<tr>
<td>Medical 39+2+2+1+1</td>
<td>0.016</td>
<td>0.653</td>
<td>0.695</td>
<td>0.731</td>
<td>0.712</td>
<td></td>
</tr>
<tr>
<td>Medical 38+3+4</td>
<td>0.023</td>
<td>0.585</td>
<td>0.622</td>
<td>0.739</td>
<td>0.675</td>
<td></td>
</tr>
<tr>
<td>Corel5k 373+1</td>
<td>0.010</td>
<td>0.057</td>
<td>0.164</td>
<td>0.059</td>
<td>0.087</td>
<td></td>
</tr>
<tr>
<td>Corel5k 368+2+1+1+2</td>
<td>0.010</td>
<td>0.055</td>
<td>0.151</td>
<td>0.062</td>
<td>0.088</td>
<td></td>
</tr>
</tbody>
</table>
7.3.4 Execution Speed

The performance of the proposed method in terms of execution speed is evaluated by the training time and the testing time of the algorithm used. The training and testing time taken by the proposed method for the aforementioned datasets with different label introduction patterns is tabulated as given in Table 7.4.

Table 7.4: Speed

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label introduction pattern</th>
<th>Train time</th>
<th>Test time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast 12+2</td>
<td>0.113</td>
<td>0.015</td>
<td></td>
</tr>
<tr>
<td>Yeast 10+1+3</td>
<td>0.115</td>
<td>0.012</td>
<td></td>
</tr>
<tr>
<td>Scene 5+1</td>
<td>2.266</td>
<td>0.042</td>
<td></td>
</tr>
<tr>
<td>Scene 4+1+1</td>
<td>2.291</td>
<td>0.080</td>
<td></td>
</tr>
<tr>
<td>Medical 44+1</td>
<td>0.604</td>
<td>0.025</td>
<td></td>
</tr>
<tr>
<td>Medical 39+2+2+1+1</td>
<td>0.611</td>
<td>0.034</td>
<td></td>
</tr>
<tr>
<td>Medical 38+3+4</td>
<td>0.614</td>
<td>0.027</td>
<td></td>
</tr>
<tr>
<td>Corel5k 373+1</td>
<td>5.348</td>
<td>0.044</td>
<td></td>
</tr>
<tr>
<td>Corel5k 368+2+1+1+2</td>
<td>5.396</td>
<td>0.056</td>
<td></td>
</tr>
</tbody>
</table>
From the table it can be seen that the proposed method has very high execution speed. The proposed technique exploits the inherent high speed nature of the extreme learning machine to perform classification at very high speed.

7.3.5 Experimentation on Real-world Dataset

In order to evaluate the performance of the proposed progressive learning multi-label classifier (PL-MLC), we use the aspect detection dataset described in Chapter 4. As discussed earlier, aspect detection is a typical multi-label problem with each input sentence having multiple aspect terms. For experimentation, we chose two datasets (Laptop and Restaurant review datasets) from SemEval 2014 [102]. The specifications of the datasets are discussed earlier in Chapter 4. The performance of the proposed method is compared with the state of the art technique available for aspect detection by Poria et. al (2016) [103]. The performance metrics used for comparison are Precision, Recall and F-measure. The key advantage of the proposed method is its ability to learn new classes on the run dynamically. The results obtained are tabulated in Table 7.5. From the table, it can be seen that the proposed progressive learning multi-label classifier performs well not only for benchmark datasets but also for real-world datasets.
Table 7.5: Experimentation Results for Aspect Detection

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Poria et al [103]</th>
<th>PL-MLC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Laptop</td>
<td>0.867</td>
<td>0.783</td>
</tr>
<tr>
<td>Restaurant</td>
<td>0.882</td>
<td>0.861</td>
</tr>
</tbody>
</table>

7.4 Summary

The proposed progressive multi-label classifier is the first of its kind. It can be used for online/streaming data applications with known number of labels as well as for real world real-time applications with dynamic and unknown number of labels. From the results it can be seen that the proposed method shows high speed and high performance metric for all the datasets used for verification and for different patterns of introduction of new labels.
Chapter 8

