Neurocomputing 260 (2017) 313–320

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A multi-label classification algorithm based on kernel extreme learning machine☆

Fangfang Luo[a,d], Wenzhong Guo[a,b,c,*], Yuanlong Yu[a,b], Guolong Chen[a,b,c]

[a] College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China
[b] Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou 350116, China
[c] Key Laboratory of Spatial Data Mining and Information Sharing, Ministry Of Education, Fuzhou 350003, China
[d] College of Computer Engineering, Jimei University, Xiamen 361021, China

ARTICLE INFO

Article history:
Received 25 September 2016
Revised 14 April 2017
Accepted 17 April 2017
Available online 11 May 2017

Communicated by Dr. G.-B. Huang

Keywords:
Multi-label learning
Extreme learning machine
Kernel extreme learning machine
Threshold selection

ABSTRACT

Multi-label classification learning provides a multi-dimensional perspective for polysemic object, and becomes a new research hotspot in machine learning in recent years. In the big data environment, it is urgent to obtain a fast and efficient multi-label classification algorithm. Kernel extreme learning machine was applied to multi-label classification problem (ML-KELM) in this paper, so the iterative learning operations can be avoided. Meanwhile, a dynamic, self-adaptive threshold function was designed to solve the transformation from ML-KELM network's real-value outputs to binary multi-label vector. ML-KELM has the least square optimal solution of ELM, and less parameters that needs adjustment, stable running, faster convergence speed and better generalization performance. Extensive multi-label classification experiments were conducted on data sets of different scale. Comparison results show that ML-KELM outperformance in large scale dataset with high dimension instance feature.

1. Introduction

Classification is an important data analysis technology, which is widely used in data mining, machine learning, pattern recognition and other fields. In traditional classification learning, each sample is only associated with a single category label which signifies a sema ntic meaning. This model is called "single label classification". At present, many accurate and efficient algorithms have been proposed for single label classification problem such as SVM, decision tree, naive Bayes method and artificial neural network. Although great progress has been made in research on single label classification, and they are widely used in the real life, but with the advent of the era of big data, the hypothesis of "one sample with single category label" cannot accurately describe the real object since the form of data have various semantic information. Because of the complexity and ambiguity of the object itself, most objects may be associated with multiple category labels simultaneously in reality. For instance, a natural scene can hold three labels simultaneously, "mountain", "sunset", "tree"; a melody can be labeled with "Orchestra", "dance","Strauss John". Multi-label classification has been widely used in many practical domains, such as media annotation[1], semantic annotation of images and video [2,3] and biological proteins function classification [4].

Multi-label classification is a kind of supervised learning. Its formal definition can be described as:

- Let $\mathcal{X} = \mathbb{R}^d$ denotes a d-dimensional instances space of numerical or categorical features.
- $\mathcal{Y} = \{y_1, y_2, \ldots, y_q\}$ be a finite set of labels, the total number of labels space is q, $q > 1$.
- $\mathcal{D} = \{(x_i, Y_i) | 1 \le i \le m\}$ denotes a training set with m instances, $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T$ is the ith training instance which has label set Y_i associated, $Y_i \subseteq \mathcal{Y}$. $Y_i = (y_1, y_2, \ldots, y_q) = \{-1, 1\}^q$ 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 $h : \mathcal{X} \to 2^{\mathcal{Y}}$ by training sample instances, which can map an instance to a label set. After that, input an unseen instance $x \in \mathcal{X}$ into multi-label classifier, which can output an associate label set $h(x)$, $h(x) \subseteq \mathcal{Y}$.

☆ This work was supported in part by the National Natural 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), and the Fujian Natural Science Funds for Distinguished Young Scholars under Grant No. 2014J06017.

* Corresponding author at: College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350116, China.
E-mail address: fzugwz@163.com (W. Guo).

http://dx.doi.org/10.1016/j.neucom.2017.04.052
0925-2312/

Compared with single label classification, multi-label classification problem coincides more properly with the characteristics and rules of objective world. Moreover, the single label classification problem is essentially a special case of multi-label classification, and the multi-label classification is a more universal and generalized version of the single label classification. However, this extension inevitably brings about more challenges and difficulties to the problem, the main difficulty lies in the over huge output label subsets space. For example, given a data set that contains 20 different category labels, the number of candidate label subsets corresponding to an unseen sample reaches a million levels (2^{20}). Choosing the correct label subset is a huge challenge from the candidate subset spaces of exponential scale. Therefore, it is of great theoretical significance and broad application prospect to design an accurate and efficient multi-label classification algorithm in machine learning field.

