B. TECH. PROJECT REPORT

ON

Multi-label Classification using Single Layer Feedforward Neural Networks

By:

SHIVAM TAYAL

SHRESTHA KUMAR

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

December 5, 2018

Multi-label Classification using Single Layer Feedforward Neural Networks

A Project Report

Submitted in partial fulfillment of the
requirements for the award of the degree

of
Bachelor of Technology
in

Discipline of Computer Science and Engineering

Submittted by:
Shivam Tayal

Shrestha Kumar

Guided by:
Dr. Aruna Tiwari
Associate Professor

Indian Institute Of Technology Indore

December 5, 2018

Candidate's Declaration

We hereby declare that the project entitled **"Multi-label Classification using Single Layer Feedforward Neural Networks"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in **"Discipline of Computer Science and Engineering"** completed under the supervision of **Dr. Aruna Tiwari**, Associate Professor, Computer Science and Engineering, IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

(Shivam Tayal) **(Shrestha Kumar)**

Certificate by BTP Guide

It is certified that the above statements made by the students are correct to the best of my knowledge.

Dr. Aruna Tiwari
Associate Professor,
Discipline of Computer Science and Engineering
Indian Institute Of Technology Indore

Preface

This report on **"Multi-label Classification using Single Layer Feedforward Neural Networks"** is prepared under the guidance of **Dr. Aruna Tiwari**, Associate Professor, Computer Science and Engineering.

Through this report we have tried to solve Multi-label Classification problem using Single Layer Feedforward Neural Network. We have tried to cover every aspect of proposed model and tested it on various datasets.

We have tried to the best of our abilities and knowledge to explain the content in a lucid manner. We have also added flowchart of proposed model, figures and pseudo-code of algorithms to make it more illustrative.

SHIVAM TAYAL
SHRESTHA KUMAR

B.Tech. IV Year
Discipline of Computer Science and Engineering
Indian Institute Of Technology Indore

Acknowledgements

We would like to thank our B.Tech Project guide **Dr. Aruna Tiwari** for her guidance and constant support in structuring the project and her valuable feedback throughout the course of this project. Her overseeing the project meant there was a lot that we learnt while working on it. We thank her for her time and efforts.

We are grateful to **Mr. Vikas Chauhan** without whom this project would have been impossible. He provided valuable guidance and also taught us how to write a thesis.

We are really grateful to the **Institute** for the opportunity to be exposed to systemic research especially Dr. Aruna Tiwari's Lab for providing the necessary hardware utilities to complete the project.

Lastly, we offer our sincere thanks to everyone who helped us complete this project, whose name we might have forgotten to mention.

Abstract

Deep structure neural networks have been applied in many fields of artificial intelligence particularly in large scale data processing. Among them the most popular deep networks are the Deep Beliefs Network(DBN), the Deep Boltzmann Machines(DBM) and the Convolutional Neural Networks(CNN). Although the deep structure has been so powerful, most of the networks suffer from the time consuming training process because of more parameters involved. This complication makes it difficult to analyze the network theoretically and we keep on stacking more layers to achieve more accuracy. To achieve this we need more and more powerful computation power and resources.

Our motivation behind this project is to propose a multi-label classification model which can be trained faster and requires less computational power than deep learing networks. In this project, a Single Layer Feedforward Neural Network(SLFNN) is proposed as they have been widely applied to solve problems such as classification and regression because of their universal approximation capability. Conventional methods to train SLFNN are gradient-descent based learning algorithms but they usually suffer from slow convergence and trap in local minimum. The proposed SLFNNs with our enhancements, Random Vector Functional Link(RVFL) and Broad Learning System(BLS) eliminate the drawbacks of existing multi-label classification models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Classification methodology have become very important tools in the field of machine learning and are widely used in pattern recognition and other fields. Traditionally classification problems comprised of assigning a single class or label to the instance from two(binary classification) or more than two(multi-class classification) labels. But in a real world scenario, a single instance can be assigned multiple labels. For example, in a movie classification problem a single movie can be assigned multiple labels such as drama, romantic and comedy. Multi-label(ML) classification problems are more universal and generalized version of the multi-class classification problems. But this generalization brings more challenges and difficulties to the problem as the size of the output spaces increases exponetially. Therefore, multi-label classification is a great field of research and developement to design an accurate and efficient multi-label classification algorithm in machine learning field.

Previous works on multi-label classification include ML-Backpropagation[1], ML-Support Vector Machine(ML-SVM)[2] and ML-K-Nearest Neighbours(ML-KNN)[3]. In this project, we propose two Single Layer Feedforward Neural Networks(SLFNNs) to tackle the problem of multi-label classification. The proposed SLFNN with our enhancements, Random Vector Functional Link(RVFL) neural network and Broad Learning System(BLS) neural network are non-iterative and thus eliminate the drawbacks of long training phase and trapping at local minima.

The effectiveness of proposed RVFL and BLS neural networks are evaluated by comparing the multi-label classification metrics like Hamming Loss, One-error, Coverage, Ranking Loss and Average Precision with other algorithms like Backpropagation, KNN and SVM. Our work justifies that the proposed neural networks can perform multi-label classification that is not reported for such an approach till date.

