Ml-rbf: RBF Neural Networks for Multi-Label Learning

Min-Ling Zhang[1,2]

[1] *College of Computer and Information Engineering, Hohai University, Nanjing 210098, China*

[2] *National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China*

Email: zhangml@hhu.edu.cn

Abstract. Multi-label learning deals with the problem where each instance is associated with multiple labels simultaneously. The task of this learning paradigm is to predict the label set for each unseen instance, through analyzing training instances with known label sets. In this paper, a neural network based multi-label learning algorithm named Ml-rbf is proposed, which is derived from the traditional radial basis function (RBF) methods. Briefly, the first layer of an Ml-rbf neural network is formed by conducting clustering analysis on instances of *each* possible class, where the centroid of each clustered groups is regarded as the prototype vector of a basis function. After that, second layer weights of the Ml-rbf neural network are learned by minimizing a sum-of-squares error function. Specifically, information encoded in the prototype vectors corresponding to all classes are fully exploited to optimize the weights corresponding to each specific class. Experiments on three real-world multi-label data sets show that Ml-rbf achieves highly competitive performance to other well-established multi-label learning algorithms.

Keywords. Machine learning, multi-label learning, radial basis function, k-means clustering

1 Introduction

Multi-label learning originated from the investigation of text categorization problems and has become one of the widely studied machine learning frameworks. Many real-world learning problems fall into the category of multi-label learning. For example, in text categorization, each document may belong to several predefined topics, such as *sports*, *Beijing Olympics* and even *torch relay* [24, 32]; In bioinformatics, each gene may be associated with a number of functional classes, such as *metabolism*, *transcription* and *protein synthesis* [11]; In automatic video annotation, each video clip may be related to several semantic classes, such as *urban* and *building* [27]. For multi-label learning problems, each instance in the training set is associated with a set of labels, and the task is to predict a label set for each unseen instance based on the model induced from multi-label training instances with known label sets.

Traditional supervised learning (binary or multi-class) can be regarded as the special cases of multi-label learning, where the labels associated with each instance are restricted to be unique. Recently, many multi-label learning algorithms have been proposed by adapting traditional supervised learning techniques to learn from multi-label training instances, such as multi-label decision trees [7, 8], multi-label bayesian approaches [24, 34], multi-label kernel methods [2, 11, 17, 20], etc. In this paper, a multi-label neural network algorithm named Ml-rbf (Multi-Label Radial Basis Function) is proposed, which is derived from the popular RBF methods [1]. Briefly, by performing k-means clustering on instances coming from each possible class, prototype vectors of first-layer basis functions are set to the centroids of clustered groups. After that, Ml-rbf's second-layer weights are learned by minimizing a sum-of-squares error function. In this way, information encoded in the prototype vectors corresponding to all classes are fully utilized when optimizing the weights connected to each output neuron (class). Applications to three real-world multi-label learning tasks, i.e. functional genomics, scene classification and web page categorization, show that Ml-rbf is superior to other well-established multi-label learning algorithms. Furthermore, Ml-rbf significantly outperforms Bp-mll in terms of both effectiveness and efficiency, which is another neural network based multi-label learner derived from the popular Backpropagation methods [30].

The rest of this paper is organized as follows. Section 2 reviews related works on multi-label learning. Section 3 proposes the Ml-rbf method. Section 4 reports the comparative experimental results. Finally, Section 5 summarizes and indicates the issues for future work.

2 Related Works

Many works on multi-label learning deal with text categorization problems. Schapire and Singer [32] proposed the famous BoosTexter approach by extending from the popular ensemble learning method AdaBoost [13], where a set of weights over all instance-label pairs are maintained and those pairs that are hard (easy) to predict correctly will get incrementally higher (lower) weights. McCallum [24] and Ueda and Saito [34] respectively proposed two multi-label text categorization approaches by assuming generative models based on text frequencies. McCallum [24] assumed that a mixture probabilistic model (one mixture component per category) is assumed to generate each document and utilized EM [10] algorithm to learn the mixture weights and the word distributions in each mixture component. Ueda and Saito [34] presented two types of probabilistic generative models for multi-label text called parametric mixture models (Pmm1, Pmm2). They assumed that multi-label text has a mixture of characteristic words appearing in single-label text that belongs to each category of the multi-categories. Gao et al. [15] proposed a maximal figure-of-merit (MFoM) approach [14] for multi-label text categorization, where classifier parameters are incorporated into a continuous and differentiable function which simulates specific performance metrics, and then learned by optimizing the specified function.

In addition, researchers have also proposed various multi-label text categorization algorithms by adapting traditional machine learning methods. Comité et al. [8] extended alternating decision tree [12] by employing it as the base learner of AdaBoost.MH [31] to train the multi-label classifier. Crammer and Singer [9] generalized the classical Perceptron algorithm [28] to learn from multi-topic documents, where a prototype is assigned to each possible topic and learned by an online style algorithm. In prediction, topics are ranked according to the prototypes' similarity with the vector representation of the document. Zhang and Zhou [36] designed multi-label Backpropagation algorithm through employing a novel error function reflecting the facts that labels belonging to an instance should be ranked higher than those not belonging to that instance. Ghamrawi and McCallum [16] and Zhu et al. [38] both extended the maximum entropy model [25] to deal with multi-label data by adding second order constraints capturing the correlations among different categories. Godbole and Sarawagi [17] extended the SVM-based text categorization method [18], where a kernel function is constructed on heterogeneous feature set comprising the original text features and artificially generated ones indicating relationships between classes. Kazawa et al. [20] solves multi-topic text categorization problem by converting it into a multi-class single-label one, where each topic set is regarded as a new class and labels are embedded into a similarity-induced vector space to cope with the data sparseness caused by the huge number of new classes.

