MULTI LABEL CLASSIFICATION ALGORITHM BASED ON KERNEL EXTREME LEARNING MACHINE

A project report,

submitted to **Dr. Vinal Patel**

in the subject of Neurocomputing Architecture

by

Amal Shaji (2017BCS-010)
Chirag Jindal (2017BCS-013)
Manvi Gupta (2017BCS-017)
Sandarbh Yadav (2017BCS-027)
Saumya Gupta (2017BCS-028)
Vaibhav Garg (2017BCS-038)

विश्वजीवनामृतं ज्ञानम्

ABV INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT, GWALIOR-474 015

TABLE OF CONTENTS

CHAPTER 1

Introduction

Classification is a widely used data analysis technology having implementation in various fields such as machine learning, pattern recognition, data mining, etc. In the traditional approach of classification learning, only a single category label with a significant semantic meaning is associated with each sample which is also known as "single label classification". Many different efficient and accurate algorithms like decision trees, SVM, ANN and naive Bayes algorithm have been proposed for this type of classification. They are widely used in real life and significant progress has been made in these algorithms. But due to the increase in the data available having numerous semantic information, this classification having only a single category label for a sample does not describe the real objects precisely. The objects are complex and ambiguous and multiple category labels can be linked with them at once. For example, a song can be labeled with "dance", "pop", "Taylor Swift" or any natural scene can be associated with "river", "sun", "clouds". The practical use of multi-label classification is in various domains such as semantic annotation of images and video, media annotation and biological proteins function classification. It falls in the domain of supervised learning.

In the real world, multi-label classification makes more sense than single label classification in terms of the rules and characteristics. Also, multi-label classification is a more comprehensive and general version of the single-label classification. But it also brings out many struggles and challenges to the problem, mainly in having a large subset space of output labels. For instance, if we have a data set having 20 distinct category labels, the amount of candidate label subsets resembling an unseen sample gives a million labels. From this number having an exponential scale, it is very difficult to choose the correct label subset. So, to design an accurate and efficient multi-label classification algorithm is of high significance and application in the field of machine learning.

A multi-label classification algorithm ML-KELM [1] based on kernel extreme learning machine is proposed. To accomplish the conversion of network real-valued output to binary vector, a threshold function is designed. The self-adaptive training set is used to collect the parameters of the threshold function. This algorithm is more stable for multi label classification problems when compared to the existing ELM algorithms. When this algorithm was compared with other algorithms using different experiments in six different data sets, the time required for training is less when the data set is large for the ML-KELM algorithm.

CHAPTER 2

Related Work

2.1 Multi Label Classification

Generalizing a multi class classification is known as multi label classification. Multi class classification is the single label classification that categorizes instances into exactly one of more than two classes. But, there is no limit to how many classes a sample can classify a multi label problem in multi label classification. To overcome multi label classification, the following are the primary methods:

2.1.1 Problem Transformation

This approach divides the learning process into one or more single-label classification tasks. For example, the binary relevance algorithm broke down a multi label classification problem into a sequence of separate binary classification problems, with each label equivalent to a binary classifier, and then incorporated all binary classifiers to get the expected label subsets for a test sample. It is the simplest and most effective approach, but the only disadvantage is that it ignores label correlation so each target variable is treated separately. As a result, the algorithm's generalisation efficiency is low.

The first classifier in a classifier chain is trained solely on the input data, with subsequent classifiers being trained on the input space as well as all other classifiers in the chain. This is close to binary significance, with the exception that it creates chains to maintain label correlation. We transform the problem into a multi class problem in label power set, and one multi class classifier is trained on all specific label combinations contained in the training results. The main drawback is that as the amount of training data grows, the number of classes grows as well. As a consequence, increasing the model complexity would reduce accuracy.

2.1.2 Algorithm Adaptation

Adapted algorithm, as the name implies, adapts the algorithm to perform multi label classification explicitly rather than translating the query into separate subsets. For example:

- Using the highest posteriori likelihood of the k-nearest neighbour samples, the ML-KNN Algorithm obtained the estimation label subset for an unseen sample.
- Rank-SVM Algorithm works by building SVM classifiers for each label based on the ranking loss function. As a result, a more precise association between samples and labels may be discovered.