Progressive Learning – Universal Classifier

Progressive learning is the next stage of advancement to the online learning methods. Existing online sequential techniques only learn to classify data among a fixed set of classes which are initialized during the initialization phase of the algorithm. They fail to dynamically adapt when introduced to new class/classes on the run. The progressive learning technique is independent of the number of class constraint and it can learn several new classes on the go by retaining the knowledge of previous classes. This is achieved by modifying the network structure by itself upon encountering a new class and updating the network parameters in such a way that it learns the new class and retains the knowledge learnt thus far.

The existing classifiers, once trained to classify a specific number of labels, cannot learn new labels without retraining all the labels anew again. Such
classifiers work well with the pre-known dataset, but they may not be apt for real-world, real-time applications where the nature of training data is unknown. For such data, the learning technique must be self-developing to adapt to dynamic needs. To overcome this shortcoming, the progressive learning technique earlier discussed is used. Also, as foreshadowed, based on the number of target labels associated with each input sample, the classification problems can be broadly categorized into single-label classification and multi-label classification. Single-label classification problems can be further divided into binary and multi-class classification problems. All the existing classifiers in the literature are limited to the corresponding classification problem they are trained to. There is no classifier that can be universally applicable to all the aforementioned classification problems incorporating the progressive learning technique.

In this chapter, the progressive learning technique will be integrated with the label-independent/universal classifier to achieve human-learning-inspired progressively learning universally generic classifier. The resulting new classifier can be used for any type of classification problem and any number of dynamic class constraints. The newly developed classifier having developed based on the extreme learning machine framework, exploits its inherent high speed training and testing. Experiments are conducted with datasets from all three classification types and various performance metrics are evaluated.
8.1 Proposed Approach

The final outcome of the thesis, development of human learning inspired progressive learning multi-label classifier is developed by integrating the LIC from chapter 5 and progressive learning from chapter 6 and chapter 7. Being an universal classifier, the three key challenges to be addressed.

1. Identification of classification type
2. Estimating the number of target labels corresponding to each input sample
3. Identifying each of the associated target labels.

The various steps involved in the proposed algorithm are summarized.

Initialization Phase. Initialization Phase involves setting up the fundamental network parameters for the target classification problem. Being an ELM based technique, the input weights and the bias values are randomly initialized. The number of hidden layer neurons and the activation function are assigned. The number of hidden layer neurons is to be selected such that the problem of overfitting is avoided.

Data Pre-processing Phase. The proposed algorithm needs to be capable of classifying both single-label and multi-label classification problems. The representation of data varies among each of the classification types. In binary and multi-class classification, the output is represented as a single value which identifies the unique target class that is associated with the input sample. On the
other hand, in multi-label classification, since each input can have multiple labels, the output is represented as a vector with dimensions equal to the total number of output labels. Thus, proper pre-processing of data is essential in achieving universal classifier. In the proposed approach, the target label of all three classification types is represented as a vector with dimension equal to the number of output labels. Each element in the vector signifies the belongingness of the input to the corresponding label.

**Training Phase.** During the training phase, the data samples and the target labels are provided as the input and the output weight values are estimated iteratively by online training. The proposed method is based on the online variant of ELM. The online training phase has two steps.

**Initial Block Step:** Let N0 be the number of training samples in the initial block of data, the initial output weight values are calculated as discussed earlier.

**Sequential Training Step:** Upon completion of the initial block step, the subsequent data samples arriving sequentially are processed in the sequential training step. The subsequent data that arrives to the network can be trained either on one-by-one or chunk-by-chunk basis. Let ‘b’ be the chunk size. Unity value for b results in training the network on one-by-one basis.

When a new data sample/chunk of data is arrived, it can fall into either of the two categories.
i. Absence of new class of data

ii. Presence of new class / classes of data

Case (i) Absence of new class of data

If there are no new classes in the current set of data, the sequential training step is similar to online universal classifier and the usual process of calculating and updating the output weights is performed. The hidden layer output vector \( h_{k+1} \) is calculated. Then, the output weight is updated based on the RLS algorithm discussed earlier.

