

Transductive Multi-Label Learning via Alpha Matting

Xiang-Nan Kong, Michael K. Ng, and Zhi-Hua Zhou, *Senior Member, IEEE*

X.-N. Kong and Z.-H. Zhou are with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China (e-mail: {kongxn, zhoush}@lamda.nju.edu.cn).

M. Ng is with the Department of Mathematics, Hong Kong Baptist University, Hong Kong, China (e-mail: mng@math.hkbu.edu.hk).

August 12, 2009

DRAFT

Abstract

Multi-label learning deals with the problems when each instance can be assigned to multiple classes simultaneously, which are ubiquitous in real-world learning tasks. In this paper, we propose a new multi-label learning method, which is able to exploit unlabeled data to obtain an effective model for assigning appropriate multiple labels to instances. The proposed method is called TRAM (TRansductive multi-label learning via Alpha Matting), which formulates transductive multi-label learning as an optimization problem. We develop an efficient algorithm which has a closed form solution to solve this optimization problem. Empirical studies on real-world multi-label learning tasks show that TRAM can effectively make use of unlabeled data information to achieve performance as good as existing state-of-the-art multi-label learning algorithms, moreover TRAM is much faster and can handle relatively larger data sets.

Index Terms

Machine learning, data mining, multi-label learning, transductive learning, optimization algorithm, unlabeled data.

I. INTRODUCTION

Multi-label learning refers to the problems when each instance can be associated with multiple classes simultaneously. It has found applications in many real-world learning problems. For example, in text categorization tasks, each document may be assigned to multiple predefined topics, such as *sports* and *entertainment* [32], [38]; in automatic image or video annotation tasks, each image or video may belong to several semantic classes, such as *urban*, *building* and *road etc.* [7], [35]; in bioinformatics, each gene may be associated with a set of functional classes, such as *metabolism* and *protein synthesis* [13]. In all these applications, the task is to output the label set whose size is unknown a priori for the test instances.

Multi-label learning problem is more general than traditional single-label (two-class or multi-class) problems, which can both be cast as a special case for the multi-label learning. One of the most intuitive approach to multi-label learning is to decompose a multi-label learning task into multiple independent binary classification problems, one for each class. The advantage of this approach is that many binary classifiers can be directly used to solve multi-label learning problems. But this kind of method ignores the correlations between different labels of each instance. Thus the expressive power of such a system can be weak [13], [32], [38]. Another approach is to treat the multi-label learning problem as a label ranking problem, *i.e.*, each

instance can belong to all the labels but with priorities, which tries to minimize a ranking loss on the predicted labels [11], [13], [38], [45]. In addition, there are also several approaches directly consider correlations between labels to tackle the multi-label learning problem [17], [30], [32], [40], [44], [52].

Many developments have been achieved in the research on multi-label learning. However, it is noteworthy that conventional multi-label learning research mainly focuses on the setting of supervised learning, *i.e.*, assuming that the amount of training data is sufficient enough for obtaining reliable classifiers. However, real-world applications of multi-label learning often feature a relatively small size of labeled data with a large amount of unlabeled data, *i.e.* in text classification and multimedia annotation, the cost for obtaining labels of data by human effort is often too high. As a result, the amount of labeled data is often small and insufficient for learning an effective and reliable classifier.

To address this issue, we propose and develop a new multi-label learning method (TRAM), which is able to exploit unlabeled data to obtain an effective model for assigning appropriate multiple labels to instances. We first formulate transductive multi-label learning as an optimization problem, then we develop an efficient algorithm which has a closed form solution to solve this optimization problem. Experiments on real-world tasks show that TRAM can effectively make use of unlabeled data information to achieve performance as good as existing state-of-the-art multi-label learning algorithms, but it is much faster than those algorithms and can handle relatively larger data sets.

The rest of this paper is organized as follows. Section II gives a brief summary of related work on multi-label learning and transductive learning. In Section III we formulate our transductive multi-label learning method (TRAM) as an optimization problem. Section IV briefly introduce alpha matting problem and its connection with transductive multi-label learning. Then we derive a closed form solution to the optimization problem and label learning procedure in Section V. Evaluation metrics used in multi-label learning are then briefly introduced and experiments of TRAM on real-world multi-label learning tasks are reported in Section VI. Finally, Section VII gives some concluding remarks.

II. RELATED WORK

A. Multi-Label Learning

Multi-label learning deals with the problems where each example can belong to multiple different classes simultaneously. Traditional two-class and multi-class problems can both be cast as a special case of multi-label learning problem. Thus multi-label problems are inevitably more difficult and complicated to solve than traditional single-label problems (i.e., two-class or multi-class problems). Until now, multi-label learning problem has been studied by a lot of researchers and many algorithms have been developed to solve different real-world application tasks, such as text categorization [10], [16], [25], [32], [38], [40], bioinformatics [13], [45], scene classification [5], association rule mining [36], [39] and image or video annotation [35].

Most researches of multi-label learning focus on text categorization. Some of these algorithms are derived from traditional learning techniques. One famous approach proposed by Schapire and Singer, `BoosTEXTER` [38], is extended from the popular ensemble learning method `AdaBoost` [14]. In the training phase, `BoosTEXTER` maintains a set of weights over both training examples and their labels, which will be incrementally enlarged if examples or labels are hard to be predicted correctly. Elisseeff and Weston [13] presented a kernel method `RANK-SVM` for multi-label classification, by minimizing a loss function named *ranking loss*. Experimental results on the Yeast gene functional classification problem demonstrate its effectiveness. Zhang and Zhou [46] extended the lazy learning algorithm, *kNN*, to a multi-label version, `ML-kNN`. It employs label prior probabilities gained from each example's *k* nearest neighbors and use maximum *a posteriori* (MAP) principle to determine labels. Extension of other traditional learning techniques have also been studied, such as probabilistic generative models [32], [40], decision trees [10], neural networks [45], maximal margin methods [18], [25], maximum entropy methods [17], [52] and ensemble methods [15].

Unlike the previous works that only consider the correlations among different categories, Liu et al. [30] presents a semi-supervised multi-label learning method to exploit unlabeled data as well as category correlations, which is based on constrained non-negative matrix factorization. Generally, in comparison with supervised methods, semi-supervised methods can efficiently make use of the information provided by unlabeled instances. Moreover, Zhou et al. [50], [51] used the Multi-Instance Multi-Label framework to help solving multi-label problems. Sun et al. [26]

employed hypergraph spectral learning to solve multi-label problems.

B. Transductive Learning

The use of unlabeled data has been increasingly popular these years in machine learning society. As in many practical learning problems, we usually need to handle the situations when a small size of labeled data with a large amount of unlabeled data are available, since the unlabeled data are usually much easier to obtain but quite expensive to identify their labels [53]. Roughly speaking, there are three main paradigms of approaches to utilize unlabeled data, that is, semi-supervised learning (*e.g.* [8], [53]) transductive learning (*e.g.* [22], [41]) and active learning (*e.g.* [1], [20]). Semi-supervised learning approaches attempt to automatically exploit unlabeled data usually assuming the test data are different from the unlabeled data; transductive learning approaches attempt to automatically exploit unlabeled data where the test data are exactly the unlabeled data; active learning approaches query an *oracle* for the labels of specific instances in the input space, in order to get better models while minimizing the number of required queries. In this paper, we focus on transductive learning.

Transductive learning was proposed by Vladimir Vapnik [41] in the 1990's where all unlabeled points belong to the test set. Many transductive learning approaches have been proposed. One famous approach would be Transductive SVMs, introduced by [41] and applied to text classification by [22]. They exploit the structure in both training and test data for better positioning the maximum margin hyperplane. Another type of approaches are graph-based methods, which define a graph with the nodes representing both labeled and unlabeled instances, and edges reflect the similarity of instances (*e.g.* [2], [48], [53]). Graph-based approaches usually assumes label smoothness over the graph. One example is to exploit the structure of the entire dataset in search for mincuts [4] or for min average cuts [23] on the graph. Note that although the work described in this paper can also be viewed as a graph-based approach, there is no previous work about graph-based multi-label learning.