Chapter 2

Literature Survey

The following chapter discusses about the multi-class and multi-label classification. We will discuss about the approaches that have been proposed to solve the multi-label classification. We will discuss about different SLFNNs that we have used to solve the multi-label classification problems.

2.1 Multi-Class Classification

Multi-Class classification[4] is the problem of classifying one instance into one of the two or more classes. It is a generalized version of binary classification in which an instance has to classified between two classes. Formally a multi-class classification problem can be defined as:

- Let $\mathcal{X} = \mathbb{R}^n$ denotes a n-dimensional instance space of numerical or categorical features.
- $\mathcal{Y} = \{1, 2, ..., Q\}$ be a finite set of labels, where Q is the total number of labels.

The task of multi-class classifier is to get a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ by training samples which can map an unseen instance to any single label.

2.2 Multi-Label Classification

Multi-Label Classification[5] problems are a generalized version of multi-class classification problems. In multi-label classification, one instance can be assigned multiple labels at once. Formally the multi-label classification can be defined as :

- Let $\mathcal{X} = \mathbb{R}^n$ denotes a n-dimensional instance space of numerical or categorical features.
- $\mathcal{Y} = \{1, 2, ..., Q\}$ be a finite set of labels, where Q is the total number of labels.

- $D = \{(X_i, Y_i)|i = 1, 2, ..., N\}(X_i \in \mathcal{X}, Y_i \subseteq \mathcal{Y})$ denotes a training set with N instances. Each element of the label set is either 1 if the label is relevant and -1 for irrelevant labels.

The task of multi-label classifier is to get a classifier $h : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$ by training samples, which can map an unseen instance to a label set.

Multiple methods and approaches have been proposed to solve the multi-label classification problems effectively and more accurately. Mainly multi-label problems are solved using two approaches:

- Problem Transformation
- Algorithm Adaptation

2.2.1 Problem Transformation

Multi-label Classification problems are transformed to multi-class or multiple binary class classification problems using below techniques.

- Binary Relevance
- Label Power Set
- Classifier Chain

Binary Relevance(BR)

Binary Relevance[6] method extents to independently training one binary classifier for each label. In this method, a multi-label problem is converted into $|\mathcal{Y}|$ number of binary single-label classification problems where $\mathcal{Y}$ is a set of labels. Each of the binary classifiers votes separately to get the final result. This method of dividing the task into multiple binary tasks has something in common with the one-vs.-all (OvA, or one-vs.-rest, OvR) method for multi-class classification.

Label Power Set(LP)

Label Power Set[7] method generates a new class for every combination of labels and then solves the problem using multi-class classification approaches. The main drawback of this approach is the exponential growth in the number of classes, leading to several generated classes having very few labeled instances leading to overfitting. BR does not consider relationship between labels. This drawback of BR is overcome by LP, also called as LC (Label Cardinality).

Classifier Chain(CC)

Classifier chain[8] method, which is based on the BR method, overcomes the disadvantages of BR and achieves higher predictive performance, but still retains important advantages of BR, most importantly low time complexity. CC offers a general problem transformation method that inherits the efficiency of BR and competes with the high accuracy of more computationally complex methods. The Classifier Chain approach like LP, also try to overcome the drawback of BR. Similar to BR, a multi-label problem is transformed into $|\mathcal{Y}|$ number of single-label problems where $\mathcal{Y}$ denotes a set of labels and for each label $\mathcal{Y}_j(j = 1, 2, ..., Q)$, a separate binary classifier C_j is designed. But the input for each classifier C_j is different. Like LP classifier, CC also needs selection of base classifier, uses J48 as base classifier by default.

2.2.2 Algorithmic Adaptation

Many algorithms have been proposed to solve a multi-class or binary classification problems. Algorithmic adaptation method is updating those algorithms to adapt the features of multi-label classifier and directly solve the multi-label classification problem without transforming it into subset of sub-problems.
Some algorithms which have been used to solve multi-label classification problem are:

- Support Vector Machine(SVM)
- K-Nearest Neighbour(KNN)
- Backpropagation algorithm(BP)

The approach proposed by us to solve the multi-label classification problem falls under algorithmic adaptation category. We have updated the RVFL and BLS neural networks to accomodate the multi-label classifier features.

2.3 Random Vector Functional Link Neural Network

Random Vector Functional Link(RVFL)[9] neural network is a SLFNN with random projection. The weight and the bias values between the input layer nodes and the enhancement nodes are chosen randomly at the beginning of the training and is fixed for that value. The output weight values (i.e. the connection with the output layer neurons) are calculated using Moore-Penrose pseudo inverse method(4.2). The

direct connection between the input layer and the output layer is an effective and simple regularization technique preventing RVFL neural network from overfitting.