These sort of algorithms works very smooth for relatively less number of data sets but it will take very long time in training large scale data sets.

2.1.3 Ensemble Approaches

This type of classifier usually ensemble the problem transformation or algorithm adaptation to solve the problem of multi-label classification. For example, Classifier Chains Algorithm make a classifier chain out of labels and each chain then will be treat as a label. The link between chains (labels) are calculated randomly because chains in classifier are randomly placed. The performance drastically degrades if previous chains are not that good.

The work presented in this report used a kernel extreme learning machine based "algorithm adaptation" multi-label classifier.

2.2 ELM Based Work

Huang et al [2] suggested the Extreme Learning Machine (ELM), which has the advantages of high speed and reliability due to its ability to skip the time-consuming iterative learning process. ELM is a feed-forward neural network with a single hidden layer. ELM will ensure that the network has strong generalisation efficiency while still avoiding the issues associated with conventional feed-forward networks' iterative learning. Sun et al [3] and Meng et al [4] tried using ELM as a way to solve multi-label classification problem.
Following points need to follow for using ELM for multi-label classification problem.

1. Input pre-processing - converting bipolar from unipolar representation. (1: relevant to input; -1: not relevant to input)

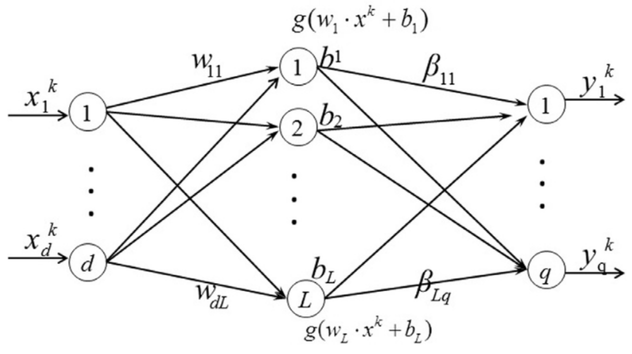

Figure 2.1: ELM Network Structure Diagram

2. Network Structure - Obtain it from the training data as shown in Fig 2.1

Basically, ELM follows 3 layers architecture having 1 input layer, 1 hidden layer, and 1 output layer. Hidden layer have L neurons and g(x) denotes the activation function. Weights between input and hidden layers are assigned randomly and then we calculate the output for each neuron using activation function. Thereafter, H matrix is being calculated. It is in the form of m * L matrix where m is the size of training set and a label is attached with this. Now, the output weights i.e. weight between hidden and output layer is calculated using multiplication of pseudo inverse of H and Y. It will be a L * q matrix. Therefore, m * q will the output size of the ELM bipolar f(x). This approach will vastly increase the neural network's generalisation potential and learning speed, as well as avoiding the complicated iterative operation. Step wise ELM algorithm to solve multi-label classification is shown below:

```
Algorithm 1 ELM Algorithm.
Input:
    training set $\mathcal{D}$; testing set $\mathcal{D}'$;
Output:
    ELM network's $finaloutputs$ for testing set $\mathcal{D}'$;
 1: Initialize hidden layer neuron number $L$ , activation function $g$,
    cost parameter $C$;
 2: Pre-process: target values of training set $\mathcal{D}$ are converted to
    bipolar representation;
 3: for training set $\mathcal{D}$ do
 4:   Assign input weight $w$ and hidden layer bias $b$ randomly;
 5:   Compute output matrix $H = g(w \cdot x + b)$ of hidden layer;
 6:   Compute output weights $\beta$ according to Eq.(3);
 7: end for
 8: for testing set $\mathcal{D}'$ do
 9:   Calculate outputs $f(x)$ according to Eq.(4);
10:   Post-process: compare $f(x)$ with a threshold value and
      obtain multi-label identification $finalouputs$ according to
      Eq.(5);
11: end for
12: return $finaloutputs$
```

CHAPTER 3

ML-KELM

The stablity of ELM's network is dented by the randomness of its hidden layers as it results in oscillation of hidden layer output matrix. Also, it lacks an appropriate method for threshold learning. But the importance of hidden layer is that it provide explicit nonlinear mapping making it possible to map input and output samples into different spaces which could linearly be separated. Using a kernel function can be solution for this issue. It could be used to map from high-dimensional to low-dimensional feature vectors with the help of inner product operation. An attempt to combine these two concepts have been done in the ML-KELM. Using kernel extreme learning machine for solving the multi-label classification problem. And comparison are made between the ELM and ML-KELM which shows that the described method is stable as well as is safer and quicker.