Kernel extreme learning machine was used to solve multi-label classification problem in this paper, and a threshold function was designed to realize the conversion from network real valued output to binary vector. Parameters of the threshold function are obtained from the training set, which is self-adaptive. Compared with existing ELM algorithms for multi-label classification problem, ML-KELM runs more stable. Experiments were conducted in six different kind datasets, and results show that, compared with other existing algorithms, the training time of ML-KELM algorithm is short in large scale data set obviously. And ML-KELM algorithm is applicable to multi-label classification problems especially for large-scale dataset.

The rest of this article is organized as follows. In Section 2, related work is provided. The framework of ML-KELM algorithm and the details are illustrated in Section 3. Section 4 shows experimental results. Finally, conclusions and discussion of the future work are made in Section 5.

2. Related work

2.1. Three ways to solve multi-label learning problem

There are mainly three methods to solve multi-label classification [5,6]: “problem transformation” method , “algorithm adaptation” method and “ensemble” method. “Problem transformation” method transforms the learning task into one or several traditional single-label classification tasks. Such as BR(Binary Relevance) Algorithm [2], it converted multi-label classification problem into a series of independent binary classification problems, where each label corresponded to a binary classifier, then combined all binary classifiers to obtain the predicted label subsets for unseen sample. Each classifier can be trained independently in parallel; the algorithm seems intuitive and efficient, but it failed to consider the correlation between labels, so the algorithm has poor generalization performance. Pairwise Multi-label Algorithm [7] established a series of multi-class classifiers through the integrated framework. Each classifier chose k subsets from all candidate subsets in label space, then constructed by LP method, and finally predicted the label set of unseen sample through the voting method finally. Generally speaking, the ability of “problem transformation” method cannot process the relationship between labels very well.

The second method is to extend special learning algorithms to handle multi-label data directly. Such as ML-KNN Algorithm [8], it obtained the prediction label subset for an unseen sample by using the maximum posteriori probability of the k-nearest neighbor samples. Rank-SVM Algorithm [9], it solved the multi-label classification problem by constructing SVM classifier for each label according to the ranking loss function. So the correlation between samples and labels could be learned more accurately. According to different objective function, BoostTexter [10,11] extended the “Adaboost integration algorithm” into two parts: “Adaboost.MH” and “Adaboost.MR”. Adaboost.MH minimized the Hamming loss function; Adaboost.MR minimized the Ranking loss function, and the associated labels would be selected and ranked in the front of the ordered list by these two parts. Although these algorithms considered the correlations between labels, and had good effect on small scale data sets (instance number less than 6000), however, it consumed long time in the training process in large scale data sets, and the efficiency was not high.

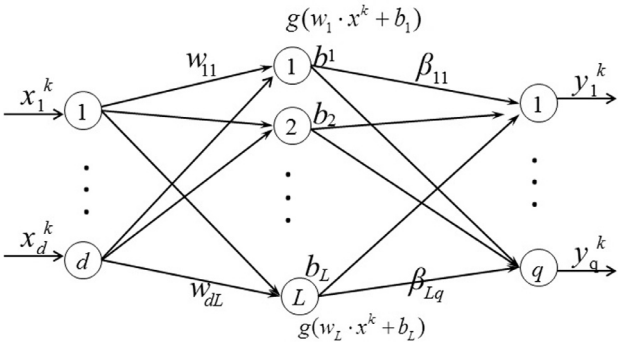

Fig. 1. ELM network structure diagram.

The third method based on multi-label classifiers used an ensemble of ‘algorithm adaptation” or “problem transfo rm” classifiers to address the multi-label classification problem. For instance, CC(classifier chains) Algorithm [12] transformed multi-label classification problem to construct a classifier chain, each classifier chain corresponded to a label, and the process of constructing the back classifier in the chain relied on the front of classifier. And then the correlation between the labels was considered by the sequence of classifier chain. However, the relationship between labels is computed randomly because the classifier in chain is randomly arranged. Moreover, if the performance of the previous classifier is not good, the classification error will spread along with the chain continually. This type of algorithm are also Random k labelsets, Random forest based on predictive clustering trees, Random Decision Tree.

This paper proposes an “algorithm adaptation” multi-label classifier based on kernel extreme learning machine.

2.2. Multi-label classification related work based on ELM

Extreme Learning Machine(ELM) was proposed by Huang et al. [13], it has the characteristics of high speed and high efficiency, since it can avoid the tedious iterative learning process. ELM is a single hidden layer feed-forward neural network. ELM can guarantee the network has good generalization performance; meanwhile, it avoids problems caused by the iterative learning of traditional feed-forward neural network, such as the learning parameter setting randomly, falling into local minimum easily and so on.