III. PROBLEM FORMULATION

We first introduce some notations that will be used throughout this paper. Suppose there are n_l labeled instances $\mathbf{x}_1, \dots, \mathbf{x}_{n_l}$ with their labels $\mathbf{y}_1, \dots, \mathbf{y}_{n_l}$, and n_u unlabeled instances $\mathbf{x}_{n_l+1}, \dots, \mathbf{x}_{n_l+n_u}$, where each $\mathbf{x}_i \in \mathbb{R}^d$ is a d -dimensional feature vector in the input space and

each $\mathbf{y}_i = (y_i^1, \dots, y_i^m)^\top \in \{0, 1\}^m$ is a m -dimensional label vector. Here m denotes the number of all possible labels. Suppose $\mathcal{L} = \{1, \dots, n_l\}$ is the index set for labeled instances and $\mathcal{U} = \{n_l + 1, \dots, n_l + n_u\}$ for unlabeled instances. Let $n = n_l + n_u$ be the total number of instances, $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ and $Y = (\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n)$.

In multi-label learning problems, each instance usually contains multiple contents in respect to different label concepts, and thus be assigned with multiple labels. Intuitively, we assume that each instance is represented as a weighted sum of label concepts. *i.e.* \mathbf{x}_i is represented as a convex combination of m label concepts $\mathbf{c}_i^1, \dots, \mathbf{c}_i^m$:

$$\mathbf{x}_i = \sum_{j=1}^m \alpha_i^j \mathbf{c}_i^j \quad (0 \leq \alpha_i^j \leq 1, \sum_{j=1}^m \alpha_i^j = 1) \quad (1)$$

where $\mathbf{c}_i^j \in \mathbb{R}^d$ is a vector corresponding to the j -th label concept in instance i , and α_i^j denotes the weight (probability) of the i -th instance being assigned to the j -th concept label. Therefore, we can construct a weight matrix α for all instances as shown in Table I, where C^j indicates the j -th label concept. For convenience, here we employ the notation $\alpha_i = (\alpha_i^1, \alpha_i^2, \dots, \alpha_i^m)^\top$ to represent a m -dimensional vector corresponding to the probabilities of instance i with every label, $\alpha_i^\top \mathbf{1} = 1$, and $\alpha^j = (\alpha_1^j, \alpha_2^j, \dots, \alpha_n^j)^\top$ is an n -dimensional vector for all instances' probabilities of having label j . Thus, the matrix $\alpha = (\alpha_1, \dots, \alpha_n) = (\alpha^1, \dots, \alpha^m)^\top$.

Note that for labeled instances, we can encode training set's label information as

$$\alpha_i^j = \begin{cases} \frac{1}{|\mathbf{y}_i|}, & \text{if } y_i^j = 1, \\ 0, & \text{if } y_i^j = 0. \end{cases} \quad (i \in \mathcal{L}, j = 1, 2, \dots, m) \quad (2)$$

where $|\mathbf{y}_i|$ denotes the number of ground truth labels the i -th instance is assigned to. In order to meet the constrains in Eq. 1 (*i.e.*, $0 \leq \alpha_i^j \leq 1, \sum_{j=1}^m \alpha_i^j = 1$), here all the α_i^j 's that corresponds to the ground truth labels are equally weighted as $1/|\mathbf{y}_i|$.

For unlabeled instances, the α_i^j 's and \mathbf{c}_i^j 's in Eq. 1 are all unknown. Thus the original problem is seriously under constrained where we have one equation and many unknown variables. However, in order to predict instance's labels in transductive multi-label learning problems, we only need to learn the weights (α_i^j), no need to know the \mathbf{c}_i^j 's explicitly. Thus, we can employ the *alpha matting* techniques to learn the α_i^j 's' values in transductive multi-label learning problems. In the next section, we briefly introduce *alpha matting* and then show its connection with transductive multi-label learning.

TABLE I: The α matrix for TRAM. Here C^j denotes the label concept j , and x_i denotes the instance i in the whole data set. And α_i^j denotes the alpha value for instance i with the label concept j .

table

	x_1	x_2	\cdots	x_n
C^1	α_1^1	α_2^1	\cdots	α_n^1
C^2	α_1^2	α_2^2	\cdots	α_n^2
\cdots	\cdots	\cdots	\cdots	\cdots
C^m	α_1^m	α_2^m	\cdots	α_n^m

IV. ALPHA MATTING

Alpha Matting [24] is a technique originated from image segmentation problems in computer vision.

A. Two-class alpha matting

Given an image as a set of pixels $\mathcal{I} = \{x_1, \cdots, x_n\}$, where each x_i is the *RGB* color vector for pixel i , each pixel is modeled as a convex combination of a foreground $F = (F_i)$ and a background $B = (B_i)$:

$$x_i = \alpha_i F_i + (1 - \alpha_i) B_i \quad (3)$$

Here $\alpha = (\alpha_i)$ with $0 \leq \alpha_i \leq 1$ for each i indicates the degree of membership of each pixel to the foreground. When $\alpha_i = 1$, the i -th pixel is a certain foreground pixel, while $\alpha_i = 0$, the pixel is a certain background pixel, which is both labeled by users. The problem of image matting is to estimate $\{F_i\}$, $\{B_i\}$ and $\{\alpha_i\}$ for all the unlabeled pixels in the given image $\{x_i\}$. Once $\{F_i\}$, $\{B_i\}$ and $\{\alpha_i\}$ are obtained, we can replace the original background $\{B_i\}$ by another totally different image to synthesize a new image. The result is a new image having the same object but with a different background. While a hard segmentation ($\alpha_i \in \{0, 1\}$) for each i can be used, the resulting synthesized image often looks unnaturally. The idea of image matting is to allow α_i to take a continuum between 0 and 1, presumably at a thin layer around the interface between the foreground and background. In this way, the pixels close to the interface can be synthesized as a blending of the foreground and background, thus looking more natural. The image matting problem is seriously under constrained where we have one equation and three

unknown variables. In order to alleviate the originally ill-posed problem, most alpha matting approaches utilize the strong correlations between nearby image pixels.

Setting α 's for the labeled pixels as,

$$\alpha_i = \begin{cases} 1, & \text{on foreground,} \\ 0, & \text{on background.} \end{cases}$$

Then the task of the alpha matte pulling is to solve the compositing equation Eq. 3 for all the unknown pixels.

Many two-class alpha matting approaches have been developed. One type of approaches is sampling-based. It assumes that colors of an unknown pixel can be explicitly estimated by the nearby pixels labeled by user as foreground or background [9], [37], [42]. Another type of approaches is propagation-based. It assumes the colors of each unknown pixel either in foreground and background region are locally smooth, such as Poisson Matting [24], closed-form matting [27] and methods based on random walks [19].

B. Multi-class alpha matting

Alpha matting approaches can be generalized to multi-class settings, which assume the input image to be a convex combination of multiple foreground image layers F^1, \dots, F^k (e.g., “sky”, “sea” and “beach”) as,

$$\mathbf{x}_i = \sum_{j=1}^m \alpha_i^j F_i^j, \quad (0 \leq \alpha_i^j \leq 1, \quad \sum_{j=1}^m \alpha_i^j = 1,)$$

Each α_i^j for pixel i could be regarded as the opacity of foreground layer j or be viewed as the probability of pixel i belonging to the corresponding foreground region j [28], [33]. And the labeled pixels is set as,

$$\alpha_i^j = \begin{cases} 1, & \text{pixel } i \text{ in foreground region } j, \\ 0, & \text{otherwise.} \end{cases}$$

C. Connection with transductive multi-label learning

It is interesting to see that the formulation in multi-class alpha matting can also be generalized to solve transductive multi-label problems, if we treat each pixel as an instance, each foreground layer as a label concept, *i.e.* labeled pixels as training instances and opacities of foreground layers

α_i^j s as the probabilities of i -th instance being assigned with j -th label concepts. Thus, under this new alpha matting framework, each instance can be assigned to multiple label concepts simultaneously, and we naturally get the same formulation as discussed before in Section III. Moreover, in transductive multi-label learning we only need to solve the α_i^j s for each instance in order to learn the labels.

V. THE PROPOSED METHOD

A. Derivation

Similar to the alpha matting problem stated in Section IV, although the original transductive multi-label learning problem in Section III is severely under-constrained, the strong correlations between neighboring instances can be leveraged to impose constraints in order to make the optimization problem well posed.

We first build a weighted neighborhood graph $G = (V, E)$ based on labeled instances and unlabeled instances, which can naturally characterize the similarity of instances among neighbors. Each vertex corresponds to an instance \mathbf{x}_i , then we put an edge between \mathbf{x}_i and \mathbf{x}_z if \mathbf{x}_i is among the k nearest neighbors of \mathbf{x}_z or \mathbf{x}_z is among the k nearest neighbors of \mathbf{x}_i . So we get a k NN graph with a sparse edge weight matrix.