2.4 Broad Learning System Neural Network

Broad Learning System(BLS)[10] neural network is an effective and efficient way of incremental learning system without the need for deep architecture. Deep architecture suffers from large time in training process due to the multiple levels present in the network. BLS neural network is a flat neural network, where the original inputs are transferred and placed as “mapped features” in feature nodes and the structure is expanded in wide sense in the “enhancement nodes". BLS neural network can be remodeled in an incremental way without the entire retraining from the beginning.

2.5 Multi-label Classification Metrics

The multi-label classification problem cannot be judged on the same parameters as that of the multi-class problems such as accuracy. The metrics[11] defined for the measurement of the correctness of multi-label classification model are as follows:

Let the total number of labels be Q , N to be the number of instances and Y be the set of labels present in a particular instance. The dataset $D = \{(X_i, Y_i)|i = 1, 2, ..., N\}(X_i \in \mathcal{X}, Y_i \subseteq \mathcal{Y})$.

2.5.1 Hamming Loss

Hamming loss is defined as the hamming distance between the predicted value and the actual result i.e. it counts for the number of instances where the predicted values is not equal to the actual result.

$$\text{hloss}_S(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{Q}|h(\mathbf{x}_i)\Delta Y_i|$$

where $h(x_i)$ is the predicted value for the instance x_i and Δ corresponds to the symmetric difference.
The smaller the value of the hamming loss, the better the performance of the model.

2.5.2 One Error

One error is defined as the number of times the top-ranked label is not present in the set of proper labels of the instance.

$$\text{one} - \text{error}_S(f) = \frac{1}{p}\sum_{i=1}^{p}[[\arg\max_{y\in\mathcal{Y}} f(\mathbf{x}_i,\ y)] \notin Y_i]]$$

The smaller the value of the one error, the better the performance of the model.

2.5.3 Coverage

Coverage is defined as the summation of the rank of the most insignificant label which belongs to the instance. In other words it says how many labels we have to check in order to cover all the proper labels of that instance.

$$\text{coverage}_S(f) = \frac{1}{p}\sum_{i=1}^{p}\max_{y\in Y_i} rank_f(\mathbf{x}_i, y) - 1.$$

The smaller the value of the coverage, the better the performance of the model. $rank_f()$ is derived from the real-valued function $f()$, which maps the outputs of $f(x_i, y)$ for any $y \in Y$ to $1, 2, \ldots, Q$ such that if $f(x_i, y_1) > f(x_i, y_2)$, then $rank_f(x_i, y_1) < rank_f(x_i, y_2)$.

2.5.4 Ranking Loss

Ranking Loss is defined as the average fraction of label pairs that are reversely ordered for the instance.

$$\text{rloss}_S(f) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i||\bar{Y}_i|}$$
$$|\{(y_1, y_2)|f(\mathbf{x}_i, y_1) \leq f(\mathbf{x}_i, y_2), (y_1)y_2) \in Y_i \times \bar{Y}_i\}|$$

The smaller the value of the ranking loss , the better the performance of the model.

2.5.5 Average Precision

Average Precision is defined as the average fraction of labels ranked above a particular label $y \in Y$ which actually are in Y.

$$\text{avgprec}_S(f) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|\{y'|rank_f(\mathbf{x}_i, y') \leq rank_f(\mathbf{x}_i, y), y' \in Y_i\}|}{rank_f(\mathbf{x_i}, y)}$$

The bigger the value of the average precision , the better the performance of the model.

Chapter 3

Analysis and Objectives

3.1 Analysis

Multi-label classification problem have been a field of research and development because of its wide range of application in real world. Many approaches have been proposed to solve the multi-label classification problem. Deep learning algorithms like Backpropagation and other algorithms based on Support Vector Machine(SVM) and K-Nearest Neighbour(KNN) are used to solve the multi-label classification problem. Deep learning networks suffer with the problem of very large training time as they comprises of multiple layers and lots of computation. The method proposed by us utilizes SLFNNs. Training process of SLFNNs is non-iterative process which reduces the training time drastically.

3.2 Objectives

The objectives of this project are:

- Understanding the concept of multi-label classification and its difference from multi-class classification using the present algorithms like backpropagation algorithm.
- Understanding the working of Random Vector Functional Link(RVFL) neural network and implementing it for multi-label classification.
- Understanding the working of Broad Learning System(BLS) neural network and implementing it for multi-label classification problem.
- Comparing the results of BLS and RVFL neural networks with other multi-label classification algorithms such as ML-Backpropagation, ML-KNN and ML-SVM.

Chapter 4

Single Layer Feedforward Neural Network

SLFNN have been widely applied to solve problems such as classification and regression because of their universal approximation capability. Conventional methods for training SLFNN are gradient-descent-based learning algorithms. The generalization performance of them is more sensitive to the parameter settings such as learning rate. Similarly, they usually suffer from slow convergence and trap in a local minimum.

4.1 Random Vector Functional Link Neural Network

Random Vector Functional Link(RVFL) neural network is the bridge between the neural networks and the Learning Using Privileged Information(LUPI) paradigm. The randomization of the parameters of the hidden layers with nonconstant piecewise continuous activation function can separate arbitrary regions of any shapes. The direct connection between the input layer and the output layer prevents the RVFL neural network from overfitting.