Case (ii) Presence of new class of data

Similar to online universal classifier from chapter 5, if there is a new class(es) in the chunk of data arrived, the progressive learning technique discussed in previous chapters is used to recalibrate the network and to accommodate learning of new class by retaining old knowledge.

Consider there are \( N' \) hidden layer neurons in the network, \( m \) classes of data are currently learnt by the network and \( b \) be the chunk size of the sequential learning. The introduction of ‘c’ new classes at any instant \( k+1 \), will modify the dimensions of the output weight matrix \( \beta \) from \( \beta_{N' \times m} \) to \( \beta_{N' \times m+c} \). The values of \( \beta_{N' \times m+c} \) are calculated based on the current values of \( \beta_{N' \times m}, (h_k)_{b \times N'} \) and \( (M_k)_{N' \times N'} \). The current \( \beta \) matrix is of the dimension \( (\beta_k)_{N' \times m} \) and ‘c’ new classes are introduced. Therefore, to accommodate the output weight matrix for the increased number of output layer neurons, the \( \beta \) matrix is transformed to \( \tilde{\beta}_k \) as given in equation.
\[
\beta_k = (\beta_k)_{N'xM} I_{mXm+c}
\]

Where \( I_{mXm+c} \) is a rectangular identity matrix of dimension \( m \times (m+c) \).

\[
\beta_{k_{N'xM+c}} = (\beta_k)_{N'xM} \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \end{bmatrix}_{mXm+c}
\]

\[
\beta_{k_{N'xM+c}} = [(\beta_k)_{N'xM} O_{N'xM+c}]_{N'xM+c}
\]

where \( O_{N'xM+c} \) is zero matrix.

Upon extending the weight matrix to accommodate the increased number of output neurons, the learning learnt thus far has to be incorporated in the newly upgraded weight matrix. Appending zero matrix is a trivial way to increase the dimensions. The matrix values have to be updated such that the network retains the knowledge of existing classes and can learn new classes as if they were available from the beginning of the training phase.

From equation 18, it can be seen that, the error difference between the target class \( t_{k+1} \) and the predicted class \( h_{k+1} \beta_k \) is scaled by a learning factor and is added to \( \beta_k \). Since ‘c’ new classes are introduced only at the \( k+1 \)th time instant, for the initial \( k \) data samples, the target class label value corresponding to the new class is -1. Therefore, the \( k \)-learning step update for the ‘c’ new classes \( ((\Delta \beta_k)_{N'Xc}) \) can be written as,

\[
(\Delta \beta_k)_{N'Xc} = (M_k)_{N'xN'} (h_k^T)_{N'xb} \begin{bmatrix} -1 & \ldots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \ldots & -1 \end{bmatrix}_{bXc}
\]
\[(\Delta \beta_k)_{N'Xc} = -(M_k)_{N'XN'} (h_k^T)_{N'Xb} J_{bXc}\]  

where \(J_{bXc}\) is an all-ones matrix.

\[J_{bXc} = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}_{bXc}\]

The \(k\)-learning step update for the new classes is then incorporated with the \(\tilde{\beta}_{kmXm+c}\) to provide the upgraded \((\beta_k)_{N'X(m+c)}\) matrix which is recalibrated to adapt learning ‘\(c\)’ new classes.

\[(\Delta \tilde{\beta}_k)_{N'Xm+c} = [O_{N'Xm} - (M_k)_{N'XN'} (h_k^T)_{N'Xb} J_{bXc}]\]

The recalibrated output weight matrix \((\beta_k)_{N'X(N'+c)}\) is calculated as,

\[(\beta_k)_{N'X(m+c)} = \tilde{\beta}_{kmXm+c} + (\Delta \tilde{\beta}_k)_{N'Xm+c}\]