In order to reduce computational cost of k NN search among instances, in this paper we create the k NN graph efficiently searched by kd-tree. As kd-trees seriously suffer from the curse of dimensionality which will degenerate to linear searches in high dimensions [43], in our context a multi-label dimensionality reduction approach (MDDM [47]) is used before the k NN graph construction.

After the k NN search, we define a sparse $n \times n$ weight matrix W indicating the similarities among neighboring instances:

$$W_{iz} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_z\|^2}{2\sigma^2}\right), & \text{if } z \in \mathcal{N}_i, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

where \mathcal{N}_i is the index set of i -th instance's k nearest neighbors. Typically, $\|\cdot\|$ refers to the Euclidean distance. And parameter σ is empirically estimated as the average distance between instances. Then we normalize the similarity W_{iz} to force $\sum_z W_{iz} = 1$ for all i .

After the k NN graph is constructed, we consider each instance's alpha values (α_i^j) as the probability or weight corresponding to different label concepts, thus we can naturally assume that these alpha values (α_i^j) vary smoothly along the edges of G since neighboring instances are very likely to contain similar label concepts.

So we can learn the optimal alpha values for each unlabeled instance by optimizing the objective function, which minimize the difference between each alpha value (α_i^j) and the weighted reconstruction by its neighboring instances ($\sum_{z \in \mathcal{N}_i} W_{iz} \alpha_z^j$). The reconstruction loss of alpha values for i -th instance is defined as $\sum_{j=1}^m \left(\alpha_i^j - \sum_{z \in \mathcal{N}_i} W_{iz} \alpha_z^j \right)^2$, thus for all the unlabeled instances we derive the following optimization problem:

$$\begin{aligned} \min_{\alpha^1, \dots, \alpha^m} \sum_{i \in \mathcal{U}} \left[\sum_{j=1}^m \left(\alpha_i^j - \sum_{z \in \mathcal{N}_i} W_{iz} \alpha_z^j \right)^2 \right] \\ \text{s.t.} \quad 0 \leq \alpha_i^j \leq 1, \sum_{j=1}^m \alpha_i^j = 1 \\ \alpha_i^j = \bar{\alpha}_i^j \quad (\forall i \in \mathcal{L}) \end{aligned} \quad (5)$$

where the $\bar{\alpha}_i^j$ encodes training set's label information, defined as

$$\bar{\alpha}_i^j = \begin{cases} \frac{1}{|\mathbf{y}_i|}, & \text{if } y_i^j = 1, \\ 0, & \text{if } y_i^j = 0. \end{cases} \quad (i \in \mathcal{L}) \quad (6)$$

Here $|\mathbf{y}_i|$ denotes the number of ground truth labels the i -th instance is assigned to. As we have

$$\begin{aligned} \sum_{i \in \mathcal{U}} \left[\sum_{j=1}^m \left(\alpha_i^j - \sum_{z \in \mathcal{N}_i} W_{iz} \alpha_z^j \right)^2 \right] &= \sum_{j=1}^m \left[\sum_{i \in \mathcal{U}} \left(\alpha_i^j - \sum_{z \in \mathcal{N}_i} W_{iz} \alpha_z^j \right)^2 \right] \\ &= \sum_{j=1}^m \|D_u(\alpha^j - W\alpha^j)\|_2^2 \end{aligned}$$

where D_u consists of the identity matrix for n_u unlabeled data and is zero everywhere else. *i.e.*

$$D_u = \begin{pmatrix} I_u & 0 \\ 0 & 0 \end{pmatrix}_{(n \times n)} \quad \text{provided that the vector } \alpha^j \text{ and matrix } W \text{ are accordingly block-partitioned.}$$

Then, an equivalent optimization problem in matrix form can be derived:

$$\min_{\alpha^1, \dots, \alpha^m} \sum_{j=1}^m \|D_u(I - W)\alpha^j\|_2^2 \quad \text{s.t.} \quad \begin{cases} \mathbf{0} \leq \alpha^j \leq \mathbf{1}, \sum_{j=1}^m \alpha^j = \mathbf{1} \\ \alpha_{\mathcal{L}}^j = \bar{\alpha}_{\mathcal{L}}^j \end{cases} \quad (7)$$

B. A Closed-Form Solution

First, we can see that the objective function and the constraints in Eq.7 are convex, and the feasible set is compact. Therefore a global minimizer exists [33]. Let $A = I - W$ in Eq.7. We split the matrix A and α^j vectors into blocks corresponding to unlabeled instances and labeled instances, as

$$A = \begin{bmatrix} A_{uu} & A_{u\mathcal{L}} \\ A_{\mathcal{L}u} & A_{\mathcal{L}\mathcal{L}} \end{bmatrix} \quad \text{and} \quad \alpha^j = \begin{bmatrix} \alpha_{u}^j \\ \alpha_{\mathcal{L}}^j \end{bmatrix}, (j = 1, \dots, m)$$

where α_{u}^j denotes the probabilities of all unlabeled instances containing j -th label concept. By first ignoring the bilateral constraints $0 \leq \alpha^j \leq 1$, the Lagrange function becomes

$$L(\alpha, \beta, \gamma) = \frac{1}{2} \sum_{j=1}^m \|D_u A \alpha^j\|_2^2 - \beta^\top \left(\sum_{j=1}^m \alpha_{u}^j - \mathbf{1} \right) - \sum_{j=1}^m \gamma^{j\top} (\alpha_{\mathcal{L}}^j - \bar{\alpha}_{\mathcal{L}}^j)$$

where $\beta \geq 0$ and $\gamma^j \geq 0$. The optimal condition for α^j is

$$\frac{\partial L}{\partial \alpha^j} = A^\top D_u^\top D_u A \alpha^j - \begin{bmatrix} \beta \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \gamma^j \end{bmatrix} = \mathbf{0} \quad (8)$$

By summing optimal conditions Eq.8 for all α^j ($j = 1, \dots, m$), we have

$$\sum_{j=1}^m (A^\top D_u^\top D_u A \alpha^j) = \begin{bmatrix} m\beta \\ \sum_{j=1}^m \gamma^j \end{bmatrix}.$$

Then using the constraints $\sum_{j=1}^m \alpha^j = \mathbf{1}$, we have

$$A^\top D_u^\top D_u A \mathbf{1} = \begin{bmatrix} m\beta \\ \sum_{j=1}^m \gamma^j \end{bmatrix}.$$

Notice that the $A\mathbf{1} = (I - W)\mathbf{1} = \mathbf{1} - W\mathbf{1} = \mathbf{0}$. So, the following equations can be derived, $\beta = 0$, $\sum_{j=1}^m \gamma^j = 0$ and then we substitute them into Eq. 8,

$$\begin{bmatrix} A_{uu}^\top A_{uu} & A_{uu}^\top A_{u\mathcal{L}} \\ A_{u\mathcal{L}}^\top A_{uu} & A_{u\mathcal{L}}^\top A_{u\mathcal{L}} \end{bmatrix} \begin{bmatrix} \alpha_{u}^j \\ \alpha_{\mathcal{L}}^j \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \gamma^j \end{bmatrix}.$$

As A_{uu} is nonsingular for a connected graph [3], therefore we get

$$A_{uu} \alpha_{u}^j + A_{u\mathcal{L}} \alpha_{\mathcal{L}}^j = \mathbf{0} \quad (9)$$

By substituting the constrain $\alpha_{\mathcal{L}}^j = \bar{\alpha}_{\mathcal{L}}^j$, the optimal alpha values of unlabeled instances for class j (i.e. α_{u}^j) can be calculated by the following linear equation:

$$A_{uu} \alpha_{u}^j = -A_{u\mathcal{L}} \bar{\alpha}_{\mathcal{L}}^j \quad (10)$$

which is a sparse, symmetric, positive-definite, linear system. The number of equations equal to n_u and the number of nonzero entries is less than $(k+1) \times n_u$. Here, the solution $\alpha_{\mathcal{U}}^j$ is guaranteed to exist and be unique with values guaranteed to lie between 0 and 1. The proofs can be found in [33], we put them in the Appendix section to make the paper self-contained.

