Multi-label Learning Based on Kernel Extreme Learning Machine

Fangfang Luo, Wenzhong Guo, Fangwan Huang and Guolong Chen
College of Mathematics and Computer Science, Fuzhou University,
3501165 Fuzhou, China

ABSTRACT

In recent years, with the increase of data scale, multi-label learning with large scale class labels has turned out to be the research hotspots. Due to the huge solution space, the problem becomes more complex. Therefore, we propose a multi-label algorithm based on kernel learning machine in this paper. Besides, the Cholesky matrix decomposition inverse method is adopted to calculate the network output weight of the kernel extreme learning machine. In particular, in terms of large matrix inverse problem, the large matrix is divided into small matrices for parallel computation through using matrix block method. Compared with several state-of-the-art algorithms on several benchmark data sets, results of the experiments show that the proposed algorithm makes a better performance with large scale class labels.

INTRODUCTION

Multi-label learning aims to assign relevant label set to an instance, which can thus provide technical support for information retrieval. For example, a natural scene can hold three labels simultaneously, respectively, "mountain", "sunset", "tree". Besides, a melody can be labeled with "Orchestra", "dance" and "Strauss Johann".

Extreme learning machine (ELM) has been applied to multi-label learning with small scale datasets, and good results have been obtained [1]. However, with the increase of label set size, the ELM’s hidden layer output matrix inversion problem becomes bottleneck of large-scale data applications. Huang et al. [2] take the kernel function into the ELM instead of hidden layer, and construct kernel extreme learning machine with the least square solution. Regarding positive definiteness of kernel ELM output weight matrix, Cholesky decomposition [3] is adopted to solve hidden layer output matrix inversion problem. Cholesky method is successfully applied to time-varying or nonstationary systems, while incremental updating of the system is implemented [4-6]. Inspired by this method, kernel extreme learning machine is applied to multi-label classification problem, which is called KELM-ML. Focusing on the multi output characteristics of multi-label problem, an adaptive threshold mechanism is proposed in this paper. Moreover, in the case of large-scale data sets that cannot directly be loaded in a single PC memory, the Cholesky method is run in a distributed cluster environment in parallel based on the matrix block method. Finally, compared with other state-of-the-art multi-label algorithms, experimental results show that the proposed KELM-ML provides significant competitive multi-label classification performance.
RELATED WORK

There are two ways of multi-label learning. One is the "problem transformation" method and the multi-label problem is transferred to other known learning problem. There exists a method which decomposes the multi-label problem into multiple independent binary classification problems, such as relevance Binary method (BR) [7]. Moreover, algorithms based on "problem transformation" include Pairwise Multi-label Algorithm [8], Efficient Classifier Chains (ECC) [9] and so on.

The second way refers to the "algorithm adaptation" method and the basic idea is to alter conventional supervised learning algorithm to adapt the multi-label classification problem. In addition, the representative algorithms include Boost method [10], ML-KNN method [11], Rank-SVM algorithm [12], ELM [1] and etc.

ELM has the characteristics of high speed and high efficiency, since it can avoid the tedious iterative learning process. The time of computing the Moore-Penrose generalized inverse matrix increase sharply, when the hidden layer output matrix becomes large. One improved approach is to compute the smallest norm of the output weights in ELM. For example, FASTA-ELM uses an extension of forward-backward splitting to compute the smallest norm of the output weights in ELM [13]. CP-DP-ELM adopts recursive orthogonal method to reduce the computational burden [14]. Besides, another approach is to introduce kernel tricks into classifier [15]. The kernel method only involves feature space inner product operation, which shows no relationship with the dimension of the feature space. Therefore, it can effectively avoid the "dimension disaster" problem. This paper attempts to apply the kernel extreme learning machine to multi-label learning, and solves the inverse problem of large matrix by "Cholesky matrix decomposition inversion".

ALGORITHM MODEL

Multi-label classification is a kind of supervised learning. Let $X = R^d$ denote a $d$ dimensional instances space. $Y = \{y_1, y_2, ..., y_q\}$ is a finite set of labels, and the total number of labels space is $q$, $q > 1$. $D = \{(x_i, Y_i) | 1 \leq i \leq m\}$ denotes a training set with $m$ instances and $x_i = (x_{i1}, x_{i2}, ..., x_{id})^T$ is the $i$-th training instance which has label set $Y_i \subseteq Y$ associated. $Y_i = (y_{i1}, y_{i2}, ..., y_{iq}) = \{-1,1\}^q$ is a $q$ dimensional binary vector, where each element is 1 if the label is relevant and -1 otherwise. The task of multi-label classification is to get a classifier $h : X \rightarrow 2^Y$ by training sample instances, which can map an instance to a label set.