3.1 ML-KELM Algorithm

Huang et al [5] replaced the hidden layer mapping in ELMs by kernel function K(u,v) to provide stability to the network. Same method is applied to multi label classification problem. ML-KELM network do not necessarily require the hidden layer output nor there is need to set number of neurons in the layer. We just need to choose the kernel function K(u, v) which replaces the HH^T (i,j) by K(x$_i$, y$_j$). The kernel matrix can be represented using the equation:

$$\Omega_{ELMi,j} = h(x_i) \cdot h|(x_j) = K(x_i, x_j)$$

The output of ML-KELM algorithm can be represented by equation:

$$f(x) = HH^T\left(\frac{I}{C} + HH^T\right)^{-1} Y = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_m) \end{bmatrix}^T \left(\frac{I}{C} + \Omega_{ELM}\right)^{-1} Y$$

ML-KELM has a better ability to classify due to use of kernel functions for mapping. Choosing the appropriate kernel is the key. Once it is done, ML-KELM becomes very stable.

3.2 Threshold Function Setting

The network output of the ML-KELM is a vector of dimension q. The component of this vector correspond to the categories needed to be classified where the total number of categories is q as well.

In multi label classifier, for any dimension input, the given output should be q-dimensional vectors containing -1 or 1. For conversion to desired binarized vector a threshold function is required. This threshold function works as a boundary which is used to classify the output into relevant and irrelevant labels using the data provided as the training set. Using constant value to generate this boundary is not possible as it should be done using the complete training data, should be dynamic and should be self-adaptive to the changes. Linear function is used as the threshold and threshold setting method in Rank-SVM to modify it. The reason to choose linear function for determining the threshold is that its implementation is simple and we need to tune very few parameters.

A linear function t(x) is defined as the sum of dot product of f(x) and a^* added to b^*. Here a^* represents the input vector and b^* is the offset vector. With this now the problem remains to find the optimal a^* and b^* which is able to separate the relevant and irrelevant labels by the output from the ML-KELM network. This factor how well the boundary value divides the labels can be denoted by $s(x_i)$. Whose calculation is given in equation:

$$s(x_i) = \arg\min_{r \in R}(|\{y_j | y_j \in Y_i, f_j(x_i) \le r\}| + |\{y_k | y_k \notin Y_i, f_k(x_i) \ge r\}|)$$

To find optimal approximation we have used candidate discrete set. On applying least square method we obtain equation:

$$\min_{\{a^*,b^*\}} \sum_{i=1}^{m} (\langle a^*, f(x_i)\rangle + b^* - s(x_i))^2$$

Objective of this equation is to minimise the sum of square of the difference of output and input. For any training set, the threshold can be calculated using the equation:

$$t = [[f(x), E_{m' \times 1}][a^*, b^*]^T]^T$$

Where threshold is of dimension m' same as the training data set. And E is matrix with m'X1 with all elements as 1.

Then we use this information to find the final output using the training equation:

$$finaloutputs(k,i) = \begin{cases} 1 & f_k(x_i) \geq t(k) \\ -1 & f_k(x_i) < t(k) \end{cases} i = 1, \ldots, m', \quad k = 1, \ldots, q$$

We can see the effect of the threshold in this equation as well how it changes the final output vector according to the threshold. Detailed processes of the ML-KELM algorithm are shown below:

Algorithm 2 ML-KELM Algorithm.

```
Input:
    training set $\mathcal{D}$; testing set $\mathcal{D}'$
Output:
    ML-KELM network's $finaloutputs$ for testing set $\mathcal{D}'$
 1: Initialize kernel function parameter $\sigma$ , cost parameter $C$
 2: Pre-process: target values of training set $\mathcal{D}$ are converted to
    bipolar representation
 3: for training set $\mathcal{D}$ do
 4:   Compute kernel matrix $\Omega_{ELM}$
 5:   Compute ML-KELM's network structure $(\frac{I}{C} + \Omega_{ELM})^{-1}Y$
 6:   Compute threshold parameter $a^*$, $b^*$ by training set $\mathcal{D}$

 7: end for
 8: for testing set $\mathcal{D}'$ do
 9:   Calculate outputs $f(x)$
10:   Construct threshold function $t$ based on parameter $a^*$, $b^*$

11:   Post-process: compare $f(x)$ with threshold $t$ to obtain
      $finaloutputs$
12: end for
13: return $finaloutputs$
```