In addition to text categorization, multi-label learning has manifested its effectiveness in bioinformatics. Clare and King [7] adapted C4.5 decision tree to handle multi-label gene expression data through modifying the definition of entropy. The learned decision tree model, equivalently a set of symbolic rules, is interpretable and can be compared with known biological knowledge. Elisseeff and Weston [11] defined a special multi-label margin and then presented a kernel implementation for multi-label classification with positive testing results on Yeast gene functional data. Brinker et al. [3] introduced *learning by pairwise comparison* techniques to multi-label scenario, where a virtual label is introduced to each instance acting as the split point between relevant and irrelevant labels. Boutell et al. [2] applied multi-label learning techniques to scene classification. They decomposed the multi-label learning problem into multiple independent binary classification problems and proposed a number of labeling criteria to combine prediction outputs from each binary classifier. Qi et al. [27] studied the problem of automatic multi-label video annotation. They transformed instances into high-dimensional vectors encoding correlations between inputs and outputs and proposed a maximum-margin type algorithm to learn from the transformed vectors.

Besides those eager-style learning algorithms, researchers have also proposed several lazy-style approaches where no training phase is involved and labels of each test instance are predicted based on its similarity with training instances [4, 19, 37]. Recently, several algorithms have been proposed to improving the performance of text categorization systems through exploring extra information provided by the hierarchical structure of classes [5, 29] or unlabeled data [6, 22]. Furthermore, multi-label learning techniques have also been applied to association rule mining [29, 33, 35].

3 Ml-rbf

Let $\mathcal{X} = \mathbb{R}^d$ be the input space and $\mathcal{L} = \{1, 2, \ldots, L\}$ be the finite set of L possible classes. Given a multi-label training set $\mathcal{D} = \{(\mathbf{x}_i, Y_i) | 1 \leq i \leq m\}$, where $\mathbf{x}_i \in \mathcal{X}$ is a single instance and $Y_i \subseteq \mathcal{L}$ is the label set associated with $\mathbf{x}_i$. The task of multi-label learning is to learn a multi-label classifier

Figure 1. Architecture of the ML-RBF neural network.

$h : \mathcal{X} \rightarrow 2^{\mathcal{L}}$ from $\mathcal{D}$ which predicts a set of labels for each unseen instance. Generally, instead of producing $h(\cdot)$, the learning system will output a real-valued function $f : \mathcal{X} \times \mathcal{L} \rightarrow \mathbb{R}$. Given $\mathbf{x}_i$ and its associated label set Y_i, $f(\cdot, \cdot)$ is supposed to have the property of $f(\mathbf{x}_i, l_1) > f(\mathbf{x}_i, l_2)$ for any $l_1 \in Y_i$ and $l_2 \notin Y_i$, i.e. yielding larger values for labels in Y_i than those not in Y_i. In addition, let $rank_f(\mathbf{x}_i, l)$ denote the ranking function derived from $f(\mathbf{x}_i, l)$ which returns the rank of l ($l \in \mathcal{L}$), i.e. the larger the value of $f(\mathbf{x}_i, l)$ the higher the rank of l. Note that the multi-label classifier $h(\cdot)$ can also be derived from $f(\cdot, \cdot)$ as: $h(\mathbf{x}_i) = \{l | f(\mathbf{x}_i, l) > t(\mathbf{x}_i),\ l \in \mathcal{L}\}$, where $t(\cdot)$ is a threshold function usually set to be the zero constant function.

Radial basis function (RBF) is one of the mostly studied neural network models, which is supposed to consist of two layers of neurons. Specifically, each hidden neuron (basis function) in the first layer is associated with a prototype vector while each output neuron corresponds to a possible class. Generally a two-stage procedure is employed to train an RBF neural network, where the basis functions are learned by performing clustering analysis on training instances and second-layer weights are optimized by solving a linear problem. Comprehensive descriptions on RBF neural networks are available at [1]. Next, we will present ML-RBF which adapts traditional RBF neural networks to learn from multi-label instances.

Figure 1 illustrates the architecture of a typical ML-RBF neural network. As it is shown, the input to an ML-RBF neural network corresponds to a d-dimensional feature vector. Furthermore, the hidden layer (i.e. first layer) of ML-RBF is composed of L sets of prototype vectors, i.e. $\bigcup_{l=1}^{L} \mathcal{C}_l$. Here, $\mathcal{C}_l$ consists of k_l prototype vectors $\{\mathbf{c}_1^l, \mathbf{c}_2^l, \ldots, \mathbf{c}_{k_l}^l\}$. In this paper, for each class