Upon simplification, \((\beta_k)_{N'X(N'+c)}\) can be expressed as,

\[(\beta_k)_{N'X(m+c)} = \begin{bmatrix} (\beta_k)_{N'Xm} & (\Delta \beta_k)_{N'Xc} \end{bmatrix}\]

\((\beta_k)_{N'Xm}\) represents the knowledge previously learnt. The dimension of \(\beta\) is increased from \(m\) to \(m+c\). As opposed to populating the increased dimension with identity matrix values, the new entries \((\Delta \beta_k)_{N'Xc}\) are calculated in such a way that the newly introduced classes will appear to the neural network as if they are present from the beginning of the training procedure and the training data samples thus far does not belong to the newly introduced class.

The network is recalibrated such that the \((\Delta \beta_k)_{N'Xc}\) matrix represents the learning of the new class from the beginning of the training phase to the current data sample.
considering that none of the previous data samples belong to the newly introduced class. i.e. The \((\Delta \beta_k)^N_{Xc}\) is computed which is equivalent to the k-learning step equivalent of the ‘c’ new classes from the beginning of the training phase.

Therefore the updated \((\beta_k)^N'_{X(m+c)}\) matrix represents the network with \((m+c)\) classes with ‘m’ previously existing classes and ‘c’ new classes.

The hidden layer output vector \(h_{k+1}\) is calculated and the output weight matrix of increased dimension to facilitate learning of new class is updated based on the RLS algorithm as discussed earlier.

**Testing Phase.** In the testing phase, the target output of the input samples is predicted using the values of \(\beta\) estimated from the training phase and the input data samples. The raw output values of the network are evaluated using the relation \(Y = H\beta\). The raw output value obtained from the testing phase is then processed to address the three challenges of the universal classifier.

**Classification Phase.** In the classification phase, the raw output values \(Y\) obtained from the training phase is used to predict the classification type, number of associated target labels and identifying each of the target labels corresponding to each input sample.

*Identifying the Classification Type:* The classification type of binary, multi-class or multi-label is identified using the classification type (CT) value and dimension of output vector ‘l’. The CT value is evaluated using the equation.
\[ CT = |HS(Y)| \]  

where \( Y \) is the raw output vector and \( HS(x) \) is the heaviside function. Identification of classification type based on all possible valid combinations of CT and L is given in Table 8.1.

Table 8.1: Identification of Classification Type

<table>
<thead>
<tr>
<th>CT = 1</th>
<th>L = 2</th>
<th>Binary Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT = 1</td>
<td>L &gt; 2</td>
<td>Multi-class Classification</td>
</tr>
<tr>
<td>CT &gt; 1</td>
<td>L &gt; 2</td>
<td>Multi-label Classification</td>
</tr>
</tbody>
</table>

*Estimating the Number of Target Labels:* Upon establishing the classification type, the number of target labels is then estimated as given in Table 8.2. For binary and multi-class classification, the number of target labels is one, since each input belongs to unique target labels. For multi-label classification, the CT value corresponds to the number of target labels associated with the input data sample.
Table 8.2: Estimation of Number of Target Labels

<table>
<thead>
<tr>
<th>Classification Type</th>
<th>Number of Target Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Classification</td>
<td>1</td>
</tr>
<tr>
<td>Multi-class Classification</td>
<td>1</td>
</tr>
<tr>
<td>Multi-label Classification</td>
<td>$\Sigma_{i=1}^L HS(Y_i)$</td>
</tr>
</tbody>
</table>

Algorithm 6: Proposed Progressive Learning Label-independent Classifier Algorithm

19. Initialization of parameters

20. Formatting input to uniform representation

21. Initial block training

Input: Initial N0 samples of data in the form \{(x_i,y_i)\}

Output: $\beta_0$

Evaluation:

$M_0 = (H_0^TH_0)^{-1}$

$\beta_0 = M_0H_0^TY_0$
22. Sequential training

Input: Sequentially arriving data blocks in the form \{(x_i, y_i)\}

Output: \( \beta_k \)