Sun et al. [14], Meng et al. [15,16] apply ELM to solve multi-label classification problem.There are several key points in applying ELM to multi-label problems. The first is the preprocessing of the input, which is converted form unipolar to bipolar representation. The multi-label output corresponding to the input sequence is in general provided as “-1” and “1” for each label, with “1” represents the labels relevant with the input sample, and “-1” is irrelevant. The second step is to obtain the ELM network structure through the training set. As show in Fig. 1, Extreme Learning Machine (ELM) is a three layers neural network with only one hidden layer. Let the neuron number of ELM’s hidden layer is L, and activation function is $g(x)$. Suppose the training set is composed of

m multi-label instances, i.e. $\{(x_1, Y_1), (x_2, Y_2), \ldots, (x_m, Y_m)\}$, where each instance $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})^T$ is a d-dimensional vector and $Y_i \subseteq \mathcal{Y}$ is the set of labels associated with this instance. So the input layer has d neurons, and output layer has q neurons. Randomly assign input weight w (the weight between input layer and hidden layer) and hidden layer bias b. Then calculate the output of the lth hidden neuron according $g(w_i^{(l)} \cdot x^{(l)} + b_i^{(l)})$, and then the hidden layer output matrix $H = [h^T(x_1), \ldots, h^T(x_m)]^T$ can be obtained, where H is an $m \times L$ matrix. The centered inner dot in $g(w_i^{(l)} \cdot x^{(l)} + b_i^{(l)})$ represents inner product operation. The hidden layer output weight $\beta = H^{\dagger}Y$ can be calculated using one step of inverse matrix operation, according to Moore–Penrose generalized inverse matrix theory. β is $L \times q$ matrix.The ELM's output $f(x)$ is the bipolar $m \times q$ matrix. This method could improve the generalization ability and learning speed of neural network greatly; this method avoids the complex iterative operation too.

The mathematical model of the ELM network for multi-label classification learning can be expressed by Eqs. (1) and (2):

$$\min L_{ELM} = \| \beta \|^2 + C \sum_{i=1}^{m} \| \xi_i \|^2 \tag{1}$$

$$s.t. \quad f(x_i) = h(x_i)\beta = Y_i - \xi_i \qquad 1 \le i \le m \tag{2}$$

Where, $\beta = [\beta_1, \ldots, \beta_L]^T$ is output weight of hidden layer, and C is cost parameter (also called ridge regression parameter). ξ_i is the error between theoretical output Y_i and the actual output $f(x_i)$. $h(x_i)$ is the hidden layer output vector of instance x_i. According to Karush–Kuhn–Tucker (KKT) optimal conditions, the hidden output weight β can be obtained by Eq. (3):

$$\beta = H^T \left(\frac{I}{C} + HH^T \right)^{-1} Y \tag{3}$$

Where, $H = [h^T(x_1), \ldots, h^T(x_m)]^T$ is the hidden layer output matrix of training set, $Y = [Y_1, \ldots, Y_m]^T$ is the target matrix of training set. I is a m-order unit matrix.

The ELM's output $f(x)$ is the multiplication of hidden layer output matrix H and hidden layer output weight β, which can be presented by Eq. (4):

$$f(x) = H\beta = HH^T \left(\frac{I}{C} + HH^T \right)^{-1} Y \tag{4}$$

In the testing phase, the test output $f(x)$ is evaluated by Eq. (4). For multi-label classification, the $m \times q$ values of output $f(x)$ are passed as arguments to the bipolar step function. A threshold of '0' is applied to resultant values [16]. Threshold processing is described by formula (5). The set of columns of the resulting matrix with values 1 gives the multi-label belongingness of the corresponding input.

$$finaloutputs(k, i) = \begin{cases} 1 & f_k(x_i) \ge 0 \\ -1 & f_k(x_i) < 0 \end{cases} i = 1, \ldots, m', \quad k = 1, \ldots, q \tag{5}$$

Detailed processes for ELM algorithm applied to multi-label problem are presented as Algorithm 1.

In practical training set, there are a lot of incomplete object labels, which will significantly increase the difficulty of multi-label classification learning. Zhang and Zhang proposes a Domain Adaptation Extreme Learning Machine (DAELM) [17,18], which learns a robust classifier by leveraging a limited number of labeled data from target domain for drift compensation. This method is of great significance and can be extended to incomplete label learning.

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 β 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** $finaloutputs$

3. Multi-label classification algorithm based on kernel extreme learning machine (ML-KELM)

The randomness of ELM's hidden layer nodes setting would give rise to the oscillation of hidden layer output matrix, which brings down stability of the network structure, and it also lacks an appropriate threshold learning mechanism. The essence of ELM's hidden layer is a nonlinear explicit mapping, which maps the input samples to another space in which they could be separated linearly. This mapping method is consistent with the principle of kernel function. The kernel function can solve the "dimension disaster" problem by mapping the high-dimensional feature vectors into a low-dimensional space through the inner product operation. Therefore, this paper will combine this two theory, using kernel extreme learning machine to solve multi-label classification problem, called ML-KELM. Compared with ELM, using ML-KELM, as long as the kernel function and related parameters are determined, the results are stable.