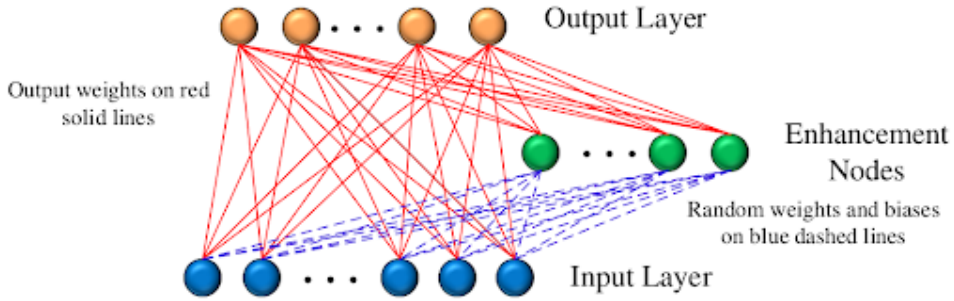

FIGURE 4.1: RVFL neural network

Given a set of labeled data $D = \{(X_i, Y_i)|i = 1, 2, ..., N\}(X_i \in \mathcal{X}, Y_i \subseteq \mathcal{Y})$, RVFL neural network with P enhancement nodes can be formulated as :

$$\mathbf{HW} = \mathbf{Y} \tag{4.1}$$

where **W** is an output weight vector , **H** is the concatenated output matrix combining input data and the output from the enhancement nodes and **Y** is a label matrix of dimensions $N \times Q$, where Q is the number of labels.

Algorithm for Random Vector Functional Link Neural Network

Input : training samples **X**
Output: W

1. H_1 matrix is the input vector matrix

$$H_1 \equiv X = \begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{Nn} \end{bmatrix}$$

2. Generate weight matrix A and bias B values between the enhancement layer and input layer. All the weight and the bias values are randomly selected from the range $[-u, u]$ and $[0, u]$ respectively where u is a positive user defined parameter.

3. The output of the enhancement layer is calculated using A, B and H_1

$$H_2 = \begin{bmatrix} G(a_1 \cdot x_1 + b_1) & \dots & G(a_P \cdot x_1 + b_P) \\ \vdots & \ddots & \vdots \\ G(a_1 \cdot x_N + b_1) & \dots & G(a_P \cdot x_N + b_P) \end{bmatrix}$$

4. The input to the output layer is calculated by concatenating the input matrix H_1 and the output from enhancement layer H_2

$$H \equiv [H_1 H_2]$$

5. The weight matrix between the output layer and the input and enhancement layer is calculated using the Moore-Penrose Pseudo inverse(4.2)

$$W = H^{\dagger} Y$$

Weight matrix **W**,

$$W = \begin{bmatrix} w_1^T \\ \vdots \\ w_{(n+P)}^T \end{bmatrix}$$

where w_i^T is a vector of dimension Q.

4.2 Moore-Penrose Pseudo Inverse

The weight matrix is calculated by taking the pseudo inverse of the concatenated matrix and multiplying it with output matrix. We define error, S in the network as a function of W. The goal is to minimise the difference between Y and HW and also to reduce the weight values.

$$S(W) = \frac{1}{2}||Y - HW||_2^2 + \frac{1}{2C}||W||_2^2 \tag{4.2}$$

where C is a user defined parameter.

$$\begin{aligned} S(W) &= \frac{1}{2}(Y - HW)^T(Y - HW) + \frac{1}{2C}W^TW \\ &= \frac{1}{2}(Y^T - W^TH^T)(Y - HW) + \frac{1}{2C}W^TW \\ &= \frac{1}{2}(Y^TY - Y^THW - W^TH^TY + W^TH^THW) + \frac{1}{2C}W^TW \end{aligned} \tag{4.3}$$

Differentiating (4.3) w.r.t. W, we get;

$$\begin{aligned} \frac{\mathrm{d}S(W)}{\mathrm{d}W} &= \frac{1}{2}(-H^TY - H^TY + 2H^THW) + \frac{1}{C}(W) \\ &= -H^TY + H^THW + \frac{1}{C}(W) \end{aligned} \tag{4.4}$$

At minima $\frac{\mathrm{d}S(W)}{\mathrm{d}W} = 0$, substituting in (4.4), we get;

$$\begin{aligned} -H^TY + (H^TH + \frac{I}{C})W = 0 \\ (H^TH + \frac{I}{C})W = H^TY \end{aligned} \tag{4.5}$$

Assuming $(H^TH + \frac{I}{C})$ is invertible, multiplying (4.5) by $(H^TH + \frac{I}{C})^{-1}$, we get;

$$\begin{aligned} W &= (H^TH + \frac{I}{C})^{-1}H^TY \\ W &= H^{\dagger}Y \end{aligned} \tag{4.6}$$

Thus, we have Moore-Penrose Pseudo Inverse from (4.6).

$$H^{\dagger} = (H^T H + \frac{I}{C})^{-1} H^T \tag{4.7}$$

4.3 Broad Learning System Neural Network

Deep structure learning suffers from a time consuming training process because of a large number of connecting parameters in filters and layers. Moreover, it encounters a complete retraining process if the structure is not sufficient to model the system. BLS neural network is established in the form of a flat neutal network, where the original inputs are transferred and placed as "mapped features" in feature nodes and the structure is expanded in wide sense in the "enhancement nodes". The incremental learning algorithms are developed for fast remodeling in broad expansion without a retraining process if the network deems to be expanded.