Evaluation:

Case 1: No new labels are introduced:

\[
M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}}
\]

\[
\beta_{k+1} = \beta_k + M_{k+1} h_{k+1} (Y_{k+1}^T - h_{k+1}^T \beta_k)
\]

Case 2: ‘c’ new labels are introduced:

\[
(\beta_k)_{N'X(m+c)} = [ (\beta_k)_{N'Xm} (\Delta \beta_k)_{N'Xc} ]
\]

\[
(\Delta \beta_k)_{N'Xc} = (M_k)_{N'XN'} (h_k^T)_{N'Xb} \begin{bmatrix} -1 & \ldots & -1 \\ \vdots & \ddots & \vdots \\ -1 & \ldots & -1 \end{bmatrix}_{bXc}
\]

\[
M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}}
\]

\[
\beta_{k+1} = \beta_k + M_{k+1} h_{k+1} (Y_{k+1}^T - h_{k+1}^T \beta_k)
\]

23. Evaluating raw output value \( Y \)

Input: Data sample \( x_i \)

Output: \( Y \)

Evaluation:
Y = Hβ

24. Evaluating CT value: \( CT = |H(Y)| \)

25. Identifying classification type based on CT and L

26. Evaluating number of associated target labels

27. Calculating the belongingness vector: \( B=H(Y) \)

28. Identifying the associated target labels

29. Evaluation of performance metrics corresponding to classification type

*Identifying the Target Labels:* The target labels are identified using the belongingness vector. The belongingness vector \( B \) is given as,

\[
B = HS(Y)
\]  

where \( Y \) is the raw output value and \( HS(x) \) is Heaviside function. Each element of the vector \( B \) denotes the belongingness of the input to the corresponding label. Thus, the label index of the non-zero entries of the \( B \) gives the target labels associated with the input samples. Upon estimating the target labels, the performance metrics of the classifier are evaluated. Thus, the proposed technique is capable of classifying all three types of classification problems. The overview of the proposed approach is summarized.
8.2 Experimentation

The experimental design, dataset specifications and the comparison methods used to evaluate the proposed method are discussed in this section. Similar to online universal classifier from chapter 5, since the proposed method is capable of classifying all the classification types, the datasets from both single-label and multi-label classification problems are chosen for experimentation. In order to evaluate the progressive learning nature of the proposed technique, the datasets are redistributed in a such a way that new classes are introduced at different time instances. The specifications of the dataset used for experimentation are given in the Table 8.3. The proposed method is evaluated using 5 single-label (2 binary and 3 multi-class) datasets and 4 multi-label datasets with different label introduction pattern.
### Table 8.3: Datasets Specifications

<table>
<thead>
<tr>
<th>Classification type</th>
<th>Dataset</th>
<th>No. of Labels</th>
<th>Label Introduction Pattern</th>
<th>Feature dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-label</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Binary</strong></td>
<td>Diabetes</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Ionosphere</td>
<td>2</td>
<td>2</td>
<td>34</td>
</tr>
<tr>
<td><strong>Multi-class</strong></td>
<td>Balance-scale</td>
<td>3</td>
<td>2+1</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Satellite image</td>
<td>6</td>
<td>4+2</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Digit</td>
<td>10</td>
<td>7+3</td>
<td>64</td>
</tr>
<tr>
<td><strong>Multi-label</strong></td>
<td>Scene</td>
<td>6</td>
<td>4+1+1</td>
<td>294</td>
</tr>
<tr>
<td></td>
<td>Yeast</td>
<td>14</td>
<td>10+1+3</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>Corel5k</td>
<td>374</td>
<td>368+2+1+1+2</td>
<td>499</td>
</tr>
<tr>
<td></td>
<td>Medical</td>
<td>45</td>
<td>38+3+4</td>
<td>1449</td>
</tr>
</tbody>
</table>
8.3 Results and Discussions

This section summarizes the experimental results of the proposed universal classifier on the datasets specified in table. The proposed method is evaluated for consistency, performance and speed.