3.1. ML-KELM algorithm

Based on inner product of kernel function theory, Huang et al. adopt kernel function $K(u, v)$ to replace ELM's hidden layer mapping in order to make network stable [19]. This method is applied in multi-label classification problem.

For ML-KELM network, it's not necessary to calculate hidden layer output, neither set neuron number of hidden layer. ML-KELM just needs to choose a kernel function $K(u, v)$, then it can calculate network output $f(x)$ by Eq. (4), where HH^T could be replaced by kernel function $HH^T(i, j) = K(x_i, x_j)$. Definition of Kernel matrix $HH^T = \Omega_{ELM}$ can be presented by Eq. (6):

$$\Omega_{ELMi,j} = h(x_i) \cdot h(x_j) = K(x_i, x_j) \tag{6}$$

So, the output of ML-KELM network $f(x)$ can be presented by Eq. (7):

$$f(x) = HH^T \left(\frac{I}{C} + HH^T \right)^{-1} Y = \begin{bmatrix} K(x, x_1) \\ \vdots \\ K(x, x_m) \end{bmatrix}^T \left(\frac{I}{C} + \Omega_{ELM} \right)^{-1} Y \tag{7}$$

In this paper, radial basis kernel function is adopted, $K(x, x_i) = exp(-\frac{\|x - x_i\|^2}{2\sigma^2})$.

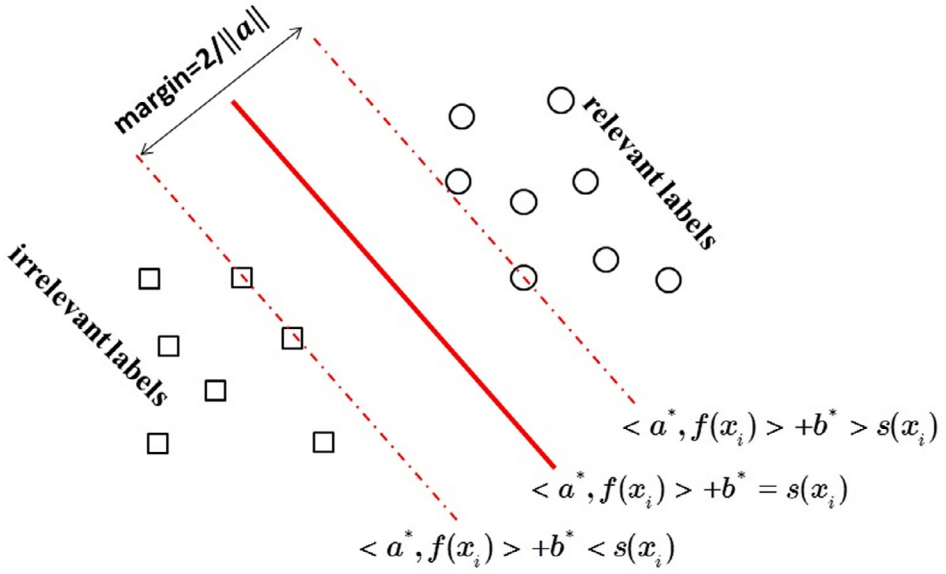

Fig. 2. Schematic diagram of threshold function.

Compared to ELM, ML-KELM has a stronger classify ability based on the powerful mapping capabilities of kernel function. If appropriate kernel parameter is adopted, ML-KELM network will work more stably.

3.2. Threshold function setting

As shown in the Eq. (7), ML-KELM network output $f(x) = (f_1(x), \ldots, f_q(x))^T \in \mathbb{R}^q$ is a q-dimensional real valued vector, whose components correspond to categories represented by neurons in output layer, and the total number of categories is q.

According to the definition of multi-label classifier in Section 1, for an input instance x , the classifier output Y is a q-dimensional vector, where its elements are -1 or 1. A threshold function is needed to transfer the output of the ML-KELM to the desired binary vector. The role of threshold function is dividing label space into relevant labels or irrelevant labels by the training set learning. In other words, threshold function is used to find a boundary between relevant labels and irrelevant labels. This boundary cannot be defined simply by a constant value, which should be related to the whole training set, dynamic and self-adaptive. In this paper, the threshold function is enlightened by the threshold setting method in Rank-SVM algorithm [9,20]. The linear function was chosen as threshold, because of its few parameters and simple implementation.