CHAPTER 4

Results

4.1 Performance Metrics

For a testing set, D' = $\{(x_i, Y_i) \mid 1 \leq i \leq m'\}$, the following sample based performance metrics are used:

4.1.1 Hamming Loss

Hamming loss is a measure of inconsistency between the predictive label sets and the actual label sets of samples. It is given by

$$hloss_s(h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{q} \mid h(x_i) \Delta Y_i \mid$$

where Δ represents symmetric difference between two label sets.

4.1.2 One Error

One error is a measure of proportion of samples whose highest scoring label is not the actual label. It is given by

$$one - error_s(h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{q} \left[\left[[\arg \max_{y \in \mathcal{Y}} f(x_i, y)] \notin Y_i \right]\right]$$

where for a predicate π, value of $||\pi||$ is 1 if predicate is satisfied else value is 0.

4.1.3 Coverage

Coverage is a measure of how much the average needs to go down the lists of labels in order to cover all the proper labels of the instance, arranging the outputs f(x) in descending order. It is given by

$$coverage_s(h) = \frac{1}{m'} \sum_{i=1}^{m'} \max_{y \in Y_i} rank_f(x_i, y) - 1$$

where rank$_f$(., .) is the ranking function corresponding to real valued function f(., .).

4.1.4 Ranking Loss

Ranking loss describes the average proportion of label pairs that are reversely ordered for the instance. It is given by

$$rloss_s(h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{|Y_i||\overline{Y_i}|} \mid \{(y', y'') | f(x_i, y') \leq f(x_i, y''), (y', y'') \in Y_i \times \overline{Y_i}\} \mid$$

where $\overline{Y}$ denotes the complementary set of Y.

4.1.5 Average Precision

Average precision is a measure of average of precisions of the predictive multi labels. It is given by

$$avgprec_s(h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{|Y_i|} \sum_{y \in Y_i} \frac{\left|\{y' | rank_f(x, y') \leq rank_f(x, y), y' \in Y_i\}\right|}{rank_f(x_i, y)}$$

Note : Smaller value is desirable for hamming loss, one error, coverage and ranking loss. Bigger value is desirable for average precision.

4.2 Data Sets Used

The following data sets are used for experimental analysis:

- Emotions data set
- Yeast gene function data set
- Natural scene data set
- Corel16k data set
- rcv1v2 data set

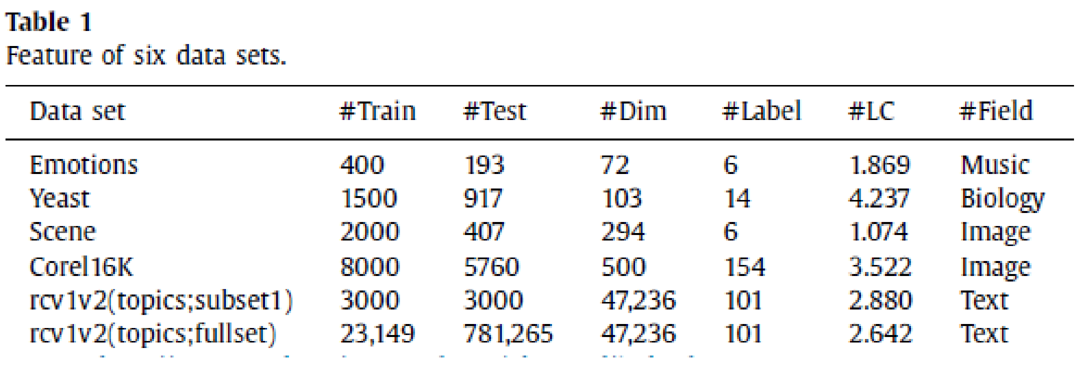
Table 1
Feature of six data sets.