8.3.1 Consistency

Consistency is one of the key virtues of any new technique developed. An algorithm that is inconsistent with results on different trials is unreliable. Cross-validation is one of the effective ways to evaluate the consistency of the method. A 10-fold cross validation is performed for each of the datasets. For single-label classification problems, the performance of the classifier is evaluated using the percentage of accuracy. For multi-label classification hamming loss is used as the performance metric. The results obtained are tabulated in Table 8.4. From the table, it can be seen that the proposed universal classifier is highly consistent for all datasets from binary, multi-class and multi-label classification.
### Table 8.4: Consistency

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label introduction pattern</th>
<th>Accuracy % (10-fcv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>2</td>
<td>78.2 ± 3.2</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>2</td>
<td>96.4 ± 2.6</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>2+1</td>
<td>91.2 ± 1.27</td>
</tr>
<tr>
<td>Satellite image</td>
<td>4+2</td>
<td>89.3 ± 1.61</td>
</tr>
<tr>
<td>Digit</td>
<td>7+3</td>
<td>97.0 ± 1.057</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label introduction pattern</th>
<th>Hamming loss (10-fcv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene</td>
<td>4+1+1</td>
<td>0.139 ± 0.002</td>
</tr>
<tr>
<td>Yeast</td>
<td>10+1+3</td>
<td>0.198 ± 0.001</td>
</tr>
<tr>
<td>Corel5k</td>
<td>368+2+1+1+1+2</td>
<td>0.009 ± 0.000</td>
</tr>
<tr>
<td>Medical</td>
<td>38+3+4</td>
<td>0.023 ± 0.001</td>
</tr>
</tbody>
</table>

#### 8.3.2 Performance

The performance of binary and multi-class classification can be evaluated using the testing accuracy and for multi-label classification it can be evaluated using hamming loss. The results are tabulated in Table 8.5. There is no progressive
learning universal classifier available in the literature to perform direct comparison with the proposed method.

Table 8.5: Performance Measure

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label introduction pattern</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>2</td>
<td>78.2</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>2</td>
<td>96.4</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>2+1</td>
<td>90.7</td>
</tr>
<tr>
<td>Satellite image</td>
<td>4+2</td>
<td>89.6</td>
</tr>
<tr>
<td>Digit</td>
<td>7+3</td>
<td>97.1</td>
</tr>
</tbody>
</table>

Since, due to the presence of partial correctness problem, several metrics in addition to hamming loss is required for multi-label classification problem. Therefore, the metrics such as accuracy, precision, recall and F1 measure are evaluated for each case. The results obtained are given in Table 8.6.
Table 8.6: Performance Metrics for Multi-label Case

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label introduction pattern</th>
<th>Hamming loss</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>10+1+3</td>
<td>0.198</td>
<td>0.492</td>
<td>0.679</td>
<td>0.578</td>
<td>0.629</td>
</tr>
<tr>
<td>Scene</td>
<td>4+1+1</td>
<td>0.139</td>
<td>0.569</td>
<td>0.584</td>
<td>0.699</td>
<td>0.636</td>
</tr>
<tr>
<td>Medical</td>
<td>38+3+4</td>
<td>0.023</td>
<td>0.585</td>
<td>0.622</td>
<td>0.739</td>
<td>0.675</td>
</tr>
<tr>
<td>Corel5k</td>
<td>368+2+1+1+2</td>
<td>0.010</td>
<td>0.055</td>
<td>0.151</td>
<td>0.062</td>
<td>0.088</td>
</tr>
</tbody>
</table>

8.3.3 Execution Speed

The execution speed of the proposed classifier is evaluated in terms of training time and testing time. Execution speed of the classifier plays a vital role for streaming data applications. In order to perform real-time streaming data classification, the execution speed of the classifier should be less than the arrival rate of the streaming data. Therefore, a high speed classifier is essential for real-time streaming data applications. The proposed progressive learning label-independent classifier (PL-LIC) exploits the inherent high-speed nature of the ELM. The training time and the testing time of the proposed PL-LIC algorithm for each dataset is given in Table 8.7.
Table 8.7: Execution Speed