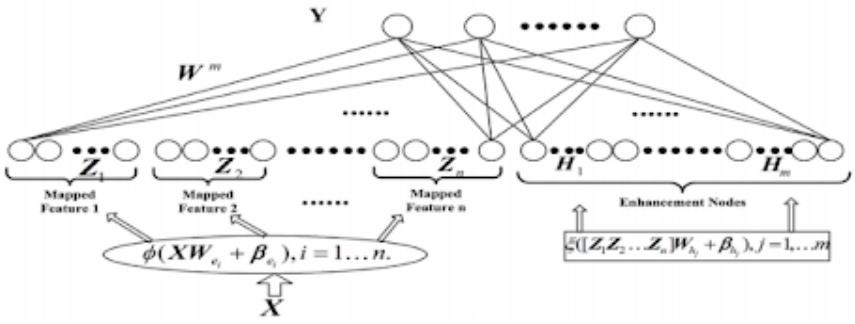

FIGURE 4.2: BLS neural network

The BLS neural network is constructed based on the above RVFL neural network. However, unlike the RVFL neural network that takes the input directly and establishes the enhancement nodes, BLS neural network first maps the inputs to construct a set of mapped features and then create the enhancement nodes from the mapped features. In addition, the randomly assigned weight values between the mapped nodes and the enhancement nodes are fine tuned using the sparse autoencoder for obtaining better features. We also develop incremental learning algorithms that can update the system dynamically.

4.3.1 Dynamically Updating the BLS neural network

In deep networks, we need to find out the efficient number of hidden neurons. We find that by varying the number of neurons and then re-training the network. BLS network provides us a advantage of not re-training the whole network on updation of the network.

Let us consider the expanded input matrix $A_n = [X|\sum(XW_h + \beta_h)]$ of dimension $n * m$ which contains all the input vectors combined with the enhancement components. If we want to update the network by adding one new enhancement node, the A_n matrix will be updated by adding one extra column to the matrix.

Let us say the new column to be added is a. Then, matrix $A_{n+1} \triangleq [A_n|a]$. Then the psuedo inverse of the new matrix:

$$A_{n+1}^{\dagger} = \begin{bmatrix} A_n^{\dagger} - db^T \\ b^T \end{bmatrix}$$

where $d = A_n^{\dagger}a$ and

$$b^T = \begin{cases} (c)^+ & \text{if } c \neq 0 \\ (1 + d^T d)^{-1} d^T A_n^+ & \text{if } c = 0 \end{cases}$$

and $c = a - A_n d$ (If A_n is a full rank matrix then c = 0)
The updated weight matrix of the network will be:

$$W_{n+1} = \begin{bmatrix} W_n - db^T Y_n \\ b^T Y_n \end{bmatrix}$$

where W_{n+1} and W_n is the weight matrix before and after adding new node respectively.

4.3.2 Sparse Autoencoder for tuning the Weight Values

The weight values between the input layer and the enhancement layer are randomly initialized at the begining of the training process. However, randomness suffers from the unpredictability and so to overcome the randomness nature, the Sparse Autoencoder is used which slightly fine-tunes the random features to a set of sparse and compact features.

The problem of extracting the sparse features from the training data X can be considered as the following optimization problem:

$$\arg\min_{\hat{W}} : \|Z\hat{W} - X\|_2^2 + \lambda\|\hat{W}\|_1 \tag{4.8}$$

where $\hat{W}$ is the sparse autoencoder solution and Z is the desired output of the linear equation $\mathbf{XW} = \mathbf{Z}$.

4.3.3 BLS Model

Assume that we present the input data X and project the data, using $\phi_i(XW_{e_i} + \beta_{e_i})$, to become the i-th mapped features, Z_i, where W_{e_i} is the random weights with the proper dimensions. Denote $Z^i \equiv [Z_1, ..., Z_i]$, which is the concatenation of all the first i groups of mapping features. Similarly, the j-th group of enhancement nodes, $\xi_j(Z_i W_{h_j} + \beta_{h_j})$ is denoted as H_j , and the concatenation of all the first j groups of enhancement nodes are denoted as $H^j \equiv [H_1, ..., H_j]$.

In BLS neural network, we fine tune the intial W_{e_i} to obtain the better features by applying the sparse autoencoder characterstics.