Data set	#Train	#Test	#Dim	#Label	#LC	#Field
Emotions	400	193	72	6	1.869	Music
Yeast	1500	917	103	14	4.237	Biology
Scene	2000	407	294	6	1.074	Image
Corel16K	8000	5760	500	154	3.522	Image
rcv1v2(topics;subset1)	3000	3000	47,236	101	2.880	Text
rcv1v2(topics;fullset)	23,149	781,265	47,236	101	2.642	Text

The features of the data sets are summarized in Table 1. #Train denotes the size of training set. #Test denotes the size of test set. #Dim denotes the dimensionality of data set. #Label denotes the number of labels in data set. #LC denotes the label cardinality of data set. #Field denotes the domain fields of data set.

4.3 Comparison of Algorithms

Four state- of-the-art multi label learning algorithms are used for comparative study with ML-KELM: Rank-SVM [6] based on kernel function, ML-KNN [7] algorithm based on k-nearest neighbour concept, Boostexter [8] algorithm based on ensemble learning mechanism and classical Extreme Learning Machine (ELM) [9] algorithm. All experiments are conducted in MATLAB and cross validation is used. The comparative results on different performance metrics are shown in Tables 2 to 7. The best performance metrics are shown in bold. Table 8 discusses the computation time of each multi label learning algorithm, where all experiments are conducted on a machine with 2.5 GHz CPU and 4 GB memory.

Table 2
Test results of five multi-label algorithms on Emotions dataset (mean± std).

	ML-KELM	Rank-SVM	ML-KNN	BoosTexter	ELM
Hamming loss↓	**0.190 ± 0.013**	0.199 ± 0.010	0.194 ± 0.003	0.214 ± 0.006	0.197 ± 0.024
One error↓	**0.249 ± 0.007**	0.253 ± 0.027	0.263 ± 0.009	0.318 ± 0.014	0.260 ± 0.011
Coverage↓	0.297 ± 0.009	**0.295 ± 0.009**	0.300 ± 0.002	0.302 ± 0.007	0.298 ± 0.013
Ranking loss↓	0.166 ± 0.002	0.173 ± 0.008	**0.163 ± 0.007**	0.175 ± 0.009	0.169 ± 0.004
Average precision↑	**0.819 ± 0.011**	0.807 ± 0.008	0.799 ± 0.002	0.796 ± 0.004	0.808 ± 0.012

Table 3
Test results of five multi-label algorithms on Yeast dataset (mean± std).

	ML-KELM	Rank-SVM	ML-KNN	BoosTexter	ELM
Hamming loss↓	0.201 ± 0.011	0.207 ± 0.006	**0.194 ± 0.011**	0.220 ± 0.005	0.200 ± 0.023
One error↓	**0.193 ± 0.017**	0.243 ± 0.017	0.228 ± 0.029	0.278 ± 0.014	0.199 ± 0.016
Coverage↓	0.459 ± 0.013	0.514 ± 0.038	**0.447 ± 0.004**	0.456 ± 0.024	0.475 ± 0.015
Ranking loss↓	**0.167 ± 0.023**	0.195 ± 0.011	**0.167 ± 0.004**	0.186 ± 0.002	0.178 ± 0.007
Average precision↑	0.728 ± 0.019	0.749 ± 0.009	**0.765 ± 0.007**	0.737 ± 0.014	0.725 ± 0.011

Table 4
Test results of five multi-label algorithms on Scene dataset (mean± std).

	ML-KELM	Rank-SVM	ML-KNN	BoosTexter	ELM
Hamming loss↓	0.085 ± 0.006	0.104 ± 0.006	**0.084 ± 0.008**	0.096 ± 0.010	0.085 ± 0.009
One error↓	**0.198 ± 0.026**	0.250 ± 0.027	0.219 ± 0.029	0.226 ± 0.034	0.217 ± 0.023
Coverage↓	**0.078 ± 0.009**	0.089 ± 0.009	**0.078 ± 0.010**	**0.091 ± 0.008**	0.088 ± 0.024
Ranking loss↓	**0.066 ± 0.007**	0.089 ± 0.011	0.076 ± 0.012	0.135 ± 0.013	0.135 ± 0.01
Average precision↑	**0.885 ± 0.011**	0.849 ± 0.016	0.869 ± 0.017	0.852 ± 0.016	0.847 ± 0.016

Table 5
Test results of five multi-label algorithms on Corel16K dataset (mean± std).