The existing ELM structure [1] for solving multi-label problem is shown in Figure 1(a). Besides, the input layer has $d$ neurons, and output layer has $q$ neurons. $L$ is the number of hidden layer neurons. $Y$ is the theoretical output for training set. Let $h(x_i)$ be the hidden layer output vector for the instance $x_i$. $H = [h^T(x_1), ..., h^T(x_m)]^T$ is the hidden layer output matrix for the whole training set. $\beta = [\beta_1, ..., \beta_L]^T$ is hidden layer output weight. Based on ELM theory, $\beta$ can be calculated by $\beta = H^T (I/C + HH^T)^{-1}Y$. The ELM’s output $f(x)$ is the multiplication of hidden
layer output matrix $H$ and hidden layer output weight $\beta$, which can be presented by Eq.(1). Where, $C$ is ridge regression parameter, and $I$ denotes the identity matrix of the adaptive size.

$$f(x) = H\beta = HH^T (I/C + HH^T)^{-1}Y$$  \hspace{1cm} (1)$$

However, in KELM-ML network, there are neither necessity to calculate the hidden layer output nor assign the number of hidden layer nodes, just requiring selecting proper kernel function $K(u, v)$. The structure of KELM-ML is presented in Figure 1(b). $HH^T$ can be replaced by kernel function, which is $HH^T(i, j) = K(x_i, x_j)$. Kernel matrix definition sees Eq.(2):

$$HH^T = h(x_i) \cdot h(x_j) = K(x_i, x_j)$$  \hspace{1.5cm} (2)$$

Let $\alpha = (I/C + HH^T)^{-1}Y$, $\alpha$ can be regarded as the output weights of KELM-ML network. Therefore, the KELM-ML’s output could obtain by Eq. (3). In this paper, the kernel function is the radial basis kernel function $K(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$.

$$f(x_i) = HH^T \left( I/C + HH^T \right)^{-1}Y = \begin{bmatrix} K(x_1, x_1) \\ \vdots \\ K(x_k, x_m) \end{bmatrix}^T \left( I/C + HH^T \right)^{-1}Y$$  \hspace{1.5cm} (3)$$

Figure 1. Network Structure.

The output of the KELM-ML network is a $q$-dimensional real-valued vector, which do not meet the multi-label classification system’s binary output vector requirement. As a result, we need a threshold function $t$ to transfer the KELM-ML network output to binary vector. Refer to references [16], threshold is set as a linear function $t(x) = (a^* \cdot f^*(x) + b^*)$, where $a^*$ is a $q$-dimension vector, $b^*$ is the offset. The least square method can be used to solve this optimization problem as shown in formula (4). For $x_i$, compared the $j$-th neuron output $f_j(x_i)$ in output layer with the $j$-th component of target value $Y_i$, and if $Y_i^{(j)}$ is relevant label, add $f_j(x_i)$ into the relevant set $U_j$ of $x_i$, or else add $f_j(x_i)$ into the irrelevant set $\bar{U}_j$. $s(x_i)$ can be considered as a boundary between relevant set and an irrelevant set. So, $s(x_i)$ can be easily approximated by $(\text{max}(U_j) + \text{min}(U_j))/2$.
\[
\min_{(a^*, b^*)} \sum_{i=1}^{m} (f^*(x_i) - b^* - s(x_i))^2
\]  

(4)

For a test set \( \mathcal{D}' = \{(x_i, Y_i) | 1 \leq i \leq m'\} \) with \( m' \) instances, threshold can be obtained by Eq.(5). Where, \( E \) is a \( m' \times 1 \) dimensional matrix with all element values of 1.

\[
t = [(f(x), E_{w\times1}) [a^*, b^*]^T]
\]  

(5)

As shown in Eq.(5), threshold \( t \) is a \( m' \) dimensional vector associated with testing set. Then, we pass output \( f(x_i) \) through threshold \( t \) to obtain multi-label classification results and threshold process method could be seen in Eq. (6):

\[
\text{final outputs}(k, i) = \begin{cases} 
1 & f_k(x_i) \geq t(k) \\
-1 & f_k(x_i) < t(k)
\end{cases} 
\]

(6)

PARALLEL COMPUTING FOR KELM-ML
Calculation of KELM-ML Output Weight

In Section 3, the output weight of KELM-ML network is \( \alpha = (I/C + HH^T)^{-1}Y \).