As shown in Fig. 2, threshold is set as a linear function $t(x) = \langle a^*, f(x)\rangle + b^*$, where a^* is a q-dimensional vector, b^* is the offset. Then the problem is converted to finding optimal parameter a^* and b^*, which makes the "margin" between relevant labels and irrelevant labels reach the maximum. For the instance x_i in the training set $\mathcal{D}$, the boundary value of the relevant labels and irrelevant labels can be found by comparing the ML-KELM network output $f(x_i)$ with the target value Yi, which is denoted as $s(x_i)$. The definition of $s(x_i)$ could be seen in formula (8). For the solution of formula (8), the optimal approximation of $s(x_i)$ can be found in the candidate discrete set.

$$s(x_i) = \arg\min_{r\in R}(|\{y_j | y_j \in Y_i, f_j(x_i) \le r\}| + |\{y_k | y_k \notin Y_i, f_k(x_i) \ge r\}|) \tag{8}$$

For a given training set $\mathcal{D} = \{(x_i, Y_i) | 1 \le i \le m\}$, the corresponding s is m-dimensional vector. On the basis of solving s, the least squares method is used to obtain the optimal parameters a^* and b^* in Eq. (9).

$$\min_{\{a^*,b^*\}} \sum_{i=1}^{m} (\langle a^*, f(x_i)\rangle + b^* - s(x_i))^2 \tag{9}$$

The objective function represented by formula(9) minimizes the sum of the distances between $\langle a^*, f(x_i)\rangle + b^*$ and $s(x_i)$ in the training set $\mathcal{D}$.

For a testing set $\mathcal{D}' = \{(x_i, Y_i) | 1 \le i \le m'\}$ with m' instances, and its ML-KELM network output is $f(x)$. Dimension of $f(x)$ is $m' \times q$. Threshold can be obtained by Eq. (10):

$$t = [[f(x), E_{m'\times 1}][a^*, b^*]^T]^T \tag{10}$$

As seen in Eq. (10), threshold t is a m'-dimensional vector; a^* and b^* are only related with training set $\mathcal{D}$, but threshold t is simply related with the testing set number m'. E is a $m' \times 1$ dimensional matrix with all element values of 1. Then we pass $f(x)$ through threshold t to obtain multi-label classification final results *finaloutputs*, threshold process method could be seen in Eq. (11):

$$finaloutputs(k, i) = \begin{cases} 1 & f_k(x_i) \ge t(k) \\ -1 & f_k(x_i) < t(k) \end{cases} i = 1, \ldots, m', \quad k = 1, \ldots, q \tag{11}$$

Detailed processes of the ML-KELM algorithm are shown in Algorithm 2.

4. Experimental analysis

4.1. Algorithm performance evaluation indicators

Experiments in this paper used the evaluation indicators based on samples [5,21], mainly included "Hamming loss", "One error", "Coverage", "Ranking loss"," Average precision". For a testing set $\mathcal{D}' = \{(x_i, Y_i) | 1 \le i \le m'\}$, the calculation methods of the specific indicators are shown as follows (by Eqs. (12)–(16)):

- Hamming loss: evaluating the inconsistency between the predictive label sets and the actual label sets of samples, the calculating method can be seen in Eq. (12):

$$hloss_s(h) = \frac{1}{m'} \sum_{i=1}^{m'} \frac{1}{q} | h(x_i) \Delta Y_i | \tag{12}$$

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 σ , 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 Ω_{ELM};
5: Compute ML-KELM's network structure $(\frac{I}{C}+\Omega_{ELM})^{-1}Y$;
6: Compute threshold parameter a^*, b^* by training set $\mathcal{D}$ according to Eq. (9);
7: **end for**
8: **for** testing set $\mathcal{D}'$ **do**
9: Calculate outputs $f(x)$ according to Eq. (7);
10: Construct threshold function t based on parameter a^*, b^* according to Eq. (10);
11: Post-process: compare $f(x)$ with threshold t to obtain $finaloutputs$ according to Eq. (11).
12: **end for**
13: **return** $finaloutputs$

Where, Δ represents the symmetric difference between two labels sets.

- One error: evaluating the proportion of samples highest score label is not the actual label of the sample, and the calculating method can be seen in Eq. (13):

$$one-error_s(h)=\frac{1}{m'}\sum_{i=1}^{m'}\frac{1}{q}\left[\left[\left[\arg\max_{y\in\mathcal{Y}} f(x_i,y)\right]\notin Y_i\right]\right] \tag{13}$$

For any predicate π, if satisfied the value of $[[\pi]]$ is 1, otherwise the value of $[[\pi]]$ is 0.

- Coverage: ranging the outputs $f(x)$ from high to low, then evaluates the numbers for the average to go down the lists of labels in order to cover all the proper labels of the instance. The calculating method can be seen in Eq. (14):

$$coverage_s(h)=\frac{1}{m'}\sum_{i=1}^{m'}\max_{y\in Y_i} rank_f(x_i,y)-1 \tag{14}$$

$rank_f(\cdot,\cdot)$ is the ranking function that corresponding to the $f(\cdot,\cdot)$ of real valued function.