	ML-KELM	Rank-SVM	ML-KNN	BoosTexter	ELM
Hamming loss↓	**0.018 ± 0.006**	0.019 ± 0.003	**0.018 ± 0.001**	**0.018 ± 0.003**	**0.018 ± 0.012**
One error↓	**0.697 ± 0.004**	0.846 ± 0.013	0.740 ± 0.006	0.861 ± 0.003	**0.697 ± 0.009**
Coverage↓	**0.315 ± 0.006**	0.321 ± 0.012	0.332 ± 0.002	0.357 ± 0.008	0.321 ± 0.007
Ranking loss↓	**0.155 ± 0.007**	0.176 ± 0.004	0.168 ± 0.002	0.179 ± 0.004	0.158 ± 0.006
Average precision↑	0.273 ± 0.012	0.194 ± 0.006	0.279 ± 0.001	**0.288 ± 0.001**	0.258 ± 0.008

Table 6
Test results of five multi-label algorithms on rcvlv2(subset1) dataset (mean± std).

	ML-KELM	Rank-SVM	ML-KNN	BoosTexter	ELM
Hamming loss↓	**0.028 ± 0.006**	0.031 ± 0.004	**0.028 ± 0.003**	0.031 ± 0.009	0.029 ± 0.019
One error↓	**0.435 ± 0.013**	0.608 ± 0.007	0.510 ± 0.004	0.603 ± 0.011	0.594 ± 0.021
Coverage↓	**0.169 ± 0.022**	0.387 ± 0.012	0.188 ± 0.007	0.458 ± 0.012	0.346 ± 0.005
Ranking loss↓	0.096 ± 0.007	0.273 ± 0.011	**0.089 ± 0.006**	0.142 ± 0.008	0.172 ± 0.004
Average precision↑	**0.580 ± 0.021**	0.415 ± 0.003	0.513 ± 0.016	0.399 ± 0.006	0.524 ± 0.008

Table 7
Test results of five multi-label algorithms on rcvlv2(fullset) dataset (mean± std).

	ML-KELM	Rank-SVM	ML-KNN	BoosTexter	ELM
Hamming loss↓	**0.058 ± 0.003**	0.071 ± 0.006	0.065 ± 0.001	0.063 ± 0.002	0.066 ± 0.016
One error↓	0.342 ± 0.006	0.504 ± 0.022	0.347 ± 0.008	**0.340 ± 0.009**	0.519 ± 0.007
Coverage↓	**0.128 ± 0.009**	0.142 ± 0.006	0.129 ± 0.004	0.129 ± 0.012	0.137 ± 0.013
Ranking loss↓	**0.063 ± 0.002**	0.108 ± 0.003	0.064 ± 0.001	0.068 ± 0.006	0.094 ± 0.004
Average precision↑	0.813 ± 0.011	0.796 ± 0.005	0.797 ± 0.008	**0.832 ± 0.021**	0.801 ± 0.013

Table 8
Time cost of each compared algorithm on both data sets, where time is measured in seconds (mean value).

	ML-KELM	Rank-SVM	ML-KNN	BoosTexter	ELM
Training phase(Emotions)	0.41	1096	0.21	345	0.19
Testing phase(Emotions)	0.04	1.11	0.18	1.57	0.04
Training phase(Yeast)	0.49	3058	0.32	428	0.27
Testing phase(Yeast)	0.07	0.10	0.34	0.97	0.06
Training phase(Scene)	0.06	543	0.31	168	0.05
Testing phase(Scene)	0.10	1.24	0.16	1.88	0.08
Training phase(Corel16K)	38.16	5974	7.38	1021.80	19.65
Testing phase(Corel16K)	0.59	8.66	1.14	9.37	0.52
Training phase rcv1v2(subset1)	0.88	1944	1.17	603.20	0.73
Testing phase rcv1v2(subset1)	0.35	3.02	0.78	2.65	0.23
Training phase rcv1v2(fullset)	95.33	24,377	606.25	4154	65.34
Testing phase rcv1v2(fullset)	164.38	525	216.47	439.20	172.44

4.4 Experiment Result Analysis

The ELM algorithm is tested with different number of hidden layer nodes and an optimal value is chosen. This optimal value is determined to be 60 in Emotions data set, 100 in Yeast data set, 85 in Scene data set, 250 in Corel16K data set, 120 in rcvlv2 (subset) data set and 600 in rcvlv2 (fullset) data set. The process of determining optimal number of hidden layer nodes is time consuming. The results of the experiments are not stable due to the random setting of input weights.