$l \in \mathcal{L}$, the popular k-means clustering[1] is performed on the set of instances U_l with label l, i.e. $U_l = \{\mathbf{x}_i | (\mathbf{x}_i, Y_i) \in \mathcal{D},\ l \in Y_i\}$. Thereafter, k_l clustered groups are formed for class l and the j-th centroid ($1 \leq j \leq k_l$) is regarded as a prototype vector $\mathbf{c}_j^l$ of basis function $\phi_j^l(\cdot)$. In this paper, k_l is set to be fraction α of the number of instances in U_l:

$$k_l = \alpha \times |U_l| \tag{1}$$

Each output neuron of the Ml-rbf neural network is related to a possible class. The weights $\mathbf{W} = [w_{jl}]_{(K+1)\times L}$ between hidden and output layers are shown as lines from basis functions to output neurons. Here, $K = \sum_{l=1}^{L} k_l$ denotes the total number of prototype vectors retained in the hidden layer. In addition, the *biases* are shown as weights w_{0l} from an extra basis function $\phi_0(\cdot)$ whose output is fixed at 1. Actually, the weight matrix $\mathbf{W}$ is learned by minimizing the following sum-of-squares error function:

$$E = \frac{1}{2}\sum_{i=1}^{m}\sum_{l=1}^{L}\left(y_l(\mathbf{x}_i) - t_l^i\right)^2 \tag{2}$$

where t_l^i is the desired output of $\mathbf{x}_i$ on the l-th class, which takes the value of $+1$ if $l \in Y_i$ and -1 otherwise. Correspondingly, $y_l(\mathbf{x}_i) = \sum_{j=0}^{L} w_{jl}\phi_j(\mathbf{x}_i)$ is the actual output of $\mathbf{x}_i$ on the l-th class. Here, all the $K+1$ basis functions $\{\phi_0(\cdot), \phi_1^1(\cdot), \ldots, \phi_{k_1}^1, \ldots, \phi_1^L(\cdot), \ldots, \phi_{k_L}^L(\cdot)\}$ in the hidden layer are put together and then re-indexed, where the j-th basis function is denoted as $\phi_j(\cdot)$ ($0 \leq j \leq K$). Similarly, all the K prototype vectors $\{\mathbf{x}_1^1, \ldots, \mathbf{x}_{k_1}^1, \ldots, \mathbf{x}_1^L, \ldots, \mathbf{x}_{k_L}^L\}$ can also be re-indexed with the j-th prototype denoted as $\mathbf{c}_j$. In this paper, the basis function $\phi_j(\cdot)$ makes the following widely-used Gaussian style activation:

$$\phi_j(\mathbf{x}_i) = \exp\left(-\frac{\mathrm{dist}(\mathbf{x}_i, \mathbf{c}_j)^2}{2\sigma_j^2}\right) \tag{3}$$

Here $\mathrm{dist}(\mathbf{x}_i, \mathbf{c}_j)$ measures the distance between $\mathbf{x}_i$ and the j-th prototype vector $\mathbf{c}_j$, which takes the usual Euclidean form. In addition, the activation of $\phi_0(\mathbf{x}_i)$ is fixed at 1 and w_{0l} acts as the bias value for the l-th class. In this paper, all the smoothing parameters σ_j ($1 \leq j \leq K$) take the same value of σ, which is set to be some multiple of the average distance between each pair of prototype vectors:

$$\sigma = \mu \times \left(\frac{\sum_{p=1}^{K-1}\sum_{q=p+1}^{K}\mathrm{dist}(\mathbf{c}_p, \mathbf{c}_q)}{K(K-1)/2}\right) \tag{4}$$

where μ is the parameter of scaling factor.

Differentiating the error function of Eq.(2) with respect to w_{jl} and setting the derivative to zero gives the following normal equations for the least sum-of-squares problem:

$$\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)\mathbf{W} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{T} \tag{5}$$

[1]Note that more sophisticated clustering algorithms, such as self-organizing maps [21], neural-gas [23], etc., could also be used. In this paper, however, Ml-rbf suffices to yield satisfactory performance by using k-means algorithm.

```
$Z$=ML-RBF($\mathcal{D}$, $\alpha$, $\mu$, $\mathbf{z}$)
**Inputs:**
$\mathcal{D}$: the multi-label training set $\{(\mathbf{x}_1, Y_1), \ldots, (\mathbf{x}_m, Y_m)\}$
$\alpha$: the fraction parameter as in Eq.(1)
$\mu$: the scaling factor as in Eq.(4)
$\mathbf{z}$: the test instance ($\mathbf{z} \in \mathcal{X}$)
**Outputs:**
$Z$: predicted label set for $\mathbf{z}$ ($Z \subseteq \mathcal{L}$)
**Process:**
1  for $l \in \mathcal{L}$ do
2      Set $U_l = \{\mathbf{x}_i | (\mathbf{x}_i, Y_i) \in \mathcal{D},\ l \in Y_i\}$;
3      Cluster $U_l$ into $k_l = \alpha \times |U_l|$ groups by invoking the $k$-means algorithm;
4      Set the centroids of all clustered groups as the prototype vectors for the $l$-th class;
5  Form matrix $\mathbf{\Phi}$ (using Eqs.(3) and (4)) and $\mathbf{T}$;
6  Compute weights $\mathbf{W}$ by solving Eq.(5) using SVD [26];
7  $Z = \{l | y_l(\mathbf{z}) = \sum_{j=0}^{K} w_{jl}\phi_j(\mathbf{z}) > 0,\ l \in \mathcal{L}\}$;
```

Figure 2. Pseudo code of ML-RBF.

Here $\mathbf{\Phi} = [\phi_{ij}]_{m\times(K+1)}$ with elements $\phi_{ij} = \phi_j(\mathbf{x}_i)$, $\mathbf{W} = [w_{jl}]_{(K+1)\times L}$ with elements w_{jl}, and $\mathbf{T} = [t_{il}]_{m\times L}$ with elements $t_{il} = t_l^i$. In this paper, the weight parameters $\mathbf{W}$ are computed by solving Eq.(5) using linear matrix inversion techniques of SVD [26].