- Ranking loss: describing the average fraction of label pairs that are reversely ordered for the instance. The calculating method can be seen in Eq. (15):

$$rloss_s(h)=\frac{1}{m'}\sum_{i=1}^{m'}\frac{1}{|Y_i||\overline{Y}_i|}\left|\{(y',y'')|f(x_i,y')\le f(x_i,y''),(y',y'')\in Y_i\times\overline{Y}_i\}\right| \tag{15}$$

Where $\overline{Y}_i$ denotes the complementary set of Y_i in label space $\mathcal{Y}$.

- Average precision: describing the average precision of the predictive multi-label. The calculating method can be seen in Eq. (16):

$$avgprec_s(h)=\frac{1}{m'}\sum_{i=1}^{m'}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{\left|\{y'|rank_f(x,y')\le rank_f(x,y),y'\in Y_i\}\right|}{rank_f(x_i,y)} \tag{16}$$

For evaluation indicators such as "Hamming loss", "One error", "Coverage" and "Ranking loss" the smaller the value, the better performance of multi-label classification system; on the contrary, for "Average precision", the bigger the better performance of the multi-label classification system.

4.2. Experimental data sets

Experiments use six kinds of data sets, which are commonly used in multi-label classification: "Emotions dataset","Yeast gene function (Yeast) dataset", "natural scene (Scene) dataset", "Corel16K dataset" and "rcv1v2 (Routers Corpus Volume1-topics) dataset" [22]. The features of the data set are shown in Table 1. "#Train" indicates the number of training instances in dataset; "#Test" indicates the number of testing instances in dataset; "#Dim" indicates the dimension of sample instance; "#Label" indicates the number of labels in dataset; "#LC" indicates label cardinality of dataset; "#Field" indicates the domain fields of dataset; "#URL" indicates the source URL of dataset.

4.3. Algorithm comparison

In order to test the effectiveness of ML-KELM, four kinds state-of-the-art multi-label learning algorithms are selected to comparison. There are Rank-SVM [9] that also based on kernel function, ML-KNN algorithm [8] based on k-nearest neighbor theory, BoosTexter algorithm [10] based on ensemble learning mechanism and classical Extreme Learning Machine (ELM) algorithm [15,16]. All experiments are conducted in MATLAB2012. Five indicators are comparatively analyzed respectively: "Hamming loss", "One error", "Coverage", "Ranking loss", and "Average precision". After determining the scale of training set and testing set, the "cross validation" method is used for the experiment. The comparison results are shown in Tables 2–7. For the first 4 indicators value, the smaller, the better performance, "↓" is used to indicate the trend. For the indicator "Average precision" , the bigger the better, indicated with "↑". The best performance indicators are showed in bold.

Table 8 reports the computation time consumed by each multi-label learning algorithm, where all experiments are conducted on PC with 2.5 GHz CPU and 4 GB memory.

4.4. Experiment result analysis

During the experiment, the ELM algorithm needs to test different hidden layer nodes number, and then choose the best value. After testing, the optimal hidden layer nodes number is 60 in Emotions dataset; the optimal hidden layer nodes number is 100 in the Yeast dataset; and the optimal hidden layer nodes number is 85 in the Scene dataset; and the optimal hidden layer nodes number is 250 in Corel16K dataset; and the optimal hidden layer nodes number is 120 in the fifth group rcvlv2 (subset1) dataset; the sixth groups rcvlv2 (fullset) dataset's optimal hidden layer nodes number is 600. Selection processes of the optimal hidden layer nodes number are time-consuming. And even the best hidden layer nodes number is already chose, results of each test are still not stable, because of the randomness setting of input weight.

For ML-KELM algorithm, the comparison tests are also needed to be carried out with kernel parameter σ and cost parameter C. After testing, the optimal kernel parameter $\sigma=2^{-2}$, cost parameter $C=2^0$ in the Emotions dataset and Yeast dataset; and the optimal kernel parameter $\sigma=2^{-2}$, cost parameter $C=2^0$ in the Scene dataset; and the optimal kernel parameter $\sigma=2^{-2}$, cost parameter $C=2^1$ in the Corel16K dataset and rcvlv2 (subset1) dataset; the sixth groups rcvlv2 (fullset) dataset's optimal kernel parameter $\sigma=2^{-2}$, cost parameter $C=2^3$. The processes of testing and optimal parameter choosing are also time-consuming, while the results of ML-KELM are stable if the parameters are determined.

As can be seen from Table 2, ML-KELM in indicators of "Hamming loss", "One error", "Average precision" perform well. In Table 3 the performance of ML-KNN algorithm is optimal in the Yeast dataset, ML-KELM is consistent with ML-KNN algorithm on

Table 1
Feature of six data sets.