Assume the input data set X, which equips with N samples, each with m dimensions, and Y is the output matrix which belongs to $\mathbb{R}^{NxC}$. For n feature mappings, each mapping generates k nodes, can be represented as the equation of the form:

$$Z_i = \phi_i(XW_{e_i} + \beta_{e_i}), i = 1, ..., n \tag{4.9}$$

where W_{e_i} and β_{e_i} are randomly generated. Denote all the feature nodes as $Z_n = [Z_1, ..., Z_n]$, and denote the mth group of enhancement nodes as :

$$H_m \equiv \xi(Z^n W_{H_m} + \beta_{H_m}) \tag{4.10}$$

Hence, the broad model can be represented as the equation of the form:

$$\begin{aligned} Y &= [Z_1, ..., Z_n | \xi(Z^n W_{h_1} + \beta_{h_1}), ...\xi(Z^n W_{h_m} + \beta_{h_m})]W_m \\ &= [Z_1, ..., Z_n | H_1, ..., H_m]W_m \\ &= [Z^n | H^m]W^m \end{aligned} \tag{4.11}$$

where $W^m = [Z^n | H^m]^\dagger Y$, W^m are the connecting weights from the broad structure which can be easily calculated using the Moore-Penrose pseudo inverse method(4.2).

Chapter 5

Design Proposal

The above discussed SLFNNs have been used to solve the binary as well as multi-class classification problems. We propose an updation to the networks for solving the multi-label classification problems using threshold function.

After the training phase of the neural network is completed on the given dataset, we will calculate the threshold value for each data instance and classify each predicted label on the basis of threshold value.

5.1 Loopholes in Naive Approaches

Generally, in binary and multi-class classification models, maximum of predicted values of all the labels is considered as the output or the relevant label. But, in multi-label classification, the output is not a single label but a label set. Also, many times labels in multi-label dataset are co-related with each other.

A constant threshold function can classify multiple associated labels but will lack co-relation between labels. Thus a non-constant threshold function is required for solving multi-label classification problems effectively.

5.2 Proposed Method for Threshold Calculation

The associated label set for X is determined by a threshold function $t(x)$, i.e. $Y = \{j | c_j > t(x), j \in \mathcal{Y}\}$ where c_j $(j = 1, 2, ..., Q)$ are the actual outputs and Q is number of distinct labels. Threshold value t for an instance is modelled by a linear function $t(x) = \alpha^T c(x) + \beta$, where $c(x) = (c_1(x), c_2(x), ..., c_Q(x))$ is the Q-dimensional vector whose j-th component corresponds to the actual output of the trained network on x on the j-th class. α is a Q-dimensional vector and β is a scalar. For each multi-label training example (X_i, Y_i) $(i = 1, 2, \ldots, N)$, let $c(X_i) = (c_1^i, c_2^i, ..., c_Q^i)$ and set the target values $t(X_i)$ as:

$$t(X_i) = \arg\min_t (|\{k | k \in Y_i,\ c_k^i < t\}| + |\{l | l \in \bar{Y}_i,\ c_l^i \geq t\}|) \tag{5.1}$$

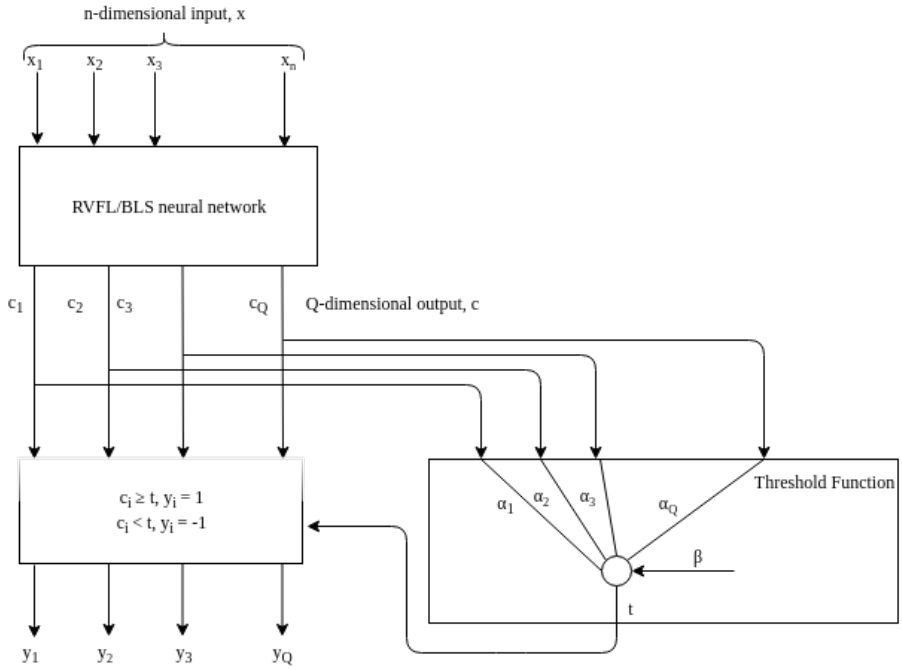

FIGURE 5.1: Proposed Design

Algorithm for determining threshold value, t for each data instance, X is discussed below.