Consider using Cholesky matrix decomposition method to reduce the computational complexity. The premise of Cholesky matrix decomposition is that the matrix is a symmetric positive definite matrix.

Proof: \( \therefore A^T = (I/C + HH^T)^T = I/C + HH^T = A \). \( \therefore A \) is a symmetric matrix.

For an KELM-ML network construct by training set which contains \( m \) instances, the dimension of \( HH^T \) is \( m \times m \). Set \( \lambda \) be no-zero \( m \)-dimensional column vector, then:

\[
\lambda^T A \lambda = \lambda^T (I/C + HH^T) \lambda = \lambda^T \lambda/C + \lambda^T HH^T \lambda = \|\lambda\|^2/C + \|H^T \lambda\|^2 > 0
\]

\( \therefore A \) is a positive definite matrix. In summary, \( A \) is a symmetric positive definite matrix.

According to the Cholesky method [2], the matrix \( A \) can be decomposed into \( A = LL^T \), in which the \( L \) is the lower triangular matrix, and the elements in \( L \) are calculated as the Formula (7), where \( i, j = 1, \ldots, m \).

\[
l_{ij} = \begin{cases} 
\sqrt{a_{ij} - \sum_{k=1}^{i-1} l_{ik}^2} & i = j \\
1/l_{jj} (a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}) & i > j
\end{cases}
\]  

(7)

Since \( A = LL^T \), \( A^{-1} = (L^{-1})^{-1} \). Based on the matrix knowledge, the inverse matrix of the lower triangular matrix is still the lower triangular matrix. Let \( P = L^{-1} \), \( P \) is a lower triangular matrix. \( p_{ij} \) can be obtained by the Eq.(8).

\[
p_{ij} = \begin{cases} 
0 & i < j \\
1/l_{ii} & i = j \\
-p_{ii} \sum_{k=j}^{i-1} l_{ik} p_{kj} & i > j
\end{cases}
\]  

(8)
So, \( A^{-1} = (L^T)^{-1}L^{-1} = P^TP \) \( (9) \)

Formula (9) shows that the output weight of KELM-ML \( \alpha = (I/C + HH^T)^{-1}Y = A^{-1}Y = P^TPY \) can be calculated only by the matrix product operation.

**Large Matrix Cholesky Decomposition Algorithm**

A double type variable takes up 8 bytes on a 64-bit operating system. If size of a matrix is 80000x80000, 80000x80000x8/1024/1024/1024=47.68GB storage space is needed to load whole matrix into memory. Most PCs are unable to load the whole matrix into memory. Therefore, Cholesky matrix decomposition method in Section 4.1 needs to be decomposed and computed in parallel.

The equation \( A = LL^T \) in Section 4.1, and both two sides of matrix of the equation decomposed into 2x2 block matrices, as shown in Eq.(10) and Figure 2 (a). The original matrix \( A \) is a \( m \times m \) square matrix, and \( A_{11} \) is a square matrix of size \( r \times r \). \( r \) is the size of a single PC which can be calculated in the acceptable time (in our experiment set \( r = 200 \) ). In Eq. (10), \( L_{11} \) and \( L_{22} \) are still lower triangular matrices.

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix} = \begin{bmatrix}
L_{11} & \\
L_{21} & L_{22}
\end{bmatrix} \begin{bmatrix}
L_{11}^T & L_{21}^T \\
L_{21} & L_{22}
\end{bmatrix} \tag{10}
\]

According to the basic matrix knowledge, following equations can be deduced, and can be seen in Eq.(11):

\[
\begin{align*}
A_{11} &= L_{11}L_{11}^T \quad (11a) \\
A_{12} &= L_{11}L_{21}^T \quad (11b) \\
A_{21} &= L_{21}L_{11}^T \quad (11c) \\
A_{22} &= L_{21}L_{21}^T + L_{22}L_{22}^T \quad (11d)
\end{align*}
\]

**Solution of \( L_{11} \).** Because \( A_{11} \) is a \( r \times r \) matrix, \( L_{11} \) can be calculated by Cholesky decomposition method directly in single PC.