One noticeable characteristic of ML-RBF is that each output neuron is connected with basis functions corresponding to all possible classes. In this way, for a specific class l, information encoded in prototype vectors of all classes will be fully exploited in either procedure of optimizing the weights connecting to l and making predictions for l. Therefore, the correlations between different classes are appropriately addressed in both ML-RBF's training and testing phases.

Figure 2 gives the complete description of ML-RBF. Firstly, the hidden layer of ML-RBF neural network is constitued by performing k-means clustering on training instances of each possible class (steps 1 to 4); After that, the weights between hidden and output layers are determined through minimizing the sum-of-squares error function as shown in Eq.(2) (steps 5 to 6); Finally, the test multi-label instance is fed to the trained neural network for prediction (step 7).

4 Experiments

4.1 Experimental Setup

Given a multi-label test set $\mathcal{Z} = \{(\mathbf{z}_i, Z_i)|1 \leq i \leq n\}$, based on the notations in Section 3, the following popular multi-label evaluation metrics [32, 36] are utilized:

1) *Hamming Loss*:

$$\text{hloss}_{\mathcal{Z}}(h) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{L}|h(\mathbf{z}_i)\Delta Z_i|$$

where Δ stands for the symmetric difference between two sets.

2) *One-error*:

$$\text{one-error}_{\mathcal{Z}}(f) = \frac{1}{n}\sum_{i=1}^{n}[\![[\arg\max_{l\in\mathcal{L}} f(\mathbf{z}_i, l)] \notin Z_i]\!]$$

where for any predicate π, $[\![\pi]\!]$ equals 1 if π holds and 0 otherwise.

3) *Coverage*:

$$\text{coverage}_{\mathcal{Z}}(f) = \frac{1}{n}\sum_{i=1}^{n}\max_{l\in Z_i} rank_f(\mathbf{z}_i, l) - 1$$

4) *Ranking Loss*:

$$\text{rloss}_{\mathcal{Z}}(f) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Z_i||\overline{Z_i}|}|\{(l_1, l_2)|f(\mathbf{z}_i, l_1) \leq f(\mathbf{z}_i, l_2),\ (l_1, l_2) \in Z_i \times \overline{Z_i}\}|$$

where $\overline{Z_i}$ denotes the complementary set of Z_i in $\mathcal{L}$.

5) *Average Precision*:

$$\text{avgprec}_{\mathcal{Z}}(f) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|Z_i|}\sum_{l\in Z_i}\frac{|\{l'|rank_f(\mathbf{z}_i, l') \leq rank_f(\mathbf{z}_i, l),\ l' \in Z_i\}|}{rank_f(\mathbf{z}_i, l)}$$

In words, *hamming loss* evaluates how many times an instance-label pair is misclassified; *One-error* evaluates how many times the top-ranked label is not in the set of proper labels of the instance; *Coverage* evaluates how many steps are needed, on average, to move down the label list in order to cover all the proper labels of the instance; *Ranking loss* evaluates the average fraction of label pairs that are reversely ordered for the instance; *Average precision* evaluates the average fraction of labels ranked above a particular label $l \in Z$ which actually are in Z. Note that for the first four metrics, the *smaller* the metric value the better the performance. While for *average precision*, the *bigger* the metric value the better the performance.

In this paper, Ml-rbf is compared with several methods including BoosTexter [32], Adtboost.MH [8] and Rank-svm [11], which are all state-of-the-art algorithms specifically designed for multi-label learning problems. Moreover, Bp-mll [36], which is also a neural network style multi-label learning algorithm is also compared with Ml-rbf. For BoosTexter[2] and Adtboost.MH[3], the number of boosting rounds is set to be 500 and 50 respectively as their performance will not significantly change after the specified boosting rounds. For Rank-svm, the best parameters reported in [11], i.e. polynomial kernels with degree 8, are used. For Bp-mll, as indicated in the literature [36], the number of of hidden neurons is set to be 20% of the number of input neurons, i.e. the dimensionality d of input space $\mathcal{X}$. Furthermore, the number of training epochs is fixed to be 100 with learning rate of 0.05. In Subsection 4.3, parameter choices for Ml-rbf will be scrutinized.

4.2 Data Sets

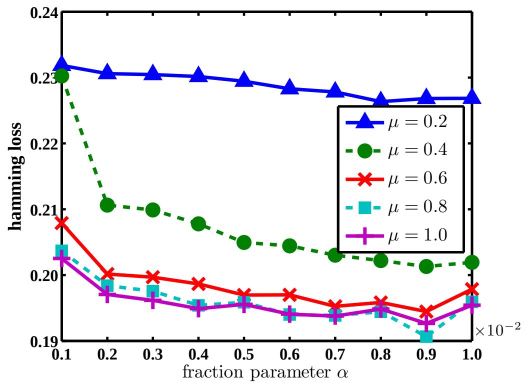
In this paper, three real-world data sets are employed to extensively evaluate the performance of each compared algorithm:

- *Yeast Gene Functional Analysis*: The first data set is the Yeast data investigated in [11, 36, 37], whose task is to predict the gene functional classes of the Yeast *Saccharomyces cerevisiae.* Each gene is described by concatenating the micro-array expression data and phylogenetic profile. Furthermore, each gene is associated with a set of functional classes, where the whole set of classes is structured into hierarchies of 4 levels deep and the maximum size can be even larger than 190. In this paper however, to simplify the problem as in [11], only functional classes in the top hierarchy are considered. The resultant data set contains 2,417 genes each represented by a 103-dimensional feature vector. There are 14 possible class labels and the average number of labels for each gene is 4.24 ± 1.57. More detailed descriptions on this data set are available in [36, 37].