Algorithm for Determining Threshold Value

Input : Predicted output vector on sample x, c and actual label vector for sample x, y
Output: Threshold value t

1. **Sort** c in increasing order and accordingly permute y;
2. $p \leftarrow [c_1 - 0.2|c|c_Q + 0.2]$;
3. $count_1 \leftarrow 0$;
4. $count_{-1} \leftarrow$ count of -1 in y;
5. $\mathcal{E} \leftarrow \infty$;
6. **for** $i \leftarrow 1; i \leq Q + 1$ **do**
7. **if** $\mathcal{E} > count_1 + count_{-1}$ **then**
8. $\mathcal{E} \leftarrow count_1 + count_{-1}$;
9. **t** $\leftarrow \frac{(p_i + p_{i+1})}{2}$;

10. **end if**
11. **if** $i \leq Q$ **then**
12. **if** $y_i = 1$ **then**
13. $count_1 \leftarrow count_1 + 1$;
14. **else**
15. $count_{-1} \leftarrow count_{-1} - 1$;
16. **end if**
17. **end if**
18. **end for**

After obtaining the threshold value t, the parameters of the threshold function can be learned through solving the following matrix equation:

$$\Phi\gamma = t \tag{5.2}$$

Here matrix Φ has dimensions $N \times (Q+1)$ whose i-th row is (c_1^i, c_2^i, ..., c_Q^i, 1), γ is the $(Q+1)$ dimensional vector (α, β) and t is the N-dimensional vector $(t(x_1), t(x_2), ...,$
$t(x_N))$. Moore-Penrose pseudo inverse(4.2) is then applied to find the solution of (5.2).

$$\gamma = \Phi^{\dagger} t \tag{5.3}$$

When a test instance x is given, it is firstly fed to the trained network to get the output vector $c(x)$. After that, the threshold value for x is computed via $t(x) = \alpha^T c(x) + \beta$.

Chapter 6

Experiments

6.1 Datasets

In this section, the performance of the algorithms are tested on 4 different multi-label datasets, which covers text, image, genetic and sound fields.

TABLE 6.1: Multilabel Datasets

Dataset	Domain	Instances	Attributes	Labels
Yeast	Biology	2417	103	14
Emotions	Music	593	72	6
Scenery	Image	2407	294	6
Reuters	Text	2000	243	7

6.2 Parameter Values

RVFL neural network consists of three user defined input parameters C(the trade-off parameter), P(number of enhancement nodes) and μ(used as a range for the assignment of random weight and bias values). Optimal results on Yeast dataset were observed when $C = 0.1$, $P = 50$ and $\mu = 1$.
BLS neural network consists of five user defined input parameters N_1(number of feature nodes per window), N_2(number of windows of feature nodes), N_3(number of enhancement nodes), S(shrinkage parameter for enhancement nodes), C(regularization parameter for sparse regularization). Optimal results on Yeast dataset were observed when $S = 0.1$, $C = 0.1$, $N_1 = 10$, $N_2 = 10$ and $N_3 = 50$.

6.3 Experimental Setup

All the experiments[1] have been conducted on MATLAB 2016a in Windows 7 (64 bit) environment with 64 GB RAM, 3.00 GHz Intel Xeon processor. Z-score normalization has been performed on each dataset before training and testing.

6.4 Results and Discussion

The above described perfomance metrices were calculated using the 4 datasets to evalute RVFL and BLS neural networks. We compared the outputs of RVFL and BLS neural networks with the results obtained from ML-Backpropagation, ML-KNN and ML-SVM model on the Yeast dataset. We have used K-fold cross validation method(K=10)[12] on the datasets for calculating the results.

6.4.1 Comparison between different models on the Yeast dataset

TABLE 6.2: Multi-label Models on Yeast Dataset

Evaluation Metrics	ML-BLS	ML-RVFL	ML-BP	ML-KNN	ML-SVM
Training Time(sec)	0.375	0.342	$1.013x10^4$	**0.209**	$2.780x10^4$
Hamming Loss	0.199	**0.197**	0.206	0.197	0.207
One-Error	**0.222**	0.223	0.233	0.239	0.243
Coverage	6.462	6.365	6.421	**6.302**	7.090
Ranking Loss	0.171	**0.167**	0.171	0.168	0.195
Average Precision	**0.762**	0.762	0.756	0.761	0.750

The training time of RVFL and BLS neural networks are very small as compared to the backpropagation and the SVM models as RVFL and BLS neural networks are non-iterative. The other parameters output from our models are also comparable to the previous used algorithms. The training time of the KNN model is slightly smaller than our models because the Yeast dataset is a smaller dataset with only 2000 entries. For bigger dataset, the training time of KNN model is expected to increase quadratically while our models' training time is expected to increase linearly.