**Solution of \( L_{21} \).** After calculating the \( L_{11} \), \( L_{11}^{-1} \) can be solved by Eq. (8). Then, based on Eq.(11c), \( L_{21} \) can be obtained by \( L_{21} = A_{21}(L_{11}^{-1})^T = A_{21}L_{11}^T \).
Solution of $L_{22}$. After changing Eq.(11d) into $L_{22}L_{22}^T = A_{22} - L_{21}L_{21}^T$, $L_{22}$ can be calculated by Cholesky decomposition method. If the size of matrix $A_{22} - L_{21}L_{21}^T$ is too large to compute in a single PC, it can be decomposed into small block matrix then recursive solution.

Figure 2 (b) presents the parallel calculation sequence diagram of a 4$r$-rank square matrix Cholesky decomposition. The number represents the calculation sequence of each sub matrix. The first, third, fifth, seventh matrices are decomposed through using Cholesky method. Calculations of these four block matrices are not parallel. The second blocks, the fourth and the sixth blocks are calculated parallel respectively after calculating completion of the previous block.

EXPERIMENTAL ANALYES
Data Sets and Evaluating Indicator

To measure the performance of our approach, three different domain data sets are involved in this paper. The features of data sets are shown in TABLE I. "#Num", "#Dim" and "#Label" are denotes the properties of number of samples, number of features, number of possible class labels respectively. Note that RCV1 (Reuters Corpus Volume 1) uses the "industries:fullsets", which is called "RCV1" for short in this paper. The multi-label learning experimental evaluation indicator adopts the indicators proposed in [12]: "Hamming loss", "One-error", "Coverage", "Ranking loss", these four indicators of lower values, better performance. The experimental results in TABLE III denote with the "↓". The index of "Average precision" the higher the value, the better the performance, the experimental results in TABLE III denote with the "↑".

<table>
<thead>
<tr>
<th>Data set</th>
<th>#Num</th>
<th>#Dim</th>
<th>#Label</th>
<th>Domain</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>2417</td>
<td>103</td>
<td>14</td>
<td>biology</td>
<td>URL1</td>
</tr>
<tr>
<td>Corel16K</td>
<td>13766</td>
<td>500</td>
<td>153</td>
<td>images</td>
<td>URL2</td>
</tr>
<tr>
<td>RCV1</td>
<td>804414</td>
<td>47236</td>
<td>313</td>
<td>text</td>
<td>URL1</td>
</tr>
</tbody>
</table>

URL1: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
URL2: http://mulan.sourceforge.net/datasets.html

Experimental Results

The experiment compares the performance of KELM-ML algorithm with the existing multi-label classification learning algorithm: Rank-SVM [9], ML-KNN [8], ECC[6]. KELM-ML is deployed on a distributed cluster built on Hadoop-2.7.3 platform. The cluster possesses 4 PC (2.5GHz CPU and 4GB RAM) with Ubuntu 14.10 operating system. Eclipse is adopted as the development tool and Java as the programming language. The initial large matrices are saved as HDFS files. After carrying out the test, for KELM-ML, the optimal kernel parameter $\sigma = 2^{-2}$, cost parameter $C = 2^9$ in the Yeast dataset, and the optimal kernel parameter $\sigma = 2^{-2}$, cost
parameter \( C = 2^3 \) in Corel16K dataset and RCV1 dataset. Certainly, the parameter selection of KELM-ML algorithm is time-consuming. In terms of ECC, ensemble size is set to be 10, and sample ratio is set to be 67\% [6]. The "cross - validation" method is used in the training process. The algorithm average training time are shown in TABLE II, and the time unit is second. Obviously, the training time of KELM-ML algorithm is less than other algorithms. Test set results are presented in TABLE III. According to TABLE III, KELM-ML algorithm in "Hamming loss" and "Ranking loss" indicators gain outstanding performance, and "One-error", "Coverage", "Average Precision" indicators are slightly inferior to the ECC algorithm. In general, KELM-ML algorithm greatly reduces the training time and achieves better overall performance with the supports of parallel computing.