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Training Time (s)</th>
<th>Testing Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-label Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diabetes</td>
<td>0.005</td>
<td>0</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.007</td>
<td>0</td>
</tr>
<tr>
<td>Iris</td>
<td>0.002</td>
<td>0</td>
</tr>
<tr>
<td>Waveform</td>
<td>0.012</td>
<td>0.001</td>
</tr>
<tr>
<td>Balance-scale</td>
<td>0.009</td>
<td>0</td>
</tr>
<tr>
<td><strong>Multi-label Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scene</td>
<td>2.291</td>
<td>0.080</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.115</td>
<td>0.012</td>
</tr>
<tr>
<td>Corel5k</td>
<td>5.396</td>
<td>0.056</td>
</tr>
<tr>
<td>Medical</td>
<td>0.614</td>
<td>0.027</td>
</tr>
</tbody>
</table>

From the table, it is evident that the proposed classifier is capable of performing progressive learning of multiple new classes dynamically for binary, multi-class and multi-label classification problems.
8.4 Summary

The progressive learning technique is integrated with the universal classifier to develop human-learning-inspired progressively learning universally generic classifier. The resulting new classifier can be used for any type of classification problem and any number of dynamic class constraints. The newly developed classifier is developed based on the extreme learning machine framework, thereby exploiting its inherent high speed training and testing. The proposed classifier is evaluated and analyzed in terms of consistency, performance and speed.
Chapter 9

Conclusions and Future Directions

9.1 Conclusions

This thesis aimed to achieve: 1. The development of a label-independent (single label or multi-label) generic classifier that is capable of performing binary, multi-class and multi-label classification, 2. Extreme learning machine based progressive learning technique for classification problems and 3. Integration of progressive learning technique to the label-independent classifier thereby resulting in a generic classifier. The progressive learning label-independent classifier which is inspired by the human learning process is capable of learning new classes dynamically and also capable of addressing binary, multi-class and multi-label classification problems.

In order to achieve the aforementioned objectives, several new techniques have been formulated. In chapter 3, an efficient high-speed ELM based multi-label
classifier is developed. This forms the basic research framework for developing OSML-ELM classifier in Chapter 4. Based on the OSML-ELM classifier, in Chapter 5, a label-independent/universal classifier capable of performing binary, multi-class and multi-label problems is developed. It is to be highlighted that there are no universal classifiers available in the literature thus far. Subsequently, the progressive learning theory which is inspired from human learning process is developed and implemented for multi-class classification problems in chapter 6. In chapter 7, the previously developed progressive learning technique is combined with the OSML-ELM from chapter 4 and progressive learning multi-label classifier is developed. Finally, the ultimate outcome of the thesis is achieved by using the progressive learning multi-label classifier from chapter 7 as the base platform and extending the algorithm to achieve progressive learning technique for label-independent classifier.

The newly developed classifier based on the extreme learning machine exploits its inherent high-speed training and testing. Thus, the resulting new classifier can be used to classify any type of classification problem with dynamic class constraints with high speed and accuracy.

9.2 Directions for Further Research

This section discusses some of the possible future research directions based on the current research work.
In this thesis, the progressive learning technique and the universal classifier are realized using ELM based algorithm. Any machine learning algorithm can be modified and adapted to achieve the progressive learning and universal classification.

Furthermore, the progressive learning universal classifier can be used to develop experiential learning algorithms based on David A. Kolb’s experiential learning theory [101]. Experiential learning is an effective learning model which is demonstrated by the human learning process. It is the process of making meaning or retrieving information from new experience/data. By integrating the progressive learning technique with the experiential learning theory, the higher order cognitive functions specified by Kolb such as reflective observation, abstract conceptualization, active experimentation and concrete experience can be achieved for machine learning methods.
Author’s Publications


References


ACM SIGIR conference on Research and development in information retrieval, Salvador, Brazil, 2005.


187