TABLE 6.3: RVFL neural network on different Multi-label datasets

Evaluation Metrics	Yeast	Emotion	Reuters	Scenery
Training Time(sec)	0.342	0.089	0.328	0.350
Hamming Loss	0.197	0.202	0.086	0.193
One-Error	0.223	0.283	0.232	0.373
Coverage	6.365	1.792	0.847	1.088
Ranking Loss	0.167	0.166	0.110	0.202
Average Precision	0.762	0.797	0.841	0.756

6.4.2 Result obtained by the RVFL neural network on different datasets

The results obtained by RVFL neural network on different datasets are also comparable to the other state-of-art algorithms. The precision score of RVFL neural network suffers a little compared to backpropagation model but the training time gives a great advantage for fast learning.

6.4.3 Result obtained by the BLS neural network on different datasets

TABLE 6.4: BLS neural network on different Multi-label datasets

Evaluation Metrics	Yeast	Emotion	Reuters	Scenery
Training Time(sec)	0.375	0.261	0.437	0.421
Hamming Loss	0.199	0.232	0.075	0.199
One-Error	0.222	0.344	0.219	0.375
Coverage	6.462	2.014	0.783	1.105
Ranking Loss	0.171	0.208	0.100	0.206
Average Precision	0.762	0.755	0.851	0.751

The results obtained by the BLS neural network on different dataset is nearly same as the results obtained by the RVFL neural network. The main advantage of the BLS neural network over RVFL neural network can be seen when using a large dataset and also where the model needs to be dynamically updated such as adding extra enhancement nodes. BLS neural network is more applicable in the real life scenarios where we need a online learning algorithm which can easily learn on the new dataset without retraining from begining.

[1]All presented results in this report are reproducible. Codes with datasets can be produced on request.

Chapter 7

Conclusion and Future Work

The objectives of this project were to

- Introduce a new and novel algorithmic approach for multi-label classification problems.
- Prove its correctness.
- Evaluate the performance of the proposed algorithmic approach against existing approaches(ML-SVM, ML-KNN and ML-Backpropagation).

In this project, we have proposed a new approach of multi-label classification by extending RVFL and BLS neural networks by applying the threshold function, computed during the training phase, on the results obtained by the network on the test instance. The results obtained were satisfactory and were ahead from most of the other approaches used for solving the problem.

Multi-label classification is a active field of research and developement. Many new models are proposed for tackling this problem. Future work and further advancement in our model can be done by incorporating the dynamic updation of the BLS model when a new instance is brought in the training phase and also increasing the enhancement nodes.

Bibliography

[1] Min-Ling Zhang and Zhi-Hua Zhou. "Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization". In: *IEEE Transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1338–1351. ISSN: 1041-4347. DOI: 10.1109/TKDE.2006.162.

[2] D. Fu, B. Zhou, and J. Hu. "Improving SVM based multi-label classification by using label relationship". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015, pp. 1–6. DOI: 10.1109/IJCNN.2015.7280497.

[3] Min-Ling Zhang and Zhi-Hua Zhou. "A k-nearest neighbor based algorithm for multi-label classification". In: *2005 IEEE International Conference on Granular Computing*. Vol. 2. 2005, 718–721 Vol. 2. DOI: 10.1109/GRC.2005.1547385.

[4] T. Kopinski et al. "A pragmatic approach to multi-class classification". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280768.

[5] J. Read, A. Puurula, and A. Bifet. "Multi-label Classification with Meta-Labels". In: *2014 IEEE International Conference on Data Mining*. 2014, pp. 941–946. DOI: 10.1109/ICDM.2014.38.

[6] P. Trajdos and M. Kurzynski. "Multi-label Stream Classification Using Extended Binary Relevance Model". In: *2015 IEEE Trustcom/BigDataSE/ISPA*. Vol. 2. 2015, pp. 205–210. DOI: 10.1109/Trustcom.2015.584.

[7] G. Nasierding and A. Z. Kouzani. "Comparative evaluation of multi-label classification methods". In: *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*. 2012, pp. 679–683. DOI: 10.1109/FSKD.2012.6234347.

[8] Z. Yu et al. "An Improved Classifier Chain Algorithm for Multi-label Classification of Big Data Analysis". In: *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*. 2015, pp. 1298–1301. DOI: 10.1109/HPCC-CSS-ICESS.2015.240.

[9] Dirk Husmeier. "Random Vector Functional Link (RVFL) Networks". In: *Neural Networks for Conditional Probability Estimation: Forecasting Beyond Point Predictions*. London: Springer London, 1999, pp. 87–97. ISBN: 978-1-4471-0847-4. DOI: 10.1007/978-1-4471-0847-4_6. URL: https://doi.org/10.1007/978-1-4471-0847-4_6.

[10] C. L. P. Chen and Z. Liu. "Broad Learning System: An Effective and Efficient Incremental Learning System Without the Need for Deep Architecture". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.1 (2018), pp. 10–24. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2017.2716952.

[11] G. Nasierding and A. Z. Kouzani. "Comparative evaluation of multi-label classification methods". In: *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*. 2012, pp. 679–683. DOI: 10.1109/FSKD.2012.6234347.

[12] S. Yadav and S. Shukla. "Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification". In: *2016 IEEE 6th International Conference on Advanced Computing (IACC)*. 2016, pp. 78–83. DOI: 10.1109/IACC.2016.25.