A Study on Threshold Selection for Multi-label Classification

Rong-En Fan and Chih-Jen Lin*

Abstract

Multi-label classification is useful for text categorization, multimedia retrieval, and many other areas. A commonly used multi-label approach is the binary method, which constructs a decision function for each label. For some applications, adjusting thresholds in decision functions of the binary method significantly improves the performance, but few studies have been done on this subject. This article gives a detailed investigation on the selection of thresholds. Experiments on real-world data sets demonstrate the usefulness of some simple selection strategies.

1 Introduction

Recently, multi-label classification arises in many applications. In contrast to traditional data classification, where each instance is assigned to only one label, for multi-label classification, one instance may be simultaneously relevant to several labels. For example, in text categorization, one document can belong to multiple topics. Such problems are also common in bioinformatics.

Given training instances $\mathbf{x}_1, \ldots, \mathbf{x}_l \in R^n$ and d labels $1, \ldots, d$, a multi-label data set has that each instance $\mathbf{x}_i$ is relevant to a subset of the d labels. We represent this subset of labels by a binary vector $\mathbf{y}^i = [y_1^i, \ldots, y_d^i] \in \{0,1\}^d$, where $y_j^i = 1$ if and only if $\mathbf{x}_i$ is associated with label j. For a new instance $\mathbf{x}$, the goal of multi-label classification is to predict possible labels through the model built upon training instances. In contrast, traditional multi-class classification assumes that each instance is related to only a single label, and the model predicts the most probable one. Therefore, multi-label classification is an extension of multi-class classification.

There are many approaches for multi-label classification. For example, Yang (1999); Joachims (1998); McCallum (1999); Shapire and Singer (2000); Clare and King (2001); Elisseeff and Weston (2002); Comité et al. (2003); Ueda and Saito (2003); Ghamrawi and McCallum (2005); Kazawa et al. (2005); Zhang and Zhou (2007). The most popular way may be the binary method, which is simple and effective. For each label, it trains a classifier by using data associated with this label as positive and all others as negative. Thus, if there are d labels, this method constructs d decision functions. In prediction, an instance is associated with a label if the corresponding decision value is positive. In this paper, we will mainly investigate the binary method.

It is known that for each decision function of the binary method, we can adjust the corresponding threshold as the cutting point for positive/negative predictions. Yang

*Department of Computer Science, National Taiwan University, Taipei 106, Taiwan, b90098@csie.ntu.edu.tw and cjlin@csie.ntu.edu.tw

(2001); Lewis et al. (2004) successfully apply thresholding techniques to obtain better performance for document data, but so far few studies have been done on this subject. Zhang and Oles (2001) list this as one important open issue in document classification. In this article, we conduct a detailed study using various multi-label data sets. We give some new observations and analysis.

This article is organized as follows. We introduce the binary method, evaluation measures for multi-label classification, and the basic concept of optimizing evaluation measures in Section 2. Section 3 discusses the thresholding schemes for the binary method. We describe data and experimental settings in Section 4. Results and some interesting analysis are in Section 5. Finally, we give concluding remarks in Section 6.

2 Binary Method, Evaluation Measures, and Optimization of Measures

In this section, we introduce the binary method to solve multi-label problems. Then, we discuss various evaluation criteria for multi-label problems. We also investigate how to optimize these criteria.

2.1 The Binary Method

This simple and commonly used multi-label approach has been used in, for example, Karalič et al. (1991); Joachims (1998); Yang (1999); Nigam et al. (2000). It is an extension of the "one-against-the rest" method (Bottou et al., 1994) for multi-class classification. Given l training instances $\mathbf{x}_1, \ldots, \mathbf{x}_l \in R^n$ and d labels, we train d decision functions:

$$\begin{aligned} f_1(\mathbf{x}) &= \mathbf{w}_1^T \mathbf{x} + b_1, \\ &\vdots \\ f_d(\mathbf{x}) &= \mathbf{w}_d^T \mathbf{x} + b_d. \end{aligned} \tag{1}$$

The decision function $f_j(\mathbf{x})$ is constructed by using training instances relevant to label j as positive and other training instances as negative. Taking support vector machines (SVMs) as an example, the jth decision function comes from solving the following optimization problem:

$$\begin{aligned} \min_{\mathbf{w}_j, b_j, \xi_j^i} \quad & \frac{1}{2}\mathbf{w}_j^T \mathbf{w}_j + C \sum_{i=1}^{l} \xi_j^i \\ \text{subject to} \quad & \mathbf{w}_j^T \phi(\mathbf{x}_i) + b_j \geq 1 - \xi_j^i, \text{ if } y_j^i = 1, \\ & \mathbf{w}_j^T \phi(\mathbf{x}_i) + b_j \leq -1 + \xi_j^i, \text{ if } y_j^i = 0, \\ & \xi_j^i \geq 0, \ i = 1, \ldots, l, \end{aligned} \tag{2}$$

where C is the penalty parameter, and y_j^i is an element of the binary vector $\mathbf{y}^i = [y_1^i, \ldots, y_d^i]^T \in \{0,1\}^d$ indicating labels of the ith instance. Note that in (2), SVM maps data to a higher dimensional space by a function ϕ.

For the prediction, label j is associated with $\mathbf{x}$ if and only if $f_j(\mathbf{x}) > 0$. For example, if there are $d = 3$ labels and we have the following decision values for an instance $\mathbf{x}$:

$$f_1(\mathbf{x}) > 0, \qquad f_2(\mathbf{x}) < 0, \qquad f_3(\mathbf{x}) > 0,$$

then we predict that the instance $\mathbf{x}$ has labels 1 and 3. In contrast, in multi-class classification, $\arg\max_j f_j(\mathbf{x})$ is often used to predict the most probable label.

We introduce a threshold T_j, so that an instance $\mathbf{x}$ is associated with label j if and only if

$$f_j(\mathbf{x}) = \mathbf{w}_j^T \phi(\mathbf{x}) + b_j > T_j. \tag{3}$$

For the binary method, clearly, $T_j = 0$. Later we will adjust T_j to have better predictions.

In this paper, we assume that SVM is used, but most results apply to other classification methods whose decision functions are in the same form as (1).

2.2 Evaluation Criteria

In traditional classification, the standard evaluation criterion is the accuracy. In multi-label classification, a direct extension of the accuracy is the exact match ratio, which considers one instance as correct if and only if all associated labels are correctly predicted. However, this ratio may not be the most suitable performance measure as it does not count partial matches. For example, in text categorization, one news story may belong to both sports and politics. If one states that this story is related to only sports, the statement is partially correct. Tague (1981) proposes two different criteria based on the F-measure: macro-average and micro-average F-measures. Both consider partial matches.

2.2.1 Exact Match Ratio

Assume there are $\bar{l}$ testing instances. Let $\mathbf{y}^i$ be the true label vector of the ith instance and $\hat{\mathbf{y}}^i$ be the predicted label vector, and the indicator function be

$$I[s] \equiv \begin{cases} 1 & \text{if } s \text{ is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

This measure (used in, for example, Kazawa et al., 2005) is defined as

$$\frac{1}{\bar{l}} \sum_{i=1}^{\bar{l}} I[\hat{\mathbf{y}}^i = \mathbf{y}^i]. \tag{5}$$

It is a direct extension of the accuracy for traditional data classification. A disadvantage of this criterion is that it does not take partial matches into account.

2.2.2 Macro-average and Micro-average F-measures

One of the most used performance measures for information retrieval systems is the F-measure, which is the harmonic mean of precision (P) and recall (R):

$$\frac{2PR}{P+R}.$$

Procedure 1 A simple way to optimize measures by sequentially adjusting one decision function.

1. Initialize $f_1^0, f_2^0, \ldots, f_d^0$ by running the binary method.
2. For $k = 1, \ldots, d$,
   (a) Solve the sub-problem with respect to the decision function f_k:
   $$\max_{f_k} \quad m(f_1^1, \ldots, f_{k-1}^1, f_k, f_{k+1}^0, \ldots, f_d^0). \tag{8}$$
   (b) Set f_k^1 to the solution of (8).
3. Return $f_1^1, f_2^1, \ldots, f_d^1$ and the final measure value is
   $$m\left(f_1^1, f_2^1, \ldots, f_d^1\right).$$

For label j,

$$P = \frac{\sum_{i=1}^{\bar{l}} \hat{y}_j^i y_j^i}{\sum_{i=1}^{\bar{l}} \hat{y}_j^i} \quad \text{and} \quad R = \frac{\sum_{i=1}^{\bar{l}} \hat{y}_j^i y_j^i}{\sum_{i=1}^{\bar{l}} y_j^i},$$

so the F-measure is

$$\frac{2\sum_{i=1}^{\bar{l}} \hat{y}_j^i y_j^i}{\sum_{i=1}^{\bar{l}} \hat{y}_j^i + \sum_{i=1}^{\bar{l}} y_j^i}.$$

To extend the F-measure from single-label to multi-label, two approaches are developed in Tague (1981). The macro-average F-measure is the unweighted mean of label F-measures,

$$\frac{1}{d}\sum_{j=1}^{d} \frac{2\sum_{i=1}^{\bar{l}} \hat{y}_j^i y_j^i}{\sum_{i=1}^{\bar{l}} \hat{y}_j^i + \sum_{i=1}^{\bar{l}} y_j^i}. \tag{6}$$

The other measure is the micro-average F-measure, which considers predictions from all instances together and calculate the F-measure across all labels:

$$\frac{2\sum_{j=1}^{d}\sum_{i=1}^{\bar{l}} \hat{y}_j^i y_j^i}{\sum_{j=1}^{d}\sum_{i=1}^{\bar{l}} \hat{y}_j^i + \sum_{j=1}^{d}\sum_{i=1}^{\bar{l}} y_j^i}. \tag{7}$$

In contrast to the exact match ratio, both micro-average and macro-average F-measures take partial matches into account.

2.3 Optimizing Measures

Optimizing an evaluation criterion means we identify predictions $\hat{\mathbf{y}}$, so that the performance is better. In this article, we optimize measures under the situation of using the binary method, for which the predicted label vector $\hat{\mathbf{y}}$ is controlled by the d decision functions. In this regard, a measure is a function of the d decision functions in (1):

$$m(f_1, f_2, \ldots, f_d).$$

Table 1: An example of how decision thresholds affect macro-average and micro-average F-measures. For each label, instances are ordered according to their decision values. Dashed lines indicate cutting points. (a) True label vectors (b) An example of predictions: the classifier predicts two instances as positive for both labels. (c) Another example of predictions: the classifier predicts more positive instances by lowering thresholds. The macro-average F-measures for (b) and (c) are 0.6 and 0.452, respectively. The micro-average F-measures are 0.6 and 0.462, respectively.

rank	label 1	label 2
1	1	0
2	1	1
3	0	0
4	0	1
5	0	0
6	1	0
7	0	0
8	0	0
9	0	1
10	0	0

(a) True labels

label 1	label 2
1	1
1	1
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

(b) An example of predictions

label 1	label 2
1	1
1	1
1	1
1	0
0	0
0	0
0	0
0	0
0	0
0	0

(c) An example of predictions

Note that functions $f_1, \ldots, f_d$ predict labels $\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^{\bar{l}}$. Together with true label vectors $\mathbf{y}^1, \ldots, \mathbf{y}^{\bar{l}}$, one can calculate the measure value. Since testing instances' true label vectors are not available, we split available data to training/validation subsets, and improve the performance on the validation set. Therefore, one optimizes a measure m by

$$\max_{f_1, f_2, \ldots, f_d} \quad m\left(f_1, f_2, \ldots, f_d\right). \tag{9}$$

Unfortunately, (9) is a difficult global optimization problem. A more reasonable goal is to improve upon the binary method. Starting from decision functions of the binary method, we sequentially adjust one f_i and fix the others. The algorithm is described in Procedure 1.

In practice, maximizing the measure with respect to only one function may still be difficult, so we can just obtain any f_k such that

$$\begin{aligned} & m\left(f_1^1, \ldots, f_{k-1}^1, f_k, f_{k+1}^0, \ldots, f_d^0\right) \\ \geq \; & m\left(f_1^1, \ldots, f_{k-1}^1, f_k^0, f_{k+1}^0, \ldots, f_d^0\right). \end{aligned} \tag{10}$$

There are several ways to modify the decision function f_k. Here, we consider adjusting the threshold T_k in the decision function (3). Table 1 illustrates how different thresholds lead to different F-measures.

If the target measure is the macro-average F-measure, then the kth sub-problem (8)

Procedure 2 Cyclic optimization for maximizing measures

1. Initialize $f_1^0, f_2^0, \ldots, f_d^0$ by running the binary method.
2. For $t = 1, 2, \ldots,$
   (a) For $k = 1, \ldots, d$

$$\max_{f_k} \quad m\left(f_1^t, \ldots, f_{k-1}^t, f_k, f_{k+1}^{t-1}, \ldots, f_d^{t-1}\right). \tag{12}$$

   (b) Set f_k^t to the solution of (12).

is

$$\begin{aligned} \max_{f_k} \quad & m\left(f_1^1, \ldots, f_{k-1}^1, f_k, f_{k+1}^0, \ldots, f_d^0\right) \\ & = \frac{1}{d}\sum_{j=1}^{d} \frac{2\sum_{i=1}^{\bar{l}} \hat{y}_j^i y_j^i}{\sum_{i=1}^{\bar{l}} \hat{y}_j^i + \sum_{i=1}^{\bar{l}} y_j^i} \\ & = \frac{1}{d}\left(\frac{2\sum_{i=1}^{\bar{l}} \hat{y}_k^i y_k^i}{\sum_{i=1}^{\bar{l}} \hat{y}_k^i + \sum_{i=1}^{\bar{l}} y_k^i} + \text{terms not related to } f_k\right). \end{aligned} \tag{11}$$

Note that f_k affects only $\hat{y}_k^1, \ldots, \hat{y}_k^{\bar{l}}$. The above problem is then equivalent to

$$\max_{f_k} \quad \frac{2\sum_{i=1}^{\bar{l}} \hat{y}_k^i y_k^i}{\sum_{i=1}^{\bar{l}} \hat{y}_k^i + \sum_{i=1}^{\bar{l}} y_k^i},$$

which is independent of other decision functions. That is, because (11) is the summation of d separable terms, optimizing the macro-average F-measure is equivalent to optimizing per-label F-measure.

For the micro-average F-measure, the situation is more complicated. The kth subproblem is

$$\begin{aligned} \max_{f_k} \quad & m\left(f_1^1, \ldots, f_{k-1}^1, f_k, f_{k+1}^0, \ldots, f_d^0\right) \\ & = \frac{2\sum_{j=1}^{d}\sum_{i=1}^{\bar{l}} \hat{y}_j^i y_j^i}{\sum_{j=1}^{d}\sum_{i=1}^{\bar{l}} \hat{y}_j^i + \sum_{j=1}^{d}\sum_{i=1}^{\bar{l}} y_j^i} \\ & = \frac{2\sum_{i=1}^{\bar{l}} \hat{y}_k^i y_k^i + \text{terms not related to } f_k}{\sum_{i=1}^{\bar{l}} \hat{y}_k^i + \text{terms not related to } f_k}. \end{aligned} \tag{13}$$

Clearly, (13) depends on decision functions other than f_k. Under different functions $f_1^1, \ldots, f_{k-1}^1, f_{k+1}^0, \ldots, f_d^0$, (13) gives different optimal solution f_k. The order of labels thus affects the result of Procedure 1. Moreover, we can run Procedure 1 several times to improve the function value. We call this as a "cyclic optimization" procedure, which is described in Procedure 2.

We stop Procedure 2 if function values of two consecutive iterations are similar. For experiments in this article, we use

$$\frac{m^t - m^{t-1}}{m^0} < 0.001, \tag{14}$$

where m^t is the function value after the tth iteration. For the macro-average F-measure, whose function form is separable, Procedure 2 does not change the measure after the first iteration. Thus, there is no need to consider the cyclic optimization. For the micro-average F-measure and the exact match ratio, the measure function is not separable, so Procedure 2 can be used. We will study if the "separable form" of the macro-average F-measure makes its optimization easier than others.

3 Optimizing Measures via Threshold Selection

Following the discussion in Section 2.3, we adjust thresholds in the decision functions of the binary method. An intuitive and direct way is to select the value that achieves the best evaluation measure on a validation set. This approach is referred to as the SCut method in (Yang, 2001), and have been used in, for example, Lewis et al. (1996); Yang (1999). Algorithm 1 shows details by using five-fold cross validation (CV). The evaluation measure, specified by users, can be any criterion such as the macro-average F-measure. Given $\hat{l}$ validation instances sorted by their decision values, there are $\hat{l}+1$ possible predictions. Sequentially we check the performance using the medium of two adjacent decision values as the threshold. For the two extreme situations, where SVM predicts all instances as positive or negative, we check a threshold slightly higher (or lower) than the highest (or lowest) decision value.

To see how this algorithm works, we create an artificial problem with $d = 2$ labels and 10 validation instances. The goal is to optimize the macro-average F-measure by adjusting T_1, T_2 of the two decision functions f_1, f_2. Assume we have the following results from the binary method:

	label 1		label 2		
	$\mathbf{y}_1$	dec. value	$\mathbf{y}_2$	dec. value	
	1	1.2	0	1.4	
	1	1.1	0	1.2	
Original $T_1 = 0$ ⟶	0	-1.1	0	-1.1	⟵ Original $T_2 = 0$
	1	-1.2	0	-1.2	
	0	-1.3	0	-1.4	
	1	-1.4	0	-1.5	
New $T_1 = -1.45$ ⟶	0	-1.5	0	-1.6	
	0	-1.6	1	-1.7	⟵ New $T_2 = -1.75$
	0	-1.7	0	-1.8	
	0	-1.8	0	-2.1	

We order validation instances according to their decision values. The columns $\mathbf{y}_1$ and $\mathbf{y}_2$ indicate true labels. With the original threshold $T_1 = T_2 = 0$, we predict two positive instances for both labels. The macro-average F-measure is

$$\frac{1}{2}\left(\frac{2\times 2}{2+4} + \frac{2\times 0}{2+1}\right) = 0.333.$$

Algorithm 1 Simple threshold tuning for the binary method with SVMs

1. For label $j = 1, \ldots, d$,
   (a) Split data into five folds. For fold $i = 1, \ldots, 5$,
      i. Train SVM^i on four folds of data.
      ii. The decision threshold T_j^i of SVM^i is the value that achieves the best validation result. This operation leads to a new decision function satisfying (10).
   (b) Train the jth SVM as in the binary method. The decision threshold T_j of the jth SVM is set to the average of the five threshold values obtained in the cross validation step:
$$T_j = \frac{1}{5}\sum_{i=1}^{5} T_j^i.$$

Using threshold $T_1 = -1.45$ and $T_2 = -1.75$, we get better macro-average F-measure

$$\frac{1}{2}\left(\frac{2\times 4}{6+4} + \frac{2\times 1}{8+1}\right) = 0.511.$$

Note that optimizing the macro-average F-measure sometimes leads to a lower micro-average F-measure. For example, if we keep $T_2 = 0$, the micro-average F-measure is $8/13 = 0.615$, better than $10/19 = 0.526$ of setting $T_2 = -1.75$. In Algorithm 1, for label j, we obtain five thresholds $T_1^j, \ldots, T_5^j$ after the five-fold CV. Then we return the average as the final threshold. This setting is reasonable if distributions of training/validation splits are not very different.

Since Algorithm 1 fits the validation sets, according to (Yang, 2001) and our own experiments, overfitting may occur when the binary problem is unbalanced. That is, very few instances are associated with the label. Then the resulting threshold is either too hight or too low. Yang (2001) further argues that

- A too high threshold lowers macro-average F-measure.
- A too low threshold lowers both macro- and micro-average F-measures.

We explain why a too high threshold does not affect the micro-average F-measure much. Note that too high/low thresholds mainly occur for unbalanced binary problems. Hence, the vector $[y_j^1, \ldots, y_j^{\bar{l}}]^T$ for label j has few non-zero elements. A too high threshold hence gives small $\sum_{i=1}^{\bar{l}} \hat{y}_j^i y_i^j$ and $\sum_{i=1}^{\bar{l}} \hat{y}_j^i$ in (7), so the micro-average F-measure is not affected much. Yang (2001) then proposes the fbr heuristic: when the F-measure of a label does not achieve a pre-defined value called fbr, we set the threshold by one of the following two ways:

- SCutFBR.0: set the threshold to ∞. This setting does not affect the micro-average F-measure too much (as the label is a rare one), but hurts macro-average.
- SCutFBR.1: set the threshold to the highest decision value of the validation data. This method less hurts the macro-average.

Algorithm 2 Threshold tuning with the fbr heuristic (SCutFBR.1 in Yang, 2001)

1. Given an fbr value.
2. For label $j = 1, \ldots, d$,
   (a) Split data into five folds. For fold $i = 1, \ldots, 5$,
      i. Train SVM^i on four folds of data.
      ii. The decision threshold T_j^i of SVM^i is set to the value that achieves the best validation result. This operation leads to a new decision function satisfying (10).
      iii. If the validation result is lower than the given fbr value, set T_j^i to the highest decision value of the ith fold of data.
   (b) Train the jth SVM as in the binary method. The decision threshold T_j of the jth SVM is set to the average of five thresholds obtained in Step 2a:

$$T_j = \frac{1}{5}\sum_{i=1}^{5} T_j^i.$$

Given a pre-specified fbr value, Algorithm 2 gives the details of using SCutFBR.1. Lewis et al. (2004) then use Algorithm 2 as a building block and conduct five-fold CV to select the best fbr value from several candidates. The procedure (called SVM.1 in their paper) is described in Algorithm 3. Due to the two-level cross validation, its computational time is longer than Algorithm 2.

Since both SCutFBR.0 and SCutFBR.1 set the threshold to a higher value, when the obtained F-measure is not good enough (less than fbr), we make conservative predictions and avoid a too low threshold. As a too high threshold hurts more the macro-average F-measure, using fbr, one expects a better micro-average F-measure, but maybe a slightly worse macro-average. However, our experiments give quite different results. Table 2 shows a comparison between the binary method and Algorithms 1-2. We use 5,000 training and 5,000 testing points from the OHSUMED data[1]. We adjust thresholds to optimize the macro-average F-measure. Clearly, Algorithms 1-2 are much better than the vanilla binary method. Thus, adjusting thresholds is very essential. Interestingly, results of Algorithm 2 using fbr are opposite to what we expect: Algorithm 2 gives better macro-average F-measure than Algorithm 1. We have similar observations for many other data sets. Therefore, Yang (2001) does not address the following issues when proposing the fbr heuristic:

- Is it possible that this fbr heuristic causes a too high threshold and hence seriously hurts macro-average?
- Table 2 shows better macro-average F-measure. How does this result occur when using the fbr heuristic?

Through some analysis in Section 5, we show new and interesting properties of fbr.

[1]Details are discussed in Section 4.1.

Algorithm 3 The SVM.1 method in Lewis et al. (2004): an outer layer of CV selects the fbr value of SCutFBR.1.

1. Given a list of fbr values.
2. For label $j = 1, \ldots, d$,
   (a) For each fbr value, conduct five-fold cross validation. Algorithm 2 is used for each training/validation split. Select the fbr value which achieves the best evaluation result.
   (b) Run Algorithm 2 with the best fbr value obtained in the previous step to get the jth SVM model and the threshold T_j.

Note that there are two levels of CV in this procedure: the outer one in Step 2a determines the best fbr value and the inner one is in Algorithm 2. One may think that for n different fbr values, the algorithm needs to run the outer CV n times. In fact, the only difference among these n CVs is in the comparison between the fbr value and the F-measure from Algorithm 2. Hence, in practice, we combine these n CVs together to save time.

Table 2: A comparison between the binary method and Algorithms 1-2. We use the OHSUMED data set (described in Section 4.1). The best result is bold-faced.

	Macro-avg.	Micro-avg.
Binary	0.118	0.454
Algo. 1	0.232	0.414
Algo. 2	**0.246**	**0.505**

A MATLAB implementation of Algorithm 3 is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/multilabel/`.

4 Data and Experimental Settings

We discuss data and experimental settings before showing detailed results and analysis,

4.1 Real-World Data Sets

All multi-label data considered here are from real-world applications. The data scene, yeast, and RCV1-V2 are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets`. Data statistics are listed in Table 3. In the table, "#inst." is the number of instances in the data set, "#feat." is the number of features, "#label" is the number of labels, and "label size frequency" (in percentage) is the relative frequency of each label size in the data set. Label size means the number of labels of an instance. For example, in the Yahoo! En data set, 72.3% instances are associated with only one label, 21.1% instances have two labels, and the rest 6.6% instances have more than two labels. "Label distr. ranking" is the ranking of labels' number of occurrences. We divide the

Table 3: Data statistics. See Section 4.1 for the explanation of each column. Except Yahoo!, all other collections have training/testing splits, but here we show only statistics of the training part. We further select subsets for our experiments (details in Table 4). The six Yahoo! sets are: Arts & Humanities (Ar), Business & Economy (Bu), Computers & Internet (Co), Education (Ed), Entertainment (En), and Health (He).

Data set	#inst.	#feat.	#label	Label size frequency (%)				
				1	2	3	4	≥5
scene	1,211	294	6	93.9	6.0	0.1	0.0	0.0
yeast	1,500	103	14	1.6	16.9	4.5	45.6	31.3
OHSUMED	40,000	36,522	94	17.3	45.9	16.3	10.3	10.2
Yahoo! Ar	7,484	23,146	26	55.6	30.5	9.7	2.8	1.4
Yahoo! Bu	11,214	21,924	30	57.6	28.8	11.1	1.7	0.8
Yahoo! Co	12,444	34,096	33	69.8	18.2	7.8	3.0	1.1
Yahoo! Ed	12,030	27,534	33	66.9	23.4	7.3	1.9	0.6
Yahoo! En	12,730	32,001	21	72.3	21.1	4.5	1.0	1.1
Yahoo! He	9,205	30,605	32	53.2	34.0	9.5	2.4	0.9
RCV1-V2	23,149	47,236	101	12.3	29.5	35.7	10.8	11.7

Data set	Label distr. ranking (%)				
	1st	2nd	3rd	4th	5th
scene	22.9	18.7	18.5	16.3	16.2
yeast	75.2	74.4	43.0	39.9	35.5
OHSUMED	37.5	20.1	17.7	16.8	14.9
Yahoo! Ar	24.6	22.4	16.0	15.8	11.8
Yahoo! Bu	86.7	28.3	8.7	5.2	4.9
Yahoo! Co	52.7	20.5	15.6	7.7	6.2
Yahoo! Ed	31.1	19.1	15.9	13.1	13.0
Yahoo! En	29.0	19.1	18.4	15.7	11.4
Yahoo! He	51.1	25.3	17.2	14.9	13.4
RCV1-V2	46.6	30.1	18.1	14.9	11.0

number of instances of each label by the total number of instances. For example, the most popular label in the Yahoo! Ar data set is associated with 24.6% of instances, and the second is with 22.4%. For each data set, we present ratios of the top five labels.

4.1.1 scene

The scene data set is a semantic scene classification problem from Boutell et al. (2004). Each of the 2,407 images is associated with some of the six different semantic scenes (beach, sunset, fall foliage, field, urban, and mountain). We use the same setting in Boutell et al. (2004), which has 1,211 training and 1,196 testing images. Each instance is an image of 294 features generated from the CIE L*U*V-like color space. We do not scale this data set in order to preserve the raw color information.

4.1.2 yeast

This is a microarray data set from Munich Information Center for Protein Sequences (MIPS) (Mewes et al., 1997). We experiment with the same data set as in Elisseeff and Weston (2002), which has 1,500 training and 917 testing instances. Each instance in the data set represents a yeast gene, and each gene has 103 features. There are 14 labels indicating gene functional groups. As each yeast gene is assigned to a set of labels, we have a multi-label data set. We linearly scale each feature to the range $[-1, +1]$.

4.1.3 OHSUMED

This data set (Hersh et al., 1994) is a collection of medical references from MEDLINE, an on-line bibliographic database for medical literature. Our experiment here is based on the specific version of Cesa-Bianchi et al. (2005), which includes 94 labels and 55,503 documents. The corpus is randomly split to 40,000 documents for training and 15,503 documents for testing. This procedure is repeated five times, so we have five pairs of training and testing instances.

We use the feature vectors with the TF-IDF (term frequency, inverse document frequency) encoding. For a term t in a document $\mathbf{\Psi}$, we get the weight

$$w_{\mathbf{\Psi}}(t) = \mathrm{tf}(t, \mathbf{\Psi}) \times \log_2 \frac{|D|}{\mathrm{df}(t)}, \tag{15}$$

where $\mathrm{tf}(t, \mathbf{\Psi})$ is the number of term t in document $\mathbf{\Psi}$, $\mathrm{df}(t)$ is the number of documents that contain term t, and $|D|$ is the number of documents. In addition, we use cosine normalization so that each vector has unit length.

4.1.4 Yahoo! Directories

This collection, used in Ueda and Saito (2003); Kazawa et al. (2005), consists of web pages from 14 top-level categories of Yahoo! directories. We treat web pages belonging to the same top-level category as a data set, so there are 14 data sets. Each top-level category includes several second-level categories, which are our labels. As a page may simultaneously belong to several second-level categories, the data set is multi-labeled. We choose six data sets: Arts & Humanities (Ar), Business & Economy (Bu), Computers & Internet (Co), Education (Ed), Entertainment (En), and Health (He). All instances are encoded with TF-IDF and are cosine normalized.

4.1.5 RCV1-V2

This data set (Lewis et al., 2004) contains newswire stories from Reuters Ltd. Lewis et al. (2004) used several schemes to process the documents, including removing stopping words, stemming, and transforming the documents to vectors with TF-IDF format, etc. Each instance is cosine normalized. There are three category sets: *Topics*, *Industries*, and *Regions*. In the article, we consider the *Topics* category set. There are 23,149 training and 781,265 testing instances. For labels, 101 appear in the training set and all 103 appear in the testing set.

Table 4: Size of training/testing subsets used in our experiments, and the SVM parameters for the binary method and Algorithms 1-3. The six Yahoo! data sets happen to share the same parameter.

Data set	# training	# testing	Parameters (C, γ)
scene	1,211	1,196	(4, 0.25)
yeast	1,500	917	(2, 2)
OHSUMED	5,000	5,000	(2, -)
Yahoo!	3,000	3,000	(2, -)
RCV1-V2	3,000	3,000	(2, -)

4.2 Experimental Settings

The three document data (Yahoo! directories, OHSUMED, and RCV1-V2) are large, so we randomly select several pairs of training and testing subsets. We use stratified selections so that all labels appear in the training subset. For the six Yahoo! sets, we select 3,000 training and 3,000 testing documents, and repeat such a selection five times. In addition, we generate another pair of training and testing subsets for SVM parameter tuning. For the OHSUMED data, its original form contains five pairs of training (40,000 instances) and testing (15,503 instances) sets. We repeatedly select 5,000 instances from the training set and 5,000 from the testing set to form five pairs. The same scheme is used for RCV1-V2 to generate five training/testing pairr, but each training (testing) set contains 3,000 instances. Table 4 summarizes the size of training and testing data used in our experiments.

For each problem, we must choose a suitable SVM kernel. As the number of features of yeast and scene is small, we use the RBF kernel, which maps data vectors to a higher dimensional space. For long feature vectors of documents, existing work indicates that it is suitable to stay in the original input space. Hence, we use the linear kernel. If the RBF kernel is used, there are two parameters: the penalty parameter C and the kernel parameter γ. We choose them by maximizing five-fold CV exact match ratios on a grid space of parameters: C in $[2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3]$ and γ in $[2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3]$. When the linear kernel is used, the only parameter is C. For Yahoo! data, we use the sixth pair of training and testing subsets to select it from three values: 1, 2, and 4. For OHSUMED and RCV1-V2, where each of them has five pairs of training and testing subsets, we use the first pair to choose C. Final parameters are listed in Table 4, and they are used in all Algorithms 1-3.

The fbr value in Algorithm 2 is set to 0.1. For Algorithm 3, we follow Lewis et al. (2004) to check eight fbr values: 0.1 to 0.8. For data sets that have several pairs of training and testing sets, we report the average result. All implementations are based on the LIBSVM (Chang and Lin, 2001) MATLAB interface.

5 Experiments and Analysis

We present and analyze results of optimizing three measures: macro-average F-measure, micro-average F-measure, and exact match ratio. We also have a subsection discussing

Table 5: Optimizing the macro-average F-measure by the binary method and Algorithms 1-3. The best result is bold-faced.

	macro-average F-measure				micro-average F-measure			
Data set	Binary	Algo. 1	Algo. 2	Algo. 3	Binary	Algo. 1	Algo. 2	Algo. 3
scene	0.653	**0.765**	**0.765**	**0.765**	0.652	**0.755**	**0.755**	**0.755**
yeast	0.405	**0.505**	0.504	0.502	**0.651**	0.640	0.646	0.644
OHSUMED	0.117	0.227	**0.247**	**0.247**	0.452	0.446	0.497	**0.499**
Yahoo! Ar	0.304	0.408	**0.410**	0.408	0.484	0.530	**0.533**	0.528
Yahoo! Bu	0.281	**0.391**	0.385	0.380	0.774	0.764	**0.774**	**0.774**
Yahoo! Co	0.292	0.382	**0.391**	0.380	0.588	0.591	0.611	**0.612**
Yahoo! Ed	0.255	0.315	**0.338**	0.334	0.507	0.524	**0.547**	**0.547**
Yahoo! En	0.410	**0.497**	0.494	0.490	0.624	**0.659**	0.658	0.656
Yahoo! He	0.337	0.389	**0.416**	0.411	0.697	0.709	**0.716**	0.715
RCV1-V2	0.341	0.484	**0.506**	0.495	0.741	0.762	0.762	**0.764**

new properties of the fbr heuristic.

5.1 Optimizing the Macro-average F-measure

Table 5 shows the results of optimizing the macro-average F-measure. Clearly, all Algorithms 1-3 give much better results than the binary method. Hence, in doing multi-label classification, one should always adjust thresholds.

For half of the data sets, the macro-average F-measure of Algorithm 1 is worse than that of Algorithms 2-3, which use fbr. Moreover, Algorithm 2 (fixed fbr) performs similar to Algorithm 3 (CV to select fbr). Overall, fbr is a useful heuristic. We indicated in Section 3 that Yang (2001) proposes the fbr heuristic to have more conservative predictions. Thus one expects better micro-average at the expense of slightly lower macro-average F-measure. However, our results are very different: We have much better macro-average and equally good micro-average F-measure. In Section 5.2, we explain why such results occur.

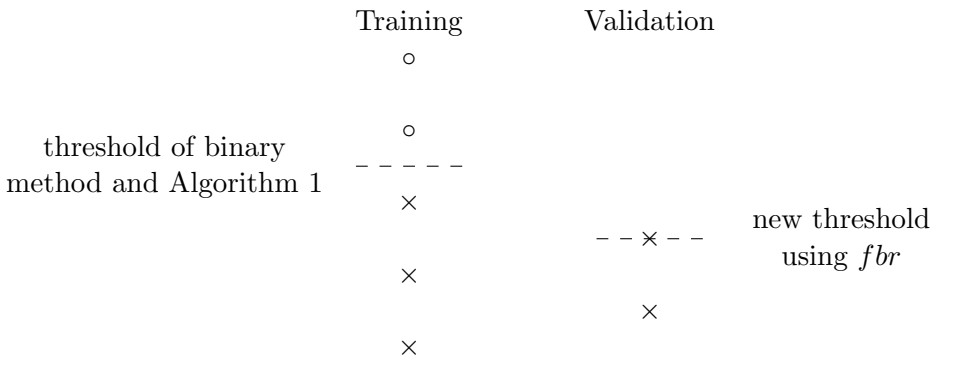
5.2 Analysis of the fbr Heuristic

By analyzing results of Table 5, we have some interesting findings:

1. **When the number of positive data is very small (e.g., ≤ 5), thresholds are often decreased, and macro-average F-measure is improved.**

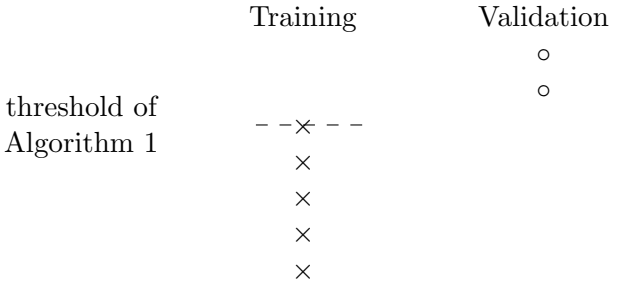
   Table 6 shows details of running the fifth subset of the OHSUMED data. We investigate the four labels which contribute the most to the increase of the macro-average F-measure. Their number of instances in the training set, the contingency tables, and the thresholds selected by Algorithms 1 and 2 are presented. Clearly, the number of positive data in a binary problem is very small (≤ 5). Algorithm 2 decreases the threshold, and dramatically improves the F-measure.

We explain why fbr leads to the decrease of the threshold. As the label is so rare, the validation fold may not contain any positive instance. We end up with the following scenario of decision values:



In this example, there are only two positive instances, and all go to the training folds. To separate positive and negative training instances, very possibly, the resulting SVM threshold is larger than the highest decision value of validation data, which contain only negative points. Algorithm 1 does not further adjust the threshold as for any value, the validation F-measure is zero. But for Algorithm 2, as the zero F-measure is less than fbr, the threshold is lowered.

When the validation fold contains positive data, very possibly we have



The best threshold from Algorithm 1 achieves an F-measure equal to one, which is higher than fbr. Thus, Algorithm 2 does not further adjust the threshold. After averaging the thresholds from different CV folds, the final threshold of Algorithm 2 is smaller than that of Algorithm 1.

An immediate concern is whether such a change causes too low thresholds. We argue that in general this problem does not happen. Now these labels are rare, and the original number of TP (True Positive) by Algorithm 1 is generally zero. Hence lowering the threshold does not quickly increase the number of FP (False Positive). Hence, micro-average F-measure is not affected much. For macro-average, since the original F-measure is close to zero, by predicting a few more points as positive, the chance of getting a better F-measure is high. Note that macro-average F-measure is the average of label F-measures. If there are many extremely rare labels, after using Algorithm 1, quite a few have F-measures zero. Improvements on some of them significantly boost the average value. Thus, in Table 5, Algorithm 2 gives a much better macro-average F-measure than Algorithm 1.

Table 6: Examples where thresholds are decreased after using fbr. Algorithm 2 then predicts more data as positive.

		Algorithm 1					Algorithm 2				
label	# training	TP	FP	FN	F	threshold	TP	FP	FN	F	threshold
16	2	0	0	2	0.00	0.70	1	2	1	0.40	0.28
17	3	0	0	2	0.00	0.63	1	10	1	0.15	0.12
32	1	0	0	1	0.00	0.84	1	8	0	0.20	0.11
35	5	0	0	2	0.00	0.68	1	0	1	0.67	0.32

Table 7: Examples where thresholds are increased after using fbr. Algorithm 2 then predicts less data as positive.

		Algorithm 1					Algorithm 2				
label	# training	TP	FP	FN	F	threshold	TP	FP	FN	F	threshold
65	28	20	3097	7	0.01	-0.07	0	0	27	0.00	0.61
85	19	22	1318	11	0.03	0.02	0	5	33	0.00	0.45
53	17	4	750	13	0.01	0.06	0	4	17	0.00	0.48
52	7	9	1084	3	0.02	0.02	0	3	12	0.00	0.34

This observation is very interesting as fbr serves a different role from what it is originally designed for. Next we show that in some other situations fbr does increase the threshold.

2. **When the number of positive data is small but not too small, fbr helps to prevent a huge number of FP by increasing the threshold.**

   For the same data (fifth subset of OHSUMED), Table 7 shows that for some labels, the fbr heuristic significantly reduces the number of FP (False Positive). Take the label 65 as an example. It has 28 of the total 5,000 instances. With Algorithm 1, the best F-measure of predicting five validations folds is 0.0127. There is no single fold of data that can achieve the specified fbr value 0.1. Therefore, the fbr heuristic sets the threshold to the highest decision value. In testing, Algorithm 1 predicts 3,104 instances as positive, but there are only 27 positive instances in the testing set. With the fbr setting, both Algorithms 2 and 3 predict no instances as positive. Therefore, fbr minimizes the impact on micro-average F-measure at the cost of slightly lowering the macro-average F-measure.

   As micro-average F-measure considers all labels together, in Table 5, the improvement is not significant. Nevertheless, we see that fbr serves as a safeguard to prevent a catastrophic situation of huge FP.

If we order labels according to their numbers of data, roughly they are separated to three segments. For the first segment, with enough data, the resulting F-measure is larger than fbr, so the threshold remains the same. For the second part, Algorithm 2 increases thresholds for more conservative predictions. For the third part, labels rarely

Table 8: Optimizing the micro-average F-measure by the binary method and Algorithms 1-2.

	macro-average F-measure			micro-average F-measure		
Data set	Binary	Algo. 1	Algo. 2	Binary	Algo. 1	Algo. 2
scene	0.653	0.766	**0.767**	0.652	0.759	**0.760**
yeast	0.405	**0.465**	0.463	0.651	0.681	**0.682**
OHSUMED	0.117	0.192	**0.230**	0.452	**0.519**	0.515
Yahoo! Ar	0.304	0.387	**0.398**	0.484	**0.538**	0.537
Yahoo! Bu	0.281	0.356	**0.370**	0.774	**0.784**	0.781
Yahoo! Co	0.292	0.367	**0.378**	0.588	**0.624**	0.622
Yahoo! Ed	0.255	0.300	**0.335**	0.507	**0.552**	0.550
Yahoo! En	0.410	0.472	**0.496**	0.624	**0.660**	0.658
Yahoo! He	0.337	0.378	**0.410**	0.697	**0.722**	0.718
RCV1-V2	0.341	0.443	**0.490**	0.741	**0.775**	0.773

occur and thresholds are decreased to have more positive predictions. Occasionally, two segments significantly overlap, so the performance is not better. Nevertheless, the fbr heuristic is generally useful, and the above analysis gives the rationale behind this method.

5.3 Optimizing the Micro-average F-measure

With the same settings as in Section 5.1, we maximize the micro-average F-measure. Since Algorithm 2 performs similar to Algorithm 3 in earlier experiments, for subsequent comparisons we exclude Algorithm 3 . As discussed in Section 2, the order of the labels may affect the result of optimizing the micro-average F-measure. Thus, we randomly permute labels and report the average from ten runs. For data with several training and testing pairs, we run each pair several times, and keep the total number of runs to ten. Table 8 shows the results.

Clearly, Algorithms 1 and 2 obtain better micro-average F-measure than the binary method. Therefore, adjusting thresholds effectively optimizes the target measure. The fbr heuristic (Algorithm 2) significantly improves the macro-average F-measure, but gives only similar micro-average F-measure. This situation is the same as that in Table 5.

Compared to Table 5, we get better micro-average F-measure in Table 8. This result is expected since the goal of Table 8 is to optimize the micro-average F-measure. However, the macro-average F-measure is generally lower than that in Table 5. The following table shows the details of the fifth run of the yeast set by using Algorithm 2 to optimize macro-average and micro-average F-measures[2].

Measure being optimized	Sum of TP	Sum of FP	Sum of FN
macro-average F-measure	3,026	2,446	873
micro-average F-measure	2,783	1,485	1,116

[2] The micro-average F-measure of the fifth run is the closest to the average of the ten runs.

Table 9: Optimizing the micro-average F-measure: the improvement of cyclic over non-cyclic optimization.

	macro-average F-measure		micro-average F-measure	
Data set	Algo. 1 (%)	Algo. 2 (%)	Algo. 1 (%)	Algo. 2 (%)
scene	**0.15**	**0.08**	**0.42**	**0.32**
yeast	-0.83	-0.69	**0.04**	**0.08**
OHSUMED	-0.95	-1.07	-0.01	**0.08**
Yahoo! Ar	-0.13	**0.04**	**0.15**	**0.22**
Yahoo! Bu	-0.09	-0.09	-0.04	-0.04
Yahoo! Co	-0.60	-0.72	**0.08**	-0.00
Yahoo! Ed	-0.23	-0.08	**0.07**	-0.01
Yahoo! En	-0.56	-0.09	**0.14**	**0.26**
Yahoo! He	-0.02	-0.06	-0.05	-0.04
RCV1-V2	-0.40	-0.41	0.00	-0.01

Tuning the micro-average F-measure predicts much less positive instances; i.e., the thresholds are generally higher, and we successfully increase the micro-average F-measure by reducing the number of false positives. The same situation occurs for most other data sets. Therefore, the better micro-average F-measure is at the expense of a lower macro-average F-measure. One must decide the target measure according to his/her needs.

We investigate how the cyclic optimization affects the optimization on the micro-average F-measure. Table 9 summarizes the improvements over the non-cyclic optimization results. The cyclic optimization algorithm usually takes three or four iterations before reaching the stopping condition (14). The results are only similar to or slightly better than those in Table 8. In practice, one may just optimize each label once for shorter training time.

5.4 Optimizing the Exact Match Ratio

We investigate how Algorithms 1 and 2 perform when the goal is to optimize the exact match ratio. We use the same settings as those in the previous two experiments, and present results in Table 10.

Observations are similar to those of optimizing other measures. Adjusting thresholds (Algorithms 1-2) significantly improves all three measures over the binary method. As for the fbr heuristic, it mainly helps the macro-average F-measure, but not the other two. Note that the present target for optimization is the exact match ratio. Therefore, macro-average (micro-average) F-measure is lower than that in Table 5 (Table 8).

By adjusting thresholds for the scene data, Algorithms 1 and 2 significantly improves the result of the binary method by 24.8% and 21%, respectively. We give a detailed investigation. We examine a run whose exact match ratio by Algorithm 1 is 0.608. The testing set has 1,196 instances but only 103 of them are multi-labeled. The number of correctly predicted instances is listed below:

Table 10: Optimizing the exact match ratio by the binary method and Algorithms 1-2.

	Macro-avg. F-measure			Micro-avg. F-measure			Exact match ratio		
Data set	Binary	Algo. 1	Algo. 2	Binary	Algo. 1	Algo. 2	Binary	Algo. 1	Algo. 2
scene	0.653	**0.761**	0.755	0.652	**0.757**	0.750	0.490	**0.611**	0.593
yeast	0.405	**0.409**	0.401	0.651	**0.657**	0.656	0.206	**0.217**	0.215
OHSUMED	0.117	0.134	**0.195**	0.452	**0.472**	0.469	0.160	**0.177**	0.174
Yahoo! Ar	0.304	0.341	**0.358**	0.484	**0.517**	**0.517**	0.320	**0.344**	0.343
Yahoo! Bu	0.281	0.295	**0.329**	0.774	**0.776**	0.774	0.617	**0.620**	0.616
Yahoo! Co	0.292	0.322	**0.337**	0.588	**0.613**	0.611	0.443	**0.464**	0.461
Yahoo! Ed	0.255	0.284	**0.325**	0.507	**0.540**	0.538	0.335	**0.363**	0.360
Yahoo! En	0.410	0.443	**0.476**	0.624	**0.652**	**0.652**	0.464	**0.502**	0.501
Yahoo! He	0.337	0.354	**0.394**	0.697	**0.711**	0.707	0.517	**0.526**	0.517
RCV1-V2	0.341	0.373	**0.432**	0.741	**0.751**	0.749	0.417	**0.437**	0.431

# of labels of an instance	1	2
Binary method	579	7
Algorithm 1	703	24

Instances associated with only one label are much easier to predict then those with multiple labels. The improvement on those one-label instances is responsible for 88% performance increase. For other data sets which are more multi-labeled, the improvement on the exact match ratio is much smaller ($< 10\%$).