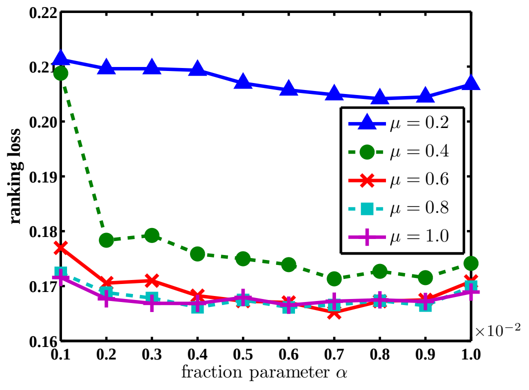
- *Natural Scene Classification*: The second data set is the natural scene classification data investigated in [37], whose task is to predict the label set for unseen image through analyzing images with known label sets. This data set is composed of 2,000 natural scene images. All the possible classes are *desert*, *mountains*, *sea*, *sunset* and *trees* and a set of labels is manually assigned to each image. The number of images belonging to more than one class (e.g. *sea+sunset*) comprises over 22% of the data set, several combined classes (e.g. *mountains+sunset+trees*) are extremely rare. The average number of labels for each image is 1.24 ± 0.44. Furthermore, each image is represented by a 294-dimensional feature vector adopting the same method used by Boutell et al. [2]. More detailed descriptions on this data set are available in [37].

- *Automatic Web Page Categorization*: The third data set is the WWW page categorization

[2]http://www.cs.princeton.edu/˜schapire/boostexter.html
[3]http://www.grappa.univ-lille3.fr/grappa/index.php3?info=logiciels.



(a) *hamming loss*



(b) *ranking loss*

Figure 3. The ten-fold cross-validation performance of ML-RBF on the Yeast data changes as α increases with fixed value of μ. In both sub-figures, the *lower* the curve the better the performance.

data investigated in [20, 34, 37], whose task is to predict a set of labels for unseen web pages by learning from those with known label sets. Web pages are collected from the "yahoo.com" domain, which consists of 14 top-level categories (e.g. "Computers & Internet", "Recreation & Sports", etc.). Each top-level category is further classified into a number of second-level categories. By focusing on the second-level categories, 11 (out of 14) different web page categorization problems are identified. Each problem contains 2,000 training web pages and 3,000 test web pages, where a considerable portion of them are multi-labeled over the 11 problems (about 20% to 45%). Each web page is described as a feature vector using the preprocessing method as in [37]. Over the 11 problems, the number of categories varies from 21 to 40 and the instance dimensionality varies from 438 to 1,047. More detailed descriptions on this data set are available in [20, 37].

4.3 Experimental Results

As shown in Figure 2, there are two different parameters governing the ML-RBF algorithm, i.e. the fraction parameter α and the scaling factor μ. To illustrate the sensitivity of ML-RBF with respect to these parameters, Figure 3 illustrates how ML-RBF performs on the Yeast data with ten-fold cross-validation, under different parameter configurations. Here, the performance is evaluated in terms of *hamming loss* and *ranking loss*[4]. Specifically, α increases from 1‰ to 1% with an interval of 1‰, while μ varies from 0.2 to 1.0 with an interval of 0.2. It is evident from Figure 3 that, when the scaling factor μ is fixed, the performance of ML-RBF improves remarkably in the initial increasing phase of α ($\alpha <$ 3‰). After that, ML-RBF tends to perform stably in the remaining increasing phase of α. On the other hand, when the fraction parameter α is fixed, the performance of ML-RBF will not significantly vary as long as μ exceeds 0.4. In the rest of this paper, instead of

[4]Observations on the other three metrics would give rise to similar conclusions.

Table 1. Performance of each compared algorithm (mean±std. deviation) on Yeast data.

Evaluation	Algorithm				
Criterion	ML-RBF	BP-MLL	BOOSTEXTER	ADTBOOST.MH	RANK-SVM
hamming loss$^\downarrow$	**0.195±0.011**	0.206±0.011	0.220±0.011	0.207±0.010	0.207±0.013
one-error$^\downarrow$	**0.233±0.037**	**0.233±0.034**	0.278±0.034	0.244±0.035	0.243±0.039
coverage$^\downarrow$	**6.352±0.244**	6.421±0.237	6.550±0.243	6.390±0.203	7.090±0.503
ranking loss$^\downarrow$	**0.169±0.017**	0.171±0.015	0.186±0.015	N/A	0.195±0.021
average precision$^\uparrow$	**0.766±0.021**	0.756±0.021	0.737±0.022	0.744±0.025	0.749±0.026

Table 2. Performance of each compared algorithm (mean±std. deviation) on image data.

Evaluation	Algorithm				
Criterion	ML-RBF	BP-MLL	BOOSTEXTER	ADTBOOST.MH	RANK-SVM
hamming loss$^\downarrow$	**0.163±0.015**	0.278±0.014	0.179±0.015	0.193±0.014	0.253±0.055
one-error$^\downarrow$	**0.294±0.033**	0.535±0.047	0.311±0.041	0.375±0.049	0.491±0.135
coverage$^\downarrow$	**0.904±0.087**	1.416±0.107	0.939±0.092	1.102±0.111	1.382±0.381
ranking loss$^\downarrow$	**0.158±0.020**	0.288±0.023	0.168±0.020	N/A	0.278±0.096
average precision$^\uparrow$	**0.809±0.021**	0.658±0.024	0.798±0.024	0.755±0.027	0.682±0.093

Table 3. Performance of each compared algorithm (mean±std. deviation) on web page data.

Evaluation	Algorithm				
Criterion	ML-RBF	BP-MLL	BOOSTEXTER	ADTBOOST.MH	RANK-SVM
hamming loss$^\downarrow$	**0.039±0.013**	0.051±0.016	0.046±0.016	0.043±0.013	0.043±0.014
one-error$^\downarrow$	**0.376±0.120**	0.524±0.150	0.446±0.139	0.461±0.137	0.440±0.143
coverage$^\downarrow$	4.644±1.344	**3.981±1.142**	4.213±1.313	4.083±1.191	7.508±2.396
ranking loss$^\downarrow$	0.106±0.038	**0.094±0.031**	0.103±0.040	N/A	0.193±0.065
average precision$^\uparrow$	**0.688±0.092**	0.609±0.104	0.638±0.108	0.632±0.105	0.605±0.117

optimizing parameters of the RBF neural networks by schemes such as gradient descent, ML-RBF is implemented with fixed parameters of $\alpha = 0.01$ and $\mu = 1.0$.

Tables 1 to 3 summarize the experimental results of each compared algorithm on the Yeast, image and web page data sets respectively. Note that for either the Yeast or image data, ten-fold cross-validation is performed on them and the performance (mean±std. deviation) out of ten independent runs are reported. For the web page data however, the performance (mean±std. deviation) out of the 11 different web page categorization problems are reported. As shown in the tables, for each evaluation metric, "↓" indicates "the *smaller* the better" while "↑" indicates "the *bigger* the better". Furthermore, the best result on each metric is highlighted in bold face[5].

To better characterize the relative performance between different algorithms, Tables 4 to 6 also present the comparative results based on statistical test on each data set respectively. As shown in the tables, for each evaluation metric, $\mathcal{A} \succ \mathcal{B}$ indicates that the performance of algorithm $\mathcal{A}$ is statistically significantly better than that of algorithm $\mathcal{B}$ based on pairwise t-test at 5% significance

[5]Note that *ranking loss* is not provided in the outputs of the ADTBOOST.MH implementation.

Table 4. Relative performance between each compared algorithm on Yeast data.

Evaluation	Algorithm
Metric	$\mathcal{A}$1-ML-RBF; $\mathcal{A}$2-BP-MLL; $\mathcal{A}$3-BOOSTEXTER; $\mathcal{A}$4-ADTBOOST.MH; $\mathcal{A}$5-RANK-SVM
hamming loss	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}2 \succ \mathcal{A}3$, $\mathcal{A}4 \succ \mathcal{A}3$, $\mathcal{A}5 \succ \mathcal{A}3$
one-error	$\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}2 \succ \mathcal{A}3$, $\mathcal{A}4 \succ \mathcal{A}3$, $\mathcal{A}5 \succ \mathcal{A}3$
coverage	$\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}2 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}5$, $\mathcal{A}4 \succ \mathcal{A}3$, $\mathcal{A}4 \succ \mathcal{A}5$
ranking loss	$\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}2 \succ \mathcal{A}3$, $\mathcal{A}2 \succ \mathcal{A}5$
average precision	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}2 \succ \mathcal{A}3$, $\mathcal{A}2 \succ \mathcal{A}4$
Total Order	ML-RBF(**13**)>BP-MLL(**5**)>ADTBOOST.MH(**1**)>RANK-SVM(**−6**)>BOOSTEXTER(**−13**)

Table 5. Relative performance between each compared algorithm on image data.

Evaluation	Algorithm
Metric	$\mathcal{A}$1-ML-RBF; $\mathcal{A}$2-BP-MLL; $\mathcal{A}$3-BOOSTEXTER; $\mathcal{A}$4-ADTBOOST.MH; $\mathcal{A}$5-RANK-SVM
hamming loss	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}2$, $\mathcal{A}3 \succ \mathcal{A}4$, $\mathcal{A}3 \succ \mathcal{A}5$, $\mathcal{A}4 \succ \mathcal{A}2$, $\mathcal{A}4 \succ \mathcal{A}5$
one-error	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}2$, $\mathcal{A}3 \succ \mathcal{A}4$, $\mathcal{A}3 \succ \mathcal{A}5$, $\mathcal{A}4 \succ \mathcal{A}2$, $\mathcal{A}4 \succ \mathcal{A}5$
coverage	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}2$, $\mathcal{A}3 \succ \mathcal{A}4$, $\mathcal{A}3 \succ \mathcal{A}5$, $\mathcal{A}4 \succ \mathcal{A}2$, $\mathcal{A}4 \succ \mathcal{A}5$
ranking loss	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}2$, $\mathcal{A}3 \succ \mathcal{A}5$
average precision	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}2$, $\mathcal{A}3 \succ \mathcal{A}4$, $\mathcal{A}3 \succ \mathcal{A}5$, $\mathcal{A}4 \succ \mathcal{A}2$, $\mathcal{A}4 \succ \mathcal{A}5$
Total Order	ML-RBF(**18**)>BOOSTEXTER(**10**)>ADTBOOST.MH(**0**)>{BP-MLL(**−14**), RANK-SVM(**−14**)}

Table 6. Relative performance between each compared algorithm on web page data.

Evaluation	Algorithm
Metric	$\mathcal{A}$1-ML-RBF; $\mathcal{A}$2-BP-MLL; $\mathcal{A}$3-BOOSTEXTER; $\mathcal{A}$4-ADTBOOST.MH; $\mathcal{A}$5-RANK-SVM
hamming loss	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}2$, $\mathcal{A}4 \succ \mathcal{A}2$, $\mathcal{A}4 \succ \mathcal{A}3$, $\mathcal{A}5 \succ \mathcal{A}2$, $\mathcal{A}5 \succ \mathcal{A}3$
one-error	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}2$, $\mathcal{A}3 \succ \mathcal{A}4$, $\mathcal{A}4 \succ \mathcal{A}2$, $\mathcal{A}5 \succ \mathcal{A}2$
coverage	$\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}2 \succ \mathcal{A}1$, $\mathcal{A}2 \succ \mathcal{A}3$, $\mathcal{A}2 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}1$, $\mathcal{A}3 \succ \mathcal{A}5$, $\mathcal{A}4 \succ \mathcal{A}1$, $\mathcal{A}4 \succ \mathcal{A}5$
ranking loss	$\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}2 \succ \mathcal{A}1$, $\mathcal{A}2 \succ \mathcal{A}3$, $\mathcal{A}2 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}5$
average precision	$\mathcal{A}1 \succ \mathcal{A}2$, $\mathcal{A}1 \succ \mathcal{A}3$, $\mathcal{A}1 \succ \mathcal{A}4$, $\mathcal{A}1 \succ \mathcal{A}5$, $\mathcal{A}3 \succ \mathcal{A}2$, $\mathcal{A}3 \succ \mathcal{A}5$
Total Order	ML-RBF(**10**)>{BOOSTEXTER(**1**), ADTBOOST.MH(**1**)}>BP-MLL(**−4**)>RANK-SVM(**−8**)

level. Note that $\mathcal{A} \succ \mathcal{B}$ only measures the relative performance between $\mathcal{A}$ and $\mathcal{B}$ on one specific metric. However, $\mathcal{A}$ would probably perform better than $\mathcal{B}$ in terms of some metrics while worse than $\mathcal{B}$ in terms of other ones. Taking the above situation into consideration, a score is assigned to each compared algorithm in order to give an *overall* performance assessment of the algorithm. Concretely, for each evaluation metric and each possible pair of algorithms $\mathcal{A}$ and $\mathcal{B}$, if $\mathcal{A} \succ \mathcal{B}$ holds, the score of $\mathcal{A}$ is added by 1 and that of $\mathcal{B}$ is subtracted by 1 accordingly. With the accumulated score of each algorithm, a total order "$>$" can be defined on the set of all compared algorithms as shown in the last lines of Tables 4 to 6. Here, $\mathcal{A} > \mathcal{B}$ denotes that $\mathcal{A}$ outperforms $\mathcal{B}$ on the corresponding data set and the accumulated score of each algorithm is reported in the parentheses.

Table 4 shows that, on the Yeast data, ML-RBF achieves rather competitive performance where on all evaluation metrics no algorithm has ever outperformed ML-RBF. Furthermore, ML-RBF significantly outperforms all the other compared algorithms in terms of *hamming loss* and *average*

Table 7. Computation time of Ml-rbf and Bp-mll (mean±std. deviation) on each data. Training time and testing time are measured in *hours* and *seconds* respectively.

Data	Training phase (Hours)		Testing phase (Seconds)	
Set	Ml-rbf	Bp-mll	Ml-rbf	Bp-mll
Yeast data	0.014±0.004	6.980±0.235	3.873±0.641	0.739±0.037
Image data	0.003±0.001	2.974±0.171	0.386±0.266	2.856±0.864
Web page data	0.054±0.016	6.078±1.664	42.401±14.458	10.805±2.134

precision. On the whole (as shown by the total order), Ml-rbf is substantially superior to all the other compared algorithms on the Yeast data.

Table 5 shows that, on the image data, Ml-rbf achieves even more impressive performance as only in terms of *coverage*, Ml-rbf is comparable to BoosTexter. While for other cases, Ml-rbf outperforms all the other compared algorithms in terms of all evaluation metrics. On the whole (as shown by the total order), Ml-rbf also outperforms other well-established algorithms on the image data.

Table 6 shows that, on the web page data, Ml-rbf still performs well compared to other state-of-the-art algorithms. Ml-rbf is only inferior to Bp-mll, BoosTexter and Adtboost.MH in terms of *coverage* and inferior to Bp-mll in terms of *ranking loss*. While on the other hand, Ml-rbf significantly outperforms all the other compared algorithms in terms of *hamming loss*, *one-error* and *average precision*. Note that although Bp-mll performs quite well in terms of *coverage* and *ranking loss*, it achieves almost the worst performance in terms of other three evaluation metrics. On the whole (as shown by the total order), Ml-rbf still performs better than all the other compared algorithms on the web page data.

Tables 4 to 6 reveal that Ml-rbf is superior to another neural network based multi-label algorithm, i.e. Bp-mll, in terms of algorithmic *effectiveness*. Table 7 further compares the algorithmic *efficiency* between Ml-rbf and Bp-mll, where all the experiments are conducted on an HP Server with 4G RAM and four Intel Xeron$^{\text{TM}}$ CPUs each running at 2.80GHz. Table 7 shows that the testing time required by either algorithm only differs in a few seconds. While the time consumed in the training phase of Bp-mll is much more than that of Ml-rbf (at least 100 times). These results confirm that Ml-rbf would be a better choice than Bp-mll to solve multi-label problems with respect to both effectiveness and efficiency.

5 Conclusion

In this paper, a novel approach to multi-label learning named Ml-rbf is proposed by adapting the popular RBF neural networks. Basis functions in Ml-rbf's first layer are constituted by performing k-means clustering on instances coming from each possible class. After that, second layer weights of Ml-rbf are optimized through minimizing a sum-of-squares error function. Correlations between different classes are appropriately addressed in both Ml-rbf's training and testing phases. Comparative studies on bioinformatic, image and text data sets show that Ml-rbf achieves rather competitive performance to other state-of-the-art multi-label learning algorithms.

Investigate whether better performance can be achieved by setting different fraction parameters (i.e. α in Eq.(1)) for each possible class is an interesting future work. In addition, enhancing the generalization abilities of multi-label learning algorithms with the help of other techniques, such as ensemble learning, is worth further study.

Acknowledgement

The author wish to thank the anonymous reviewers for their helpful comments. This work was supported by the National Science Foundation of China (60805022), 863 High-Tech Program of China (2007AA01Z178), Open Foundation of National Key Laboratory for Novel Software Technology of China (KFKT2008B12), and Startup Foundation for Excellent New Faculties of Hohai University.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[2] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[3] K. Brinker, J. Fürnkranz, and E. Hüllermeier. A unified model for multilabel classification and ranking. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pages 489–493, Riva del Garda, Italy, 2006.

[4] K. Brinker and E. Hüllermeier. Case-based multilabel ranking. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 702–707, Hyderabad, India, 2007.

[5] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, pages 78–87, Washington, D.C., 2004.

[6] G. Chen, Y. Song, F. Wang, and C. Zhang. Semi-supervised multi-label learning by solving a sylvester equation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 410–419, Atlanta, GA, 2008.

[7] A. Clare and R. D. King. Knowledge discovery in multi-label phenotype data. In L. De Raedt and A. Siebes, editors, *Lecture Notes in Computer Science 2168*, pages 42–53. Springer, Berlin, 2001.

[8] F. D. Comité, R. Gilleron, and M. Tommasi. Learning multi-label altenating decision tree from texts and data. In P. Perner and A. Rosenfeld, editors, *Lecture Notes in Computer Science 2734*, pages 35–49. Springer, Berlin, 2003.

[9] K. Crammer and Y. Singer. A new family of online algorithms for category ranking. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 151–158, Tampere, Finland, 2002.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society -B*, 39(1):1–38, 1977.

[11] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press, Cambridge, MA, 2002.

[12] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proceedings of the 16th International Conference on Machine Learning*, pages 124–133, Bled, Slovenia, 1999.

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[14] S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua. A maximal figure-of-merit learning approach to text categorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–181, Toronto, Canada, 2003.

[15] S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua. A MFoM learning approach to robust multiclass multi-label text categorization. In *Proceedings of the 21st International Conference on Machine Learning*, pages 329–336, Banff, Canada, 2004.

[16] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 195–200, Bremen, Germany, 2005.

[17] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In H. Dai, R. Srikant, and C. Zhang, editors, *Lecture Notes in Artificial Intelligence 3056*, pages 22–30. Springer, Berlin, 2004.

[18] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, Germany, 1998.

[19] F. Kang, R. Jin, and R. Sukthankar. Correlated label propagation with application to multi-label learning. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1719–1726, New York, NY, 2006.

[20] H. Kazawa, T. Izumitani, H. Taira, and E. Maeda. Maximal margin labeling for multi-topic text categorization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 649–656. MIT Press, Cambridge, MA, 2005.

[21] T. Kohonen. *Self-Orgnanizing Maps.* Springer, Berlin, 2nd edition, 1997.

[22] Y. Liu, R. Jin, and L. Yang. Semi-supervised multi-label learning by constrained non-negative matrix factorization. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 421–426, Boston, MA, 2006.

[23] T. M. Martinetz and K. J. Schulten. A "neural-gas" network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402. North-Holland, Amsterdam, 1991.

[24] A. McCallum. Multi-label text classification with a mixture model trained by EM. In *Working Notes of the AAAI'99 Workshop on Text Learning*, Orlando, FL, 1999.

[25] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, Stockholm, Sweden, 1999.

[26] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: the Art of Scientific Computing.* Cambridge University Press, New York, 1992.

[27] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th ACM International Conference on Multimedia*, pages 17–26, Augsburg, Germany, 2007.

[28] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

[29] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Learning hierarchical multi-category text classifcation models. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 774–751, Bonn, Germany, 2005.

[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press, Cambridge, MA, 1986.

[31] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the 11th Annual Conference on Computational Learning Theory*, pages 80–91, Madison, WI, 1998.

[32] R. E. Schapire and Y. Singer. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[33] F. A. Thabtah, P. I. Cowling, and Y. Peng. MMAC: a new multi-class, multi-label associative classification approach. In *Proceedings of the 4th IEEE International Conference on Data Mining*, pages 217–224, Brighton, UK, 2004.

[34] N. Ueda and K. Saito. Parametric mixture models for multi-label text. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 721–728. MIT Press, Cambridge, MA, 2003.

[35] A. Veloso, M. Jr. Wagner, M. Gonçalves, and M. Zaki. Multi-label lazy associative classification. In J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, editors, *Lecture Notes in Artificial Intelligence 4702*, pages 605–612. Springer, Berlin, 2007.

[36] M.-L. Zhang and Z.-H. Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[37] M.-L. Zhang and Z.-H. Zhou. Ml-knn: a lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[38] S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labelled classification using maximum entropy method. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 274–281, Salvador, Brazil, 2005.