Data set	#Train	#Test	#Dim	#Label	#LC	#Field	#URL
Emotions	400	193	72	6	1.869	Music	URL2
Yeast	1500	917	103	14	4.237	Biology	URL1
Scene	2000	407	294	6	1.074	Image	URL2
Corel16K	8000	5760	500	154	3.522	Image	URL2
rcv1v2(topics;subset1)	3000	3000	47,236	101	2.880	Text	URL3
rcv1v2(topics;fullset)	23,149	781,265	47,236	101	2.642	Text	URL3

URL1:http://cse.seu.edu.cn/PersonalPage/zhangml/index.htm
URL2:http://mulan.sourceforge.net/datasets.html
URL3:https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

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

"One error" and "Ranking loss". In Table 4 of the Scene dataset, ML-KELM algorithm gains the best performance in indicators of "One error", "Coverage", "Ranking loss" and "Average precision". In Table 5 Corel16K dataset, "Hamming loss", "One error", "Coverage", "Ranking loss" these four indicators ML-KELM perform best. On the rcvlv2 (subset1) dataset of Table 6, the indicators of ML-KELM are optimal in indicators of "Hamming loss", "One error", "Coverage" and "Average Precision". In Table 7 rcvlv2 (fullset) dataset, "Hamming loss", "Coverage", "Ranking loss" these three indicators ML-KELM perform best, other two indicators "One error", "Average precision" are also superior.

As shown in Table 8, ML-KELM consumes a little bit more time than ELM algorithm, because it spends more time to calculate kernel matrix. But ML-KELM runs much faster than the other three algorithms obviously. Samples that less than two thousand, the training time is usually less than one second; and the training time of thirty thousand samples is only about two minutes. In addition to the ELM algorithm, 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 test phase, ML-KELM algorithm spends little time, because it only needs inner product and matrix multiplication operation.

Thus it can be seen that ML-KELM perform better than other algorithm in a large amount of data (the instance number of rcvlv2 subset1 is 6000, and the instance number of Corel16K is $13,760$, and the instance number of rcvlv2 fullset is $804,414$), high feature dimension (the feature dimension of rcvlv2 dataset is $47,236$), multi-label categories various situation. After suitable parameter setting, ML-KELM algorithm is adapted for multi-label classification learning in big data environment.

Zhang and Zhang analyzed the sensitivity of parameters in the reference [23]. We can use this method to analyze the kernel parameter sensitivity of ML-KELM. Zhang et al. applied sparse coding and latent subspace to domain adaptation learning in reference [24]. The sparse coding can be considered to improve the kernel mapping ability of ML-KELM algorithm.

5. Conclusions

The goal of multi-label classification learning is to designate a set of appropriate labels to an unseen object. It is of great significance for the study of polysemy object modeling. In this paper, multi-label classification algorithm ML-KELM was designed based on the principle of kernel extreme learning machine, and set an adaptive threshold function. The ML-KELM algorithm is tested on the Emotions dataset, the Yeast dataset, the Scene dataset and the large scale text data sets. Comparison experiments show that ML-KELM has less parameter needing adjustment, faster convergence speed, and better generalization performance. It performs better and stably in large scale data environment, and has practical significance in the big data research. The next step of the research work is to design a ML-KELM algorithm with optimal kernel parameters by sensitivity analysis of parameter settings. Furthermore, sparse coding will be used to improve the performance of ML-KELM.