### TABLE II. TRAINING TIME OF EACH ALGORITHM (MEAN VALUE).

<table>
<thead>
<tr>
<th>Data set</th>
<th>KELM-ML</th>
<th>Rank-SVM</th>
<th>ML-KNN</th>
<th>ECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>0.49</td>
<td>3058</td>
<td>0.32</td>
<td>473</td>
</tr>
<tr>
<td>Corel16K</td>
<td>38.16</td>
<td>5974</td>
<td>37.38</td>
<td>947.34</td>
</tr>
<tr>
<td>RCV1</td>
<td>126.35</td>
<td>33215</td>
<td>829.20</td>
<td>327.45</td>
</tr>
</tbody>
</table>

### TABLE III. RESULTS OF COMPARING ALGORITHM (MEANS±STD).

<table>
<thead>
<tr>
<th>Data set</th>
<th>criterion</th>
<th>KELM-ML</th>
<th>Rank-SVM</th>
<th>ML-KNN</th>
<th>ECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>Hamming loss↓</td>
<td>0.194±0.013</td>
<td>0.207±0.006</td>
<td>0.194±0.011</td>
<td>0.208±0.002</td>
</tr>
<tr>
<td></td>
<td>One-error↓</td>
<td>0.193±0.017</td>
<td>0.243±0.017</td>
<td>0.228±0.029</td>
<td>0.176±0.022</td>
</tr>
<tr>
<td></td>
<td>Coverage↓</td>
<td>0.459±0.013</td>
<td>0.514±0.038</td>
<td>0.463±0.012</td>
<td>0.516±0.015</td>
</tr>
<tr>
<td></td>
<td>Ranking loss↓</td>
<td>0.167±0.023</td>
<td>0.195±0.011</td>
<td>0.167±0.004</td>
<td>0.171±0.006</td>
</tr>
<tr>
<td></td>
<td>Average Precision↑</td>
<td>0.728±0.019</td>
<td>0.749±0.009</td>
<td>0.765±0.001</td>
<td>0.767±0.019</td>
</tr>
<tr>
<td>Corel16K</td>
<td>Hamming loss↓</td>
<td>0.018±0.006</td>
<td>0.019±0.003</td>
<td>0.018±0.001</td>
<td>0.030±0.001</td>
</tr>
<tr>
<td></td>
<td>One-error↓</td>
<td>0.697±0.004</td>
<td>0.846±0.013</td>
<td>0.740±0.006</td>
<td>0.657±0.020</td>
</tr>
<tr>
<td></td>
<td>Coverage↓</td>
<td>0.315±0.006</td>
<td>0.321±0.012</td>
<td>0.332±0.002</td>
<td>0.365±0.009</td>
</tr>
<tr>
<td></td>
<td>Ranking loss↓</td>
<td>0.155±0.007</td>
<td>0.176±0.004</td>
<td>0.168±0.002</td>
<td>0.168±0.012</td>
</tr>
<tr>
<td></td>
<td>Average Precision↑</td>
<td>0.273±0.012</td>
<td>0.194±0.006</td>
<td>0.271±0.001</td>
<td>0.199±0.008</td>
</tr>
<tr>
<td>RCV1</td>
<td>Hamming loss↓</td>
<td>0.081±0.002</td>
<td>0.102±0.007</td>
<td>0.089±0.002</td>
<td>0.092±0.006</td>
</tr>
<tr>
<td></td>
<td>One-error↓</td>
<td>0.384±0.010</td>
<td>0.646±0.009</td>
<td>0.386±0.006</td>
<td>0.392±0.011</td>
</tr>
<tr>
<td></td>
<td>Coverage↓</td>
<td>0.301±0.008</td>
<td>0.322±0.009</td>
<td>0.320±0.006</td>
<td>0.286±0.005</td>
</tr>
<tr>
<td></td>
<td>Ranking loss↓</td>
<td>0.123±0.005</td>
<td>0.147±0.008</td>
<td>0.126±0.004</td>
<td>0.129±0.006</td>
</tr>
<tr>
<td></td>
<td>Average Precision↑</td>
<td>0.614±0.013</td>
<td>0.600±0.047</td>
<td>0.530±0.019</td>
<td>0.626±0.018</td>
</tr>
</tbody>
</table>
CONCLUSIONS

In this paper, a multi-label learning algorithm KELM-ML is designed based on the kernel extreme learning machine, and an adaptive threshold function is set up. Specially, the KELM-ML output weight is obtained with the application of the Cholesky Matrix Decomposition method. The inverse problem of large matrix is solved by matrix block method. Finally, the comparison experiment shows that KELM-ML has better effect and fast training time.

ACKNOWLEDGEMENT

This work was supported by the National Science Foundation of China under Grant No.61672159 and 61571129, the Fujian Collaborative Innovation Center for BigData Application in Governments, the Technology Innovation Platform Project of Fujian Province(Grant No.2009J1007 and 2014H2005).

REFERENCES