From Table 2 of Emotions data set, we can observe that ML-KELM performs best in “Hamming loss”, “One error” and “Average precision”. In Table 3 of Yeast data set, the performance of ML-KNN algorithm is optimal in “One error” and “Ranking loss”. In Table 4 of Scene data set, ML-KELM algorithm gains the best performance in “One error”, “Coverage”, “Ranking loss” and “Average precision”. In Table 5 of Corel16K dataset, ML-KELM algorithm has best performance in “Hammingloss”, “One error”, “Coverage” and “Ranking loss”. On the rcvlv2 (subset1) data set of Table 6, ML-KELM is optimal in “Hamming loss”, “One error”, “Coverage” and “Average Precision”. In Table 7 of rcvlv2 (full set) data set, ML-KELM algorithm has optimal performance in “Hamming loss”, “Coverage” and “Ranking loss”.

As can be seen from Table 8 , ML-KELM consumes slightly more time than ELM algorithm. This is because it spends more time to calculate kernel matrix. But ML-KELM runs much faster than the other three algorithms. For samples smaller than 2000, the training time is usually less than 1 second and the training time of 30000 samples is only about 2 minutes. Except the ELM algorithm, the training time of ML-KELM algorithm is far less than the other three algorithms. Testing time of ML-KELM algorithm is also far less than the other three algorithms. In the testing phase, ML-KELM algorithm spends very less time, because it only needs to perform inner products and matrix multiplications.

CHAPTER 5

Conclusion

The main purpose of multi-label classification learning is to allocate a set of suitable labels to the unseen objects. It is very important when we are studying the polysemy object modeling. A multi-label classification algorithm ML-KELM is designed based on the system of kernel extreme learning machine. Different types of datasets were used to test the algorithm like the Scene dataset, the Yeast dataset, the Emotions dataset and the large scale text data sets. On performing various experiments for comparison it was found out that ML-KELM gives better generalization performance, needs less parameter adjustment and has faster convergence speed. It is also practically significant in the research of big data as it performs stably and better on large data sets.

For the next step of research, to further enhance the performance of the algorithm sparse coding can be used. Also, the algorithm can be designed by using optimal kernel parameters by sensitivity analysis of parameter setting.

REFERENCES

[1] F. Luo, W. Guo, Y. Yu, and G. Chen, “A multi-label classification algorithm based on kernel extreme learning machine,” *Neurocomputing*, vol. 260, pp. 313–320, 2017.

[2] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[3] X. Sun, J. Wang, C. Jiang, J. Xu, J. Feng, S.-S. Chen, and F. He, “Elm-ml: study on multi-label classification using extreme learning machine,” in *Proceedings of ELM-2015 Volume 2*, pp. 107–116, Springer, 2016.

[4] M. J. Er, R. Venkatesan, and N. Wang, “A high speed multi-label classifier based on extreme learning machines,” in *Proceedings of ELM-2015 Volume 2*, pp. 437–454, Springer, 2016.

[5] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme learning machine for regression and multiclass classification,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.

[6] A. Elisseeff, J. Weston, *et al.*, “A kernel method for multi-labelled classification.,” in *NIPS*, vol. 14, pp. 681–687, 2001.

[7] M.-L. Zhang and Z.-H. Zhou, “Ml-knn: A lazy learning approach to multi-label learning,” *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[8] R. E. Schapire and Y. Singer, “Boostexter: A boosting-based system for text categorization,” *Machine learning*, vol. 39, no. 2, pp. 135–168, 2000.

[9] R. Venkatesan and M. J. Er, “Multi-label classification method based on extreme learning machines,” in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, pp. 619–624, IEEE, 2014.