References

[1] E. Cakir, T. Heittola, H. Huttunen, et al., Polyphonic sound event detection using multi label deep neural networks[C], in: International Joint Conference on Neural Networks, IEEE, 2015, pp. 1–7.
[2] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (9) (2004) 1757–1771.
[3] Z. Chen, Z. Chi, H. Fu, D. Feng, Multi-instance multi-label image classification: a neural approach, Neurocomputing 99 (1) (2013) 298–306.
[4] S. Wan, M.W. Mak, S.Y. Kung, Mem-ADSVM: a two-layer multi-label predictor for identifying multi-functional types of membrane proteins, J. Theor. Biol. 398 (2016) 32–42.
[5] M.L. Zhang, Z.H. Zhou, A review on multi-label learning algorithms, IEEE Trans. Knowl.-Data Eng. 26 (8) (2014) 1819–1837.
[6] G. Tsoumakas, I. Katakis, Multi-label classification: an overview, Int. J. Data Warehous.-Min. 3 (3) (2010) 1–13.
[7] E.L. Mencía, S.H. Park, J. Fürnkranz, Efficient voting prediction for pairwise multilabel classification, Neurocomputing 73 (7–9) (2010) 1164–1176.
[8] M.L. Zhang, Z.H. Zhou, ML-KNN: a lazy learning approach to multi-label learning, Pattern Recognit. 40 (7) (2007) 2038–2048.
[9] A.E. Elisseeff, J. Weston, A kernel method for multi-labelled classification, Adv. Neural Inf. Process. Syst. 14 (2002) 681–687.
[10] R.E. Schapire, Y. Singer, Boostexter: a boosting-based system for text categorization, Mach. Learn. 39 (2–3) (2000) 135–168.
[11] C. Rudin, R.E. Schapire, Margin-based ranking and an equivalence between AdaBoost and RankBoost, J. Mach. Learn. Res. 10 (3) (2009) 2193–2232.
[12] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, Mach. Learn. 85 (3) (2009) 254–269.
[13] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[14] X. Sun, J. Wang, C. Jiang, J. Xu, J. Feng, S.S. Chen, F. He, ELM-ML: Study on Multi-label Classification Using Extreme Learning Machine, Springer International Publishing, 2016.
[15] J.E. Meng, R. Venkatesan, W. Ning, A High Speed Multi-label Classifier Based on Extreme Learning Machines[M], in: Proceedings of ELM-2015 Volume 2, Springer International Publishing, 2016.
[16] R. Venkatesan, J.E. Meng, Multi-label classification method based on extreme learning machines, in: Proceedings of the International Conference on Control Automation Robotics and Vision, 2015, pp. 619–624.
[17] L. Zhang, D. Zhang, Robust visual knowledge transfer via extreme learning machine based domain adaptation 25 (10) (2016) 1.
[18] L. Zhang, D. Zhang, Domain adaptation extreme learning machines for drift compensation in e-nose systems, IEEE Trans. Instrum. Measur. 64 (7) (2015) 1790–1801.
[19] G.B. Huang, H. Zhou, X. Ding, et al., Extreme learning machine for regression and multiclass classification[J], IEEE Transactions on Systems Man & Cybernetics Part B 42 (2) (2012) 513.
[20] L. Skorkovská, Dynamic Threshold Selection Method for Multi-label Newspaper Topic Identification, Springer, Berlin Heidelberg, 2013.
[21] E. Gibaja, S. Ventura, A tutorial on multilabel learning, ACM Comput. Surv. 47 (3) (2015) 1–38.
[22] D.D. Lewis, Y. Yang, T.G. Rose, F. Li, RCV1: a new benchmark collection for text categorization research, J. Mach. Learn. Res. 5 (2) (2004) 361–397.
[23] L. Zhang, D. Zhang, Evolutionary cost-sensitive extreme learning machine., IEEE Trans. Neural Netw. Learn. Syst. PP (99) (2016).
[24] L. Zhang, W. Zuo, D. Zhang, LSDT: Latent Sparse Domain Transfer Learning for Visual Adaptation[J], IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society 25 (3) (2016) 1177–1191.

Fangfang Luo received the B.S. and M.S. degrees from Institute of Mathematics and Computer Science, Fuzhou University, Fuzhou, China in 2003 and 2006, respectively. Currently, she is a Ph.D. candidate in the Institute of Physics and Information Engineering, Fuzhou University. Her current research interests include computational intelligence, big data analysis, and so on.

Wenzhong Guo received the B.S. and M.S. degrees in computer science and technology and the Ph.D. degree in communication and information system from Fuzhou University, Fuzhou, China, in 2000, 2003, and 2010, respectively. He completed the postdoctoral fellow at Institute of computer Science, National University of Defense and Technology, Changsha, China in 2013, and senior visiting scholar at Faculty of Engineering, Information and System, University of Tsukuba, Japan in 2013. He also is a member of ACM and IEEE. He is currently a professor at the Institute of Mathematics and Computer Science, Fuzhou University, and leads the Network Computing and Intelligent Information Processing Laboratory, which is a key laboratory of Fujian Province, China. He also is the deputy director of the Laboratory of Spatial Data Mining & Information Sharing, which is a key laboratory of ministry of education, China. He has published over 130 technical articles in scientific journals and conference proceedings. His current research interests include VLSI physical design, wireless sensor networks, big data, image processing, and so on.

Yuanlong Yu received the B.S. degrees in automatic control from the Beijing Institute of Technology, Beijing, China in 2000. The M.S. degree in computer applied technology from Tsinghua University, Beijing, China in 2003, and the Ph.D. degree in electrical engineering from the Memorial University of Newfoundland, St. John's, NL, Canada in 2010. He is currently professor in the College of Mathematics and Computer Sciences, Fuzhou University, China. His current research interests mainly focus on machine learning, computer vision and cognitive robotics.

Guolong Chen received the B.S. and M.S. degrees in computational mathematics from Fuzhou University, Fuzhou, China in 1987 and 1992, respectively, and the Ph.D. degree in computer science from Xian Jiaotong University, Xian, China, in 2002. He completed the postdoctoral fellow at Institute of Computer Science, National University of Defense and Technology, Changsha, China in 2007. He also is a member of ACM. He is currently a professor at the Institute of Mathematics and Computer Science, Fuzhou University. His current research interests include VLSI layout design, information security, big data analysis, computational intelligence, and so on.