After the optimal alpha matte is solved, the alpha value outputs can naturally be used for label ranking. Then, in order to predict label set, a natural way is to set a threshold and use alpha to make predictions. Nevertheless, in this paper, a label set predicting mechanism is used which generalizes the above natural solution. In detail, the label predicting function $f(\alpha(\mathbf{x}))$ is modeled by a linear function $f(\mathbf{x}) = P\alpha(\mathbf{x})$, where $\alpha(\mathbf{x}) = (\alpha^1(\mathbf{x}), \dots, \alpha^m(\mathbf{x}))$ is the m -dimensional vector of the actual alpha values for unlabeled instance \mathbf{x} , and the procedure used to learn the parameters is described as follows:

We first perform the leave-one-out *alpha matting* process on the training set to estimate the alpha matting output for training instances. By combining $\hat{\alpha}_i^j (i \in \mathcal{L})$ into vector, the estimated alpha matting output for all training instances can be solved by the following equation:

$$\hat{\alpha}_{\mathcal{L}}^j = (I - A_{\mathcal{L}\mathcal{L}})\alpha_{\mathcal{L}}^j = W_{\mathcal{L}\mathcal{L}}\alpha_{\mathcal{L}}^j \quad (j = 1, \dots, m) \quad (11)$$

Suppose the output vector for instance i is $\hat{\alpha}_i = (\hat{\alpha}_i^1, \hat{\alpha}_i^2, \dots, \hat{\alpha}_i^m)^\top$ ($i \in \mathcal{L}$). The ground-truth labels for instance i are known, *i.e.* $\mathbf{y}_i \in \{0, 1\}^m$. Here for convenience of prediction, we set these ground-truth labels as $\tilde{\mathbf{y}}_i \in \{-1, 1\}^m$. Then, the label could be learned by solving a linear regression problem to get the linear transformation matrix P , by minimizing the following sum-of-squares error function with a regular term,

$$P = \arg \min_P \sum_{i \in \mathcal{L}} \|\tilde{\mathbf{y}}_i - P\hat{\alpha}_i\|_2^2 + \lambda \sum_j \|P_j\|_2^2$$

where P_j denotes the j -th row of matrix P . Then the solution is

$$P = \tilde{\mathbf{y}}_{\mathcal{L}}\hat{\alpha}^\top (\hat{\alpha}\hat{\alpha}^\top + \lambda I)^{-1}. \quad (12)$$

In practice, in order to avoid singularity, we set λ as a small number (it is set to be 1×10^{-7} in the experiment). Then, with the linear transforms matrix P , we can predict label vector for unlabeled instances from their alpha matting output vectors by

$$\mathbf{y}_i = \text{sign}(P\alpha_i) \quad (\forall i \in \mathcal{U}).$$

The TRAM method is briefly summarized in Figure 1.

$$(Y_{\mathcal{U}}, \alpha_{\mathcal{U}}) = \text{TRAM}(X, Y_{\mathcal{L}})$$

Input:

$X : (\mathbf{x}_1, \dots, \mathbf{x}_n)$ encoding features of the whole data set

$Y_{\mathcal{L}}: (\mathbf{y}_1, \dots, \mathbf{y}_l)$ encoding labels of training set

Process:

- 1 Construct k NN graph among instances.
- 2 Initialize the similarities on each edge as $W_{iz} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_z\|^2}{2\sigma^2})$ and normalize to $\sum_z W_{iz} = 1$;
- 3 Determine the $\alpha_{\mathcal{U}}^j$ values for all unlabeled data by solving the alpha matting linear system in Eq.10;
- 4 Compute the label set prediction matrix P by solving Eq.12;
- 5 Predict label set for each unlabeled instance by $\mathbf{y}_i = \text{sign}(P\alpha_i)$ ($\forall i \in \mathcal{U}$).

Output:

$Y_{\mathcal{U}}$: the predicted labels for unlabeled instances.

$\alpha_{\mathcal{U}}$: the alpha value outputs for unlabeled instances.

Fig. 1: The TRAM algorithm

figure

C. Computational complexity

Beyond the computational cost of MDDM dimensionality reduction ($O(m \cdot n)$) in the training step and the neighborhood graph searched by kd-tree ($O(n \log n)$) in the testing step, the alpha solutions and the label learning procedure of TRAM involve the following costs: In the worst case, the least-square solution of the linear systems in Eq.10 requires $O(n_u^3 + n_l \cdot n_u)$ operations when all data points are connected in a full graph (*i.e.* $k = n$). However, this cost can be significantly reduced using a k -nearest neighbor graph ($k \ll n$) which leads directly to a sparse matrix ($A_{\mathcal{U}\mathcal{U}}$). Thus the linear systems are large, sparse, symmetric, many good solvers can be employed, *e.g.*, direct methods (*e.g.*, LU factorizations), or iterative solvers [21]. In practice, “the cost of computing the sparse LU factorization depends in a complicated way on the size of $A_{\mathcal{U}\mathcal{U}}$, the number of nonzero elements, its sparsity pattern, but is often dramatically smaller than the cost of a dense LU factorization. In many cases the cost grows approximately linearly with n_u , when n_u is large. This means that when $A_{\mathcal{U}\mathcal{U}}$ is sparse, we can solve $A_{\mathcal{U}\mathcal{U}}\alpha_{\mathcal{U}}^j = \mathbf{b}$ very efficiently, often with an order approximately n_u ” [6].

For simplicity, we have used QR factorization designed for sparse matrix in `MATLAB` to compute the R factor very cheaply, which avoids the expensive computation of an explicit Q, details are described in [31]. Then for label learning procedure of TRAM, the computation of $\hat{\alpha}_L^j$ and transforms matrix P costs respectively $O(m \cdot n_l)$ and $O(n_l \cdot m + m^3)$.

The computational complexity of RANK-SVM [13] is currently of the order $O(m \cdot n_l^2)$ in each iteration for training. ML-KNN [46] as a lazy learning algorithm requires $(O(n_l^2 + n_l \cdot m))$ for training, and $O(n_l \cdot n_u + n_u \cdot m)$ for testing. BOOSTEXTER [38] requires $O(n_l \cdot m)$ for each iteration round in training with additional cost for the training of base learners. CNMF [30] as a transductive learning method requires $O(n^2)$ for similarity calculation between samples and $O(m \cdot n_u)$ in each iteration for testing.

VI. EXPERIMENTS

In this section, we implement the TRAM method described in Section V and illustrate the performance of TRAM on several real-world multi-label tasks. Table II summarizes the characteristics of the data sets used. For comparison, we also compare with several general-purpose multi-label learning algorithms, including CNMF [30], BOOSTEXTER [38], RANK-SVM [13] and ML-KNN [46], which are applicable to various multi-label problems, and represent the state-of-the-art techniques in multi-label learning:

1. TRAM: The proposed algorithm TRAM, *i.e.* a transductive multi-label learning algorithm by Alpha Matting (implementation in `MATLAB`);
2. CNMF: The CNMF [30] is a semi-supervised multi-label learning algorithm by constrained non-negative matrix factorization. The key assumption behind CNMF is that two instances tend to have large overlap in their assigned class memberships if they share high similarity in their input patterns. By minimizing the difference between inputs similarity with class label overlaps, CNMF can determine the labels of unlabeled data;
3. BOOSTEXTER: The BOOSTEXTER [38] (implementation in C) is a Boosting style multi-label ranking system, which has been shown with excellent performance in previous studies, especially on text categorization tasks;
4. RANK-SVM: The RANK-SVM [13] (implementation in `MATLAB`) is an SVM style multi-label learning algorithm which minimizes ranking loss directly and has also exhibited excellent performance in previous studies;

5. **ML-KNN**: The **ML-KNN** [46] (implementation in **MATLAB**) is a k NN style multi-label learning algorithm which often outperforms other existing multi-label algorithms.

Parameters are used in their default settings unless otherwise specified. For **BoosTEXTER**¹, the number of boosting rounds is set to 500 because on all data sets studied in this paper, the performance of **BoosTEXTER** will not significantly change after the specified boosting rounds; For **RANK-SVM**, parameters were selected by 5 fold cross validation, with the cost parameter in $\{0.01, 0.1, 1, 10, 100\}$ and polynomial kernel degrees in $\{2, 4, 6, 8, 10\}$, including the best parameter reported in the literature [13]; For **CNMF**, the best parameters in [30] are used.

Our **TRAM** implementation is in **MATLAB** and the size of neighbors k is 10. Moreover, the influence of **TRAM**'s parameters will be discussed in Section VI-G.

A. Evaluation Metrics

Multi-label learning systems require much more complicated evaluation criteria than traditional single-label systems. In this section we briefly summarize the criteria used for performance evaluation from various perspectives. Since our approach not only produces a ranked list of class labels, but also produces a predicted label set, in this paper we employ two sets of evaluation metrics to evaluate the performance of label ranking as well as the label set prediction. Adopting the same notations as used in Section III, for a test set $\mathcal{S}_U = \{(\mathbf{x}_{l+1}, \mathbf{y}_{l+1}), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$, the following multi-label evaluation criteria are used in this paper, which have been used in [13], [38], [45], [46].

Label Ranking Performances: The first group of evaluation criteria are concerning algorithm's label ranking performance for each instance, they are based on the real-valued output function $f : \mathbb{R}^d \times \mathcal{Y} \rightarrow \mathbb{R}$ of each algorithm, here $\mathcal{Y} = \{1, 2, \dots, m\}$.

1) *Ranking loss*: evaluates the average fraction of label pairs that are not correctly ordered.

$$\text{RankLoss}(f, \mathcal{S}_U) = \frac{1}{|\mathcal{S}_U|} \sum_{i \in U} \frac{1}{|\mathbf{y}_i| |\bar{\mathbf{y}}_i|} |\{(y_1, y_2) \in \mathbf{y}_i \times \bar{\mathbf{y}}_i | f(\mathbf{x}_i, y_1) \leq f(\mathbf{x}_i, y_2)\}|$$

Where the $\bar{\mathbf{y}}$ denotes the complementary set of \mathbf{y} in $\{0, 1\}^m$. The performance is perfect when $\text{RankLoss}(f) = 0$. The smaller the value, the better the performance.

¹<http://www.cs.princeton.edu/~schapire/boostexter.html>

- 2) *Average Precision*: evaluates the average fraction of labels ranked above a particular label $y \in \mathbf{y}$ which actually is in \mathbf{y} .

$$AvePrec(f, \mathcal{S}_{\mathcal{U}}) = \frac{1}{|\mathcal{S}_{\mathcal{U}}|} \sum_{i \in \mathcal{U}} \frac{1}{|\mathbf{y}_i|} \sum_{y \in \mathbf{y}_i} \frac{|\{y' \in Y_i | r_f(\mathbf{x}_i, y') \leq r_f(\mathbf{x}_i, y)\}|}{r_f(\mathbf{x}_i, y)}$$

The bigger the value, the better the performance.

- 3) *One-error*: evaluates how many times the top-ranked label is not in the set of ground-truth labels of the instance.

$$OneError(f, \mathcal{S}_{\mathcal{U}}) = \frac{1}{|\mathcal{S}_{\mathcal{U}}|} \sum_{i \in \mathcal{U}} \llbracket [\arg \max_{1 \leq y \leq m} f(\mathbf{x}_i, y)] \notin \mathbf{y}_i \rrbracket$$

where for any predicate π , $\llbracket \pi \rrbracket$ equals 1 if π holds and 0 otherwise. The performance is perfect when $OneError(f, \mathcal{S}_{\mathcal{U}}) = 0$. The smaller the value, the better the performance.

- 4) *Coverage*: evaluates how far, on average, we need to go down label ranking list to cover all the ground-truth labels of the instance.

$$Coverage(f, \mathcal{S}_{\mathcal{U}}) = \frac{1}{|\mathcal{S}_{\mathcal{U}}|} \sum_{i \in \mathcal{U}} \max_{y \in \mathbf{y}_i} r_f(\mathbf{x}_i, y) - 1$$

where $r_f(\cdot, \cdot)$ is the ranking function mapping from the outputs of $f(\mathbf{x}_i, y)$ for any $y \in \mathbf{y}_i$ to $\{1, 2, \dots, m\}$, such that if $f(\mathbf{x}_i, y_1) > f(\mathbf{x}_i, y_2)$, then $r_f(\mathbf{x}_i, y_1) < r_f(\mathbf{x}_i, y_2)$. It is loosely related to precision with a perfect recall.

Label Set Prediction Performances: The last evaluation criterion is concerning algorithm's performance on label set prediction for each instance. It is based on multi-label classifier's label prediction function $h : \mathbb{R}^d \rightarrow \{0, 1\}^m$, assume $h(\mathbf{x}_i)$ be the set of labels predicted by a multi-label classifier for instance \mathbf{x}_i .

- 5) *Hamming loss*: evaluates how many times an instance-label pair is misclassified.

$$HammingLoss(h, \mathcal{S}_{\mathcal{U}}) = \frac{1}{|\mathcal{S}_{\mathcal{U}}|} \sum_{i \in \mathcal{U}} \frac{1}{m} |h(\mathbf{x}_i) \Delta \mathbf{y}_i|$$

where Δ stands for the symmetric difference of two sets. The smaller the value, the better the performance. This is one of the most important multi-label criteria.

Note that all the criteria evaluate the performance of multi-label learning systems from different aspects. Usually few algorithms could outperform another algorithm on all those criteria. In order to make our evaluation criteria more comprehensive, we will use the value of $1 - AvePrec$ to replace the original *Average Precision*. Thus under all these evaluation criteria, smaller values are all indicating better performances.

In addition to these evaluation criteria, we also compare the CPU time consumed by each multi-label learning algorithm on each task, including training time, testing time and total time costs. All experiments are conducted on machines with 32GB RAM and 32 Intel Xeron™ CPUs of 2.13 GHz².

TABLE II: Summary of experimental data sets

table

Task Studied	Data Set	# Instances	# Attributes	# Labels
Automatic Image Annotation	annotation	4,800	500	43
Gene Functional Analysis	yeast	2,417	103	14
Web Page Categorization	yahoo (11 subsets)	5,000	(462 ~ 1,047)	(21 ~ 40)
Text Categorization	RCV1-v2	103,149	9,479	68
Natural Scene Classification	scene	2,407	294	6

B. Application to Automatic Image Annotation

We first experiment with the automatic image annotation task on Coral dataset used in [12]. The original data set contains 5,000 images each was segmented into several regions and tagged with several words. The regions of similar features are clustered into 500 clusters, known as blobs [12]. Then, each image is represented by a binary vector of these 500 blobs. The average annotated words for each image is 3.5. We remove the words that occur less than 100 times, and obtain 4,800 images and 43 annotation words.

This data set is partitioned randomly into labeled/unlabeled data sets according to certain ratios, the same setup in [49]. In detail, we randomly draw 25% of the data as unlabeled test examples and use the remaining 75% of the data to generate training examples under different *label rates* from 10% to 90%. For instance, assuming the data set contains 4,800 examples and the label rate is 20%, we randomly draw 1,200 examples as test examples; and $3,600 \times 20\% = 720$ examples from the remaining data set as labeled training examples. Thirty runs of experiments

²BoosTEXTER was implemented in C, while other algorithms were all developed in MATLAB. So the comparison of time costs between BoosTEXTER and other algorithms could only be used as a reference.

are conducted under every label rate; in each run, algorithms are evaluated on random data set partitions and the average performance and 95% confidence intervals of t-test are recorded.

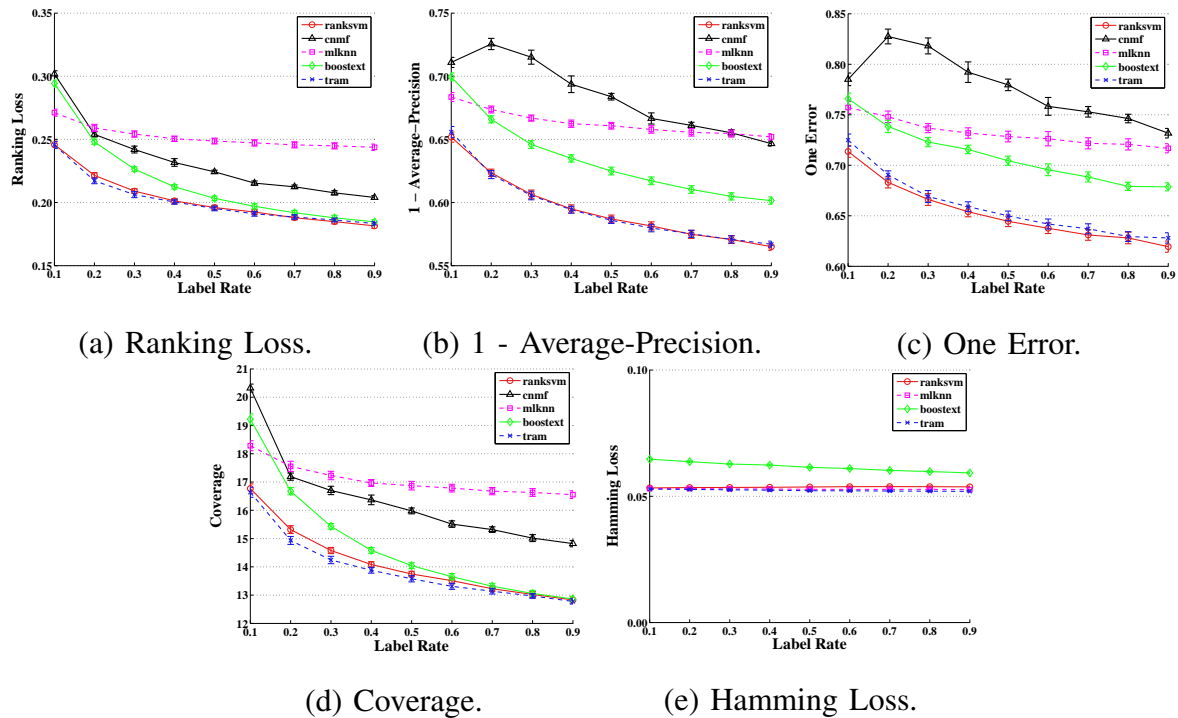


Fig. 2: Results on automatic image annotation task under different label rates. The lower the curve, the better the performance. Along the curves, we also plot the 95% confidence intervals of t-test.

figure

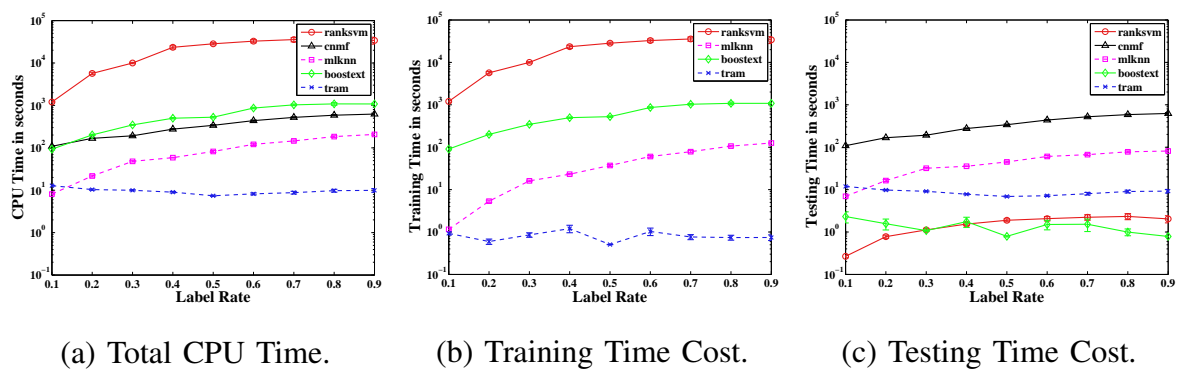


Fig. 3: Time costs on automatic image annotation task with different label rates. Note that the time costs are in log scale. Along the curves, we also plot the 95% confidence intervals of t-test.

figure

Experimental results are shown in Figure 2³. As can be seen, TRAM is as accurate as the other algorithms. TRAM’s performance concerning label ranking evaluation criteria, *i.e.*, ranking loss, average precision, one-error and coverage, are all as good as or better than other methods. This may be explained by that, TRAM not only uses similarities between training data and unlabeled data, but also uses similarities among unlabeled data for solving the optimal alpha values, which may significantly help to improve the ranking performance especially when there are not sufficient training data.

In Figure 3, we not only record the total CPU time cost for each method, but also record the training time and unlabeled classification time separately. As can be seen, TRAM requires very little training time for the MDDM step, and in the unlabeled classification step, the time cost of TRAM is not as fast as BOOSTEXTER and RANK-SVM, but is much faster than other transductive learning method (CNMF) and lazy learning method (ML-KNN). And in total CPU time cost, which both counts training and testing time together, TRAM is much faster than other methods as the size of the training set increases.

C. Application to Yeast Gene Functional Analysis

The task of the yeast gene functional analysis has been studied as a multi-label learning problem in many works (e.g., [13] and [34]). Following [13], we aim at predicting the functional classes in the gene of yeast *Saccharomyces cerevisiae*. These functional classes are structured into 4 levels of hierarchies⁴. As in [13], only top level hierarchy is considered. The whole data set has 2,417 instances of genes and 14 possible class labels. Each of the gene is represented by a 103-dimensional vector and the average number of class labels is 4.24 ± 1.57 for each instance.

The data set is partitioned randomly into labeled/unlabeled data sets according to certain ratios, the same setup as in the automatic image annotation task. Thirty runs of experiments are conducted under every label rate; in each run, algorithms are evaluated on random data set partitions and the average performance is recorded.

As can be seen from Figure 4 and Figure 5, TRAM is, again, as accurate as the others. Recall that the criteria used in this experiment evaluate the performance of multi-label learning

³Evaluation results of *Hamming Loss* is not available for CNMF.

⁴Details in <http://mips.gsf.de/proj/yeast/catalogues/funecat/>.

algorithms from different aspects. Usually few algorithms could outperform other algorithms on all those criteria. Moreover, for satisfactory classification performance, usually different task problems require different parameters for algorithms, but TRAM uses the same parameters in all

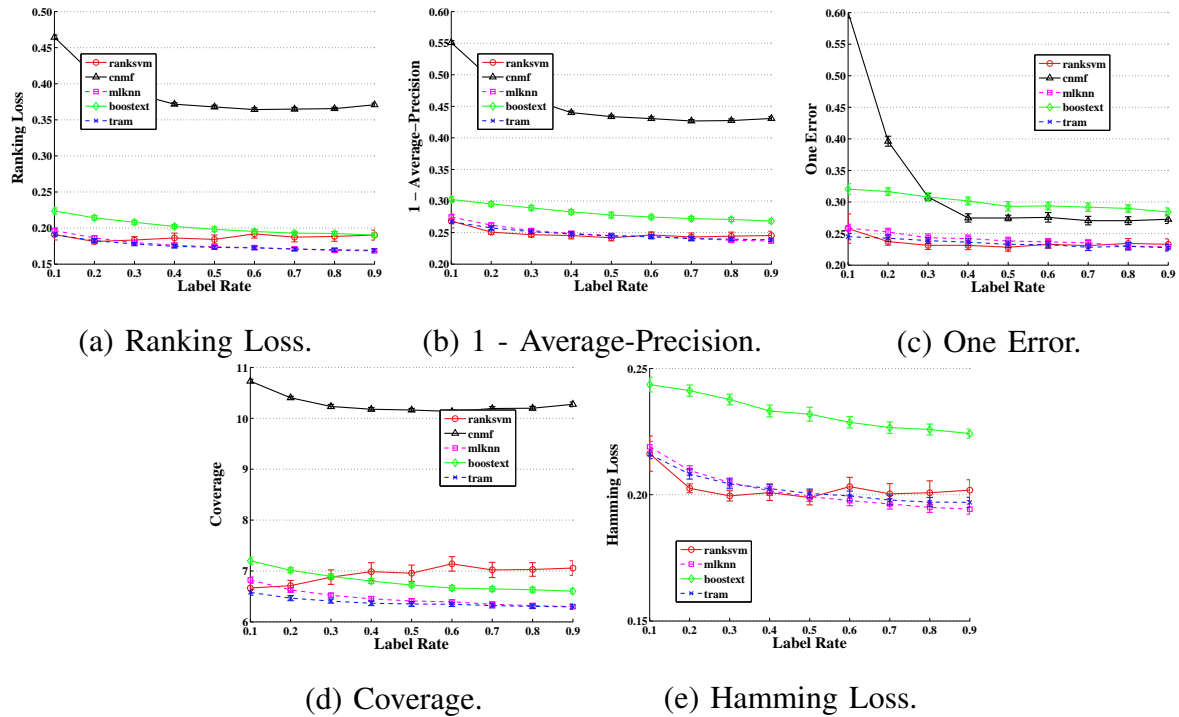


Fig. 4: Results on yeast gene function analysis task with different label rates. The lower the curve, the better the performance. Along the curves, we also plot the 95% confidence intervals of t-test.

figure

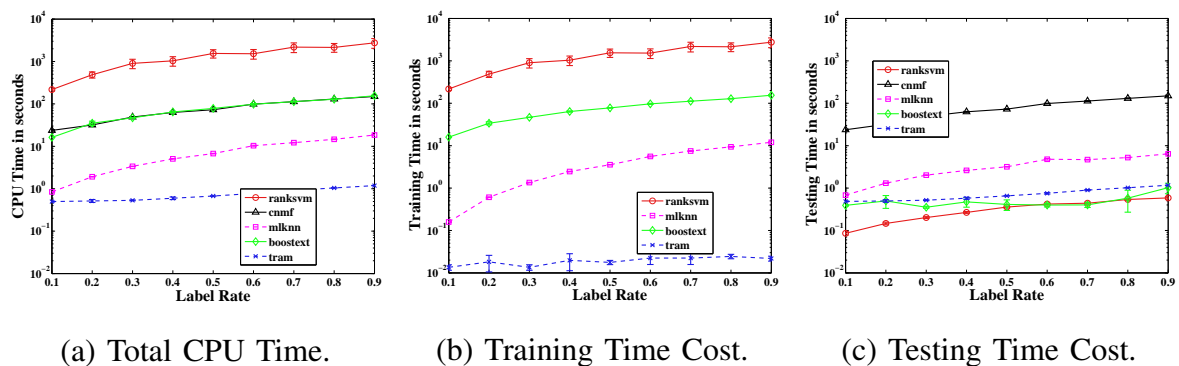


Fig. 5: Time costs on yeast gene function analysis task with different label rates. Note that the time costs are in log scale. Along the curves, we also plot the 95% confidence intervals of t-test.

figure

TABLE III: Data subsets used in the automatic web page categorization task. “*MDoc%*” denotes the percentage of web pages belonging multiple categories, and “*#AveLabel*” represents the average number of labels for each web page.

table

Data Subset	Number of Labels	Vocabulary Size	Training Set		Test Set	
			<i>MDoc%</i>	<i>#AveLabel</i>	<i>MDoc%</i>	<i>#AveLabel</i>
Arts&Humanities	26	462	44.50%	1.627	43.63%	1.642
Business&Economy	30	438	42.20%	1.590	41.93%	1.586
Computers&Internet	33	681	29.60%	1.487	31.27%	1.522
Education	33	550	33.50%	1.465	33.73%	1.458
Entertainment	21	640	29.30%	1.426	28.20%	1.417
Health	32	612	48.05%	1.667	47.20%	1.659
Recreation&Sports	22	606	30.20%	1.414	31.20%	1.429
Reference	33	793	13.75%	1.159	14.60%	1.177
Science	40	743	34.85%	1.489	30.57%	1.425
Social&Science	39	1,047	20.95%	1.274	22.83%	1.290
Society&Culture	27	636	41.90%	1.705	39.97%	1.684

tasks without many parameters needed to tune, and could still achieve satisfactory classification performances as accurate as the others. Nevertheless, the influence of parameters will be further discussed in Section VI-G

D. Application to Automatic Web Page Categorization

The web page categorization task has been studied in [25], [40], [46]. In this experiment, our task is to classify web pages in a collection of eleven data subsets⁵. The web pages were collected from the “yahoo.com” domain, represented by the form of “*Bag-of-Words*”, *i.e.* each dimension of the feature vector represents the number of times a word appearing in the web page. Each data subset corresponds to a top-level category (e.g. “Entertainment”, “Education”, etc.), which contains 2,000 web pages in the training set and 3,000 web pages in the test set. Each web page is assigned to several second-level categories and may belongs to multiple categories simultaneously.

⁵Data set available at <http://www.kecl.ntt.co.jp/as/members/ueda/yahoo.tar.gz>

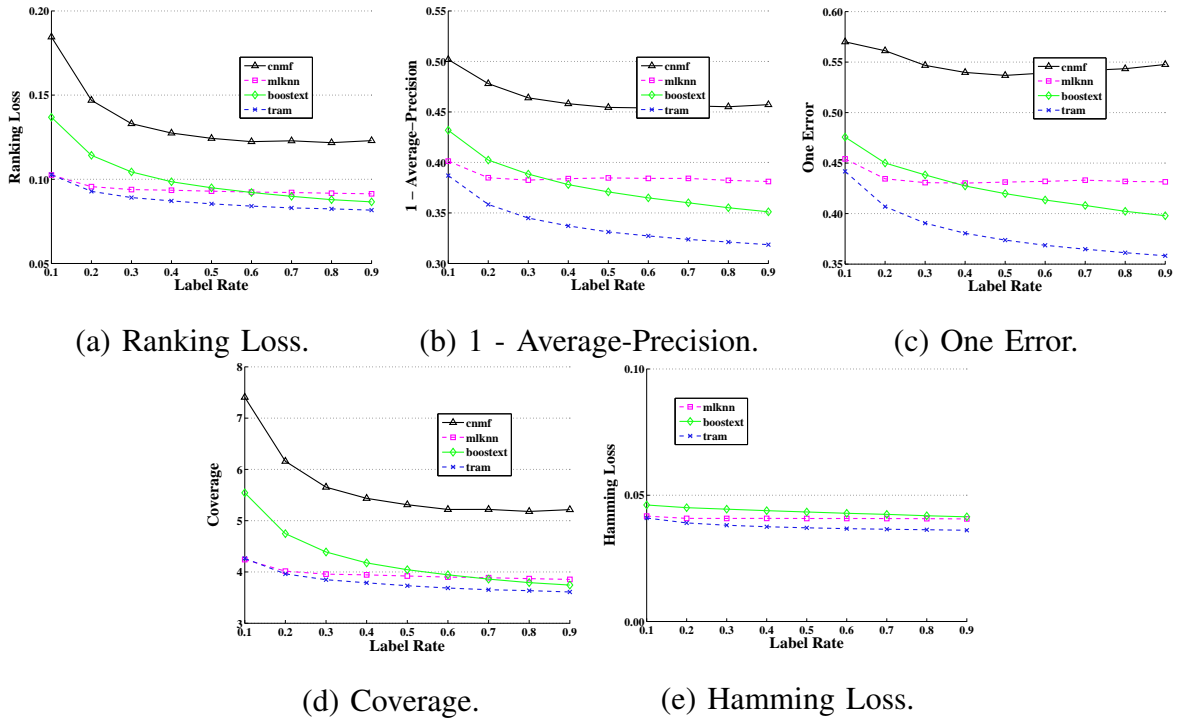


Fig. 6: Results on automatic web page categorization task with different label rates. Note that the values in each figure are reported as the geometrical means across the 11 data subsets.

figure

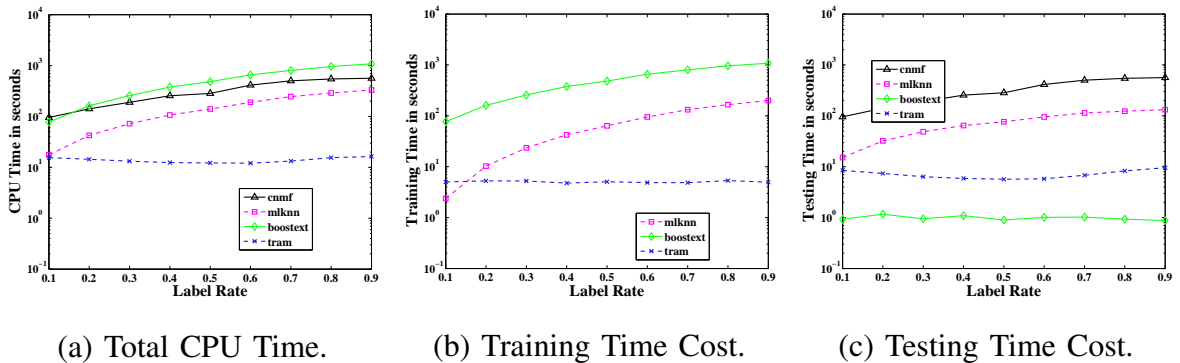


Fig. 7: Time costs on automatic web page categorization task with different label rates. Note that the values in each figure are reported as the geometrical means across the 11 data subsets and the time costs are in log scale.

figure

The web page data subsets are briefly summarized in Table III. Details of these data subsets can also be found in [46]. Comparing with the data sets used in previous tasks, the number of instances and size of vocabulary size in those 11 data subsets are much larger. Furthermore, a

larger percentage of instances (about 30% ~ 40%) are assigned to multiple labels. Thus, the data subsets used in automatic web page categorization tasks are more difficult to learn from.

The same setup as in previous experiments are used to randomly partition the data subset into labeled/unlabeled sets according to different label rates. To make a more meaningful comparison among 11 subsets, we used the geometrical means of the evaluation values across the 11 data subsets instead of simply using the average values. Such that, only the algorithms that can perform well over all 11 data subsets will have good performance values.

The comparison results can be seen in Figure 6 and Figure 7. Surprisingly, our TRAM on this task takes significantly less computational time and yields a lower testing errors than all other methods in most evaluation criteria. Moreover, TRAM’s CPU time is not significantly increased as the *label rate* of training data increases. This could be explained by the closed form solution for solving the alpha matting problem. The size of the alpha matting linear system in TRAM is the same as the number of testing examples, which is fixed when label rate rises. Nonetheless, the computational cost for label set prediction step is linear to the size of training samples.

E. Application to Text Categorization

In this Section, we perform text categorization using RCV1-v2 dataset [29]. The original data set has 804,414 documents, and 47,236 features. To better study the scaling behavior of various algorithms, we use two subsets of the RCV1-v2. The size of Set A is relatively small, while Set B is quite larger. The following subsets are thus created:

TABLE IV: Summary of data subsets in text categorization task

table

Data Set	# Documents	# Attributes	# Labels
Set A	6,000	4,238	54
Set B	103,149	9,479	68

1. Set A: This subset is rcv1v2 (topics; subset)⁶, and contains 6,000 documents. We removed the words that occur less than 5 times and topics with less than 50 positive examples, thus obtain 4,238 words and 54 topics. Note that the number of examples in this subset (6,000)

⁶Data set available at <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multi-label.html>

is much larger than in previous tasks in this paper, and the dimensionality (4,238) is also very high.

2. Set B: This subset contains documents at the scale of 10^5 , *i.e.* 103,149 documents are randomly sampled from the full set of RCV1-v2. We removed the words that occur less than 10 times and topics with less than 100 positive examples, and obtain 9,479 words and 68 topics. This data subset is very large, in either the number of instances (103,149), or the dimensionality (9,479), or the number of class labels (68).

In this experiment, a sparse matrix data format for instance features and similarities is used, which is more appropriate for this data set. Recall that the intent of this experiment is on studying the scaling behavior rather than on obtaining state-of-the-art text categorization performance. In Set A, we use the same setup as in previous experiments and only give the results of the algorithms that cost less than 10^4 seconds for running each round. In Set B, 10-fold cross validation is executed. We could only give the result of TRAM on Set B, as all the comparing methods are not designed to handle 90K training samples with high dimensionality, and suffer from memory problems, therefore, could not work when applied to the Set B. Nevertheless, as Set A and Set B are both RCV1-v2’s subset, the ability of TRAM in handling text categorization task could still be seen from the comparison result on Set A.

The experimental results on Set A are reported in Figure 8 and Figure 9. As can be seen, TRAM performs well on the relatively small Set A in nearly all criteria. Since that the Set A’s features are very high dimensional, the time cost of TRAM is dominated by the extra time costs used by MDDM in training step. while, in testing step, the computational cost for TRAM is still not significantly increased when the label rate raises.

The experimental results on Set B are reported in Table V, which demonstrates the ability of TRAM in handling large dataset. TRAM is able to handle Set B with 10^5 samples and near 10,000 dimensionalities, while other algorithms all encounter memory problems. This can be explained by that, in TRAM’s matrix pulling step, the linear system is at the size of test examples and the computational cost for label set prediction is linear to the size of training samples. Thus make our TRAM algorithm efficient and scalable to this large-scale data set.

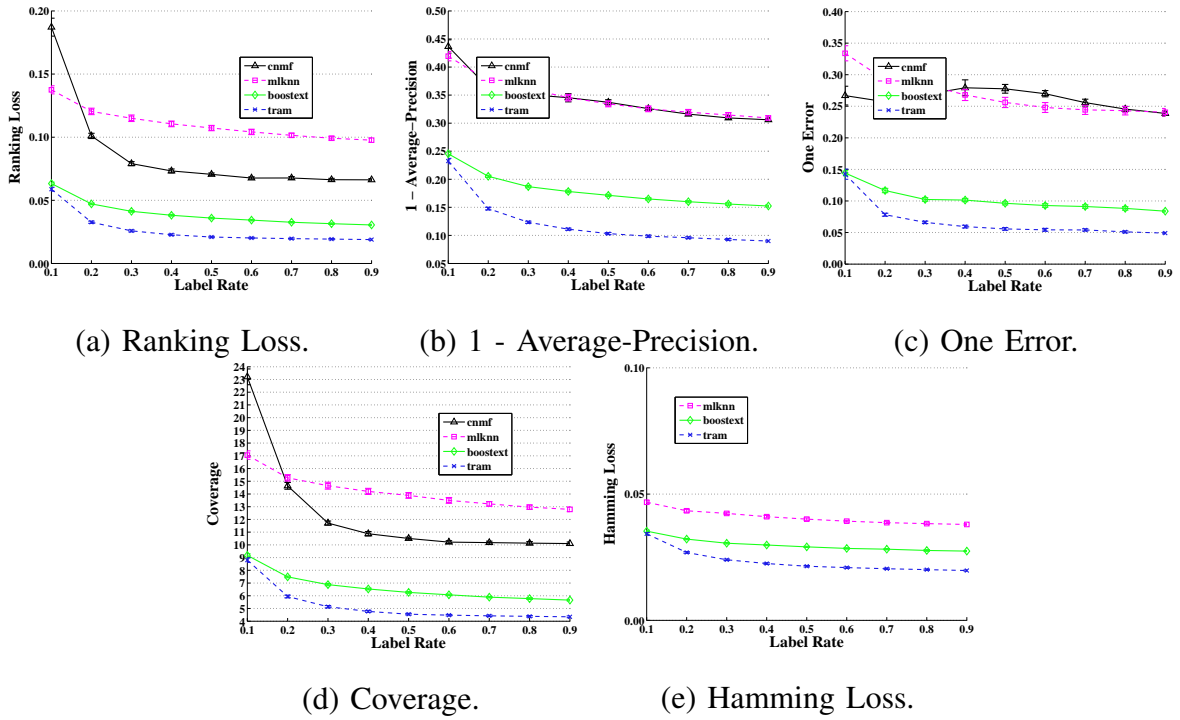


Fig. 8: Results on text categorization task (with Set A) under different label rates. The lower the curve, the better the performance. Along the curves, we also plot the 95% confidence intervals of t-test.

figure

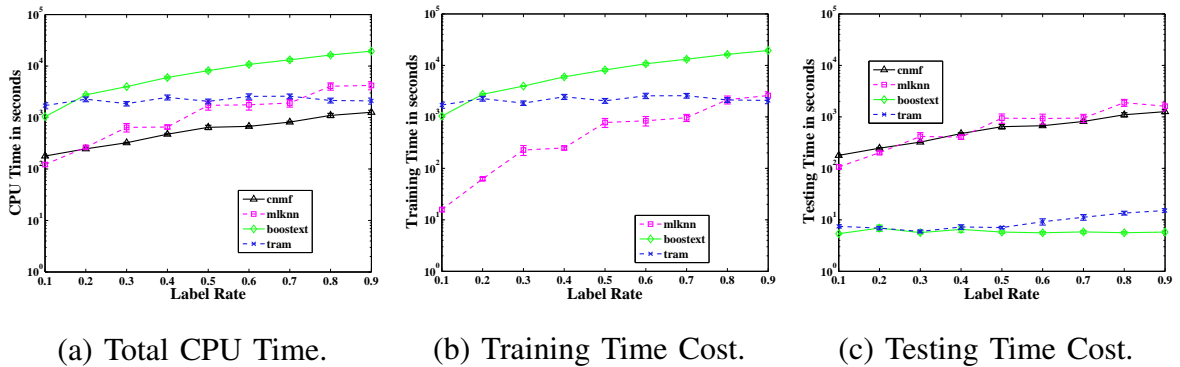


Fig. 9: Time costs on text categorization task (set A) under different label rates. Note that the time costs are in log scale. Along the curves, we also plot the 95% confidence intervals of t-test.

figure

F. Application to Natural Scene Classification

The last multi-label task studied in this paper is natural scene classification. The data set is relatively small, and consists of 2,400 natural scene images belonging to different classes, which

is also used in [5]. Following [5], we first convert each color image to the CIE Luv space, where the Euclidean distances closely correspond to the color differences perceived by human. Then the image is divided into 7×7 blocks using grids of equal width, and in each block the first and second moments of each color band are calculated, which is equal to resizing the image to a low-resolution and calculating simple texture features. Thus, each image is represented as a feature vector with $7 \times 7 \times 3 \times 2 = 294$ -dimensions. The percentage of images that have multiple labels is over 22%. The same setup as in previous experiments are used to randomly partition the data set into labeled/unlabeled sets according to different label rates.

TABLE V: Results (mean \pm std.) on text categorization task with Set B (Except for the TRAM, the other approaches have to terminate early because of not enough memory and/or the training time is too long).

table

Criteria	Ranking Loss ($\times 10^{-1}$)	1 - AveragePrecision	One-error ($\times 10^{-1}$)	Coverage	Hamming Loss ($\times 10^{-1}$)
TRAM	0.105 ± 0.004	0.062 ± 0.001	0.301 ± 0.016	3.888 ± 0.069	0.130 ± 0.002