Next, we investigate how the cyclic optimization performs when the target criterion is the exact match ratio. The improvements over non-cyclic optimization are in Table 11. Unlike the case of optimizing the micro-average F-measure, where cyclic optimization does not help, here the exact match ratio is slightly improved for most data sets.

5.5 Additional Experiments

For the same measure, we compare the results when it is and is not the target optimization criterion. For example, how good is the macro-average F-measure when we optimize the micro-average F-measure or the exact match ratio? Table 12 shows the comparison results. Reading each row of the table reveals that micro-average F-measure is less sensitive to the optimization of other measures. This result may be due to that micro-average F-measure considers all labels together and is less affected by some rarely occurred labels. Moreover, since the exact match ratio does not take partial matches into account, a slight change of thresholds easily results in very different ratios.

We close this section by summarizing in Table 13 the improvements over the binary method. The cyclic optimization is not considered. Clearly, we are very successful on optimizing the macro-average F-measure.

Table 11: Optimizing the exact match ratio: the improvement of cyclic over non-cyclic optimization.

Data set	Macro-average F		Micro-average F		Exact match ratio	
	Algo. 1 (%)	Algo. 2 (%)	Algo. 1 (%)	Algo. 2 (%)	Algo. 1 (%)	Algo. 2 (%)
scene	**0.01**	**0.99**	-0.01	**1.15**	**0.18**	**3.53**
yeast	-0.03	**0.06**	**0.08**	**0.12**	**0.05**	**1.57**
OHSUMED	**1.04**	**0.21**	**0.59**	**0.46**	**0.78**	**0.89**
Yahoo! Ar	**1.57**	**0.58**	**0.79**	**0.59**	**0.95**	**1.10**
Yahoo! Bu	-0.61	**0.38**	-0.04	-0.03	**0.02**	-0.05
Yahoo! Co	**0.55**	-0.19	**0.20**	-0.01	**0.05**	**0.16**
Yahoo! Ed	**0.33**	-0.24	-0.27	-0.39	**0.43**	**0.71**
Yahoo! En	**0.29**	-0.07	**0.26**	**0.05**	**0.32**	-0.21
Yahoo! He	**0.29**	**0.13**	**0.05**	**0.06**	**0.20**	**0.08**
RCV1-V2	**0.07**	**0.41**	-0.01	**0.11**	**0.34**	**0.39**

6 Conclusions

Adjusting the decision thresholds in the binary method is a simple and effective way to improve the performance of multi-label classification. As the binary method treats labels separately, optimizing measures with the same property is easier: The improvements of the macro-average F-measure, which can be decomposed into independent tasks of measuring each label, is bigger than those of the other two measures (micro-average F-measure and the exact match ratio).

The fbr heuristic improves the performance when some labels rarely occur. In such a situation, tuning thresholds overfits validation instances. We show that this heuristic nicely increases/decreases the threshold according to two different situations. It better helps the macro-average F-measure than the others.

Though intuitively the cyclic optimization should improve the performance, our results show that it marginally helps. Hence, optimizing each label once is enough in practice.

References

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: a case study in handwriting digit recognition. In *International Conference on Pattern Recognition*, pages 77–87. IEEE Computer Society Press, 1994.

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

N. Cesa-Bianchi, C. Gentile, A. Tironi, and L. Zaniboni. Incremental algorithms for hierarchical classification. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

Table 12: For each measure (corresponding to a row in the table), we compare the value when it is optimized with the value when another measure is optimized. For example, if using Algorithm 1 to optimize the micro-average F-measure, the resulting macro-average F-measure is 6.2% lower than the value of optimizing the macro-average F-measure.

	Algorithm 1 optimizes		
	macro-average F-measure (%)	micro-average F-measure (%)	exact match ratio (%)
macro-average F-measure		-6.2	-16.9
micro-average F-measure	-3.8		-2.8
exact match ratio	-18.9	-6.4	

	Algorithm 2 optimizes		
	macro-average F-measure (%)	micro-average F-measure (%)	exact match ratio (%)
macro-average F-measure		-2.9	-11.1
micro-average F-measure	-1.5		-2.8
exact match ratio	-12.4	-6.0	

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

A. Clare and R. King. Knowledge discovery in multi-label phenotype data. In *Proceedings of ECML/PKDD*, 2001.

F. D. Comité, R. Gilleron, and M. Tommasi. Learning multi-label alternating decision trees from texts and data. In *MLDM*, pages 35–49, 2003.

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*, 2002.

N. Ghamrawi and A. McCallum. Collective multi-label classification. In *CIKM*, 2005.

W. Hersh, C. Buckley, T. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of SIGIR-94*, pages 192–201, 1994.

T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE. URL `citeseer.ist.psu.edu/joachims97text.html`.

A. Karalič, , and V. Pirnat. Significance level based multiple tree classification. *Informatica*, 15(5), 1991.

Table 13: The improvements over the binary method by optimizing different evaluation criterion.

Data set	macro-avg. F-measure		micro-avg. F-measure		exact match ratio	
	Algo. 1 (%)	Algo. 2 (%)	Algo. 1 (%)	Algo. 2 (%)	Algo. 1 (%)	Algo. 2 (%)
scene	17.1	17.1	16.4	16.6	24.8	21.0
yeast	24.6	24.3	4.6	4.6	5.3	4.5
OHSUMED	94.5	111.4	14.7	13.8	11.0	8.1
Yahoo! Ar	34.3	35.1	11.2	10.9	7.5	7.1
Yahoo! Bu	38.9	36.8	1.2	0.9	0.5	-0.2
Yahoo! Co	30.6	33.6	6.2	5.9	4.8	3.9
Yahoo! Ed	23.8	32.9	8.8	8.5	8.3	7.4
Yahoo! En	21.2	20.3	5.6	5.4	8.3	8.0
Yahoo! He	15.6	23.5	3.5	3.0	1.9	0.0
RCV1-V2	41.7	48.1	4.7	4.4	4.9	3.6

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.

D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–306, 1996.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

A. McCallum. Multi-label text classification with a mixture model trained by EM. In *Proceedings of the AAAI'99 Workshop on Text Learning*, 1999.

H.-W. Mewes, K. Albermann, K. Heumann, S. Lieb, and F. Pfeiffer. MIPS: a database for protein sequences, homology data and yeast genome information. *Nucleic Acids Research*, 25(1):28–30, 1997.

K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000. URL `citeseer.ist.psu.edu/article/nigam99text.html`.

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

J. M. Tague. Information retrieval experiment. In K. S. Jones, editor, *The pragmatics of information retrieval experimentation*, chapter 5, pages 59–102. Butterworths, London, 1981.

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

Y. Yang. A study on thresholding strategies for text categorization. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, pages 137–145, New Orleans, US, 2001. ACM Press, New York, US.

Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2):69–90, 1999.

M.-L. Zhang and Z.-H. Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31, 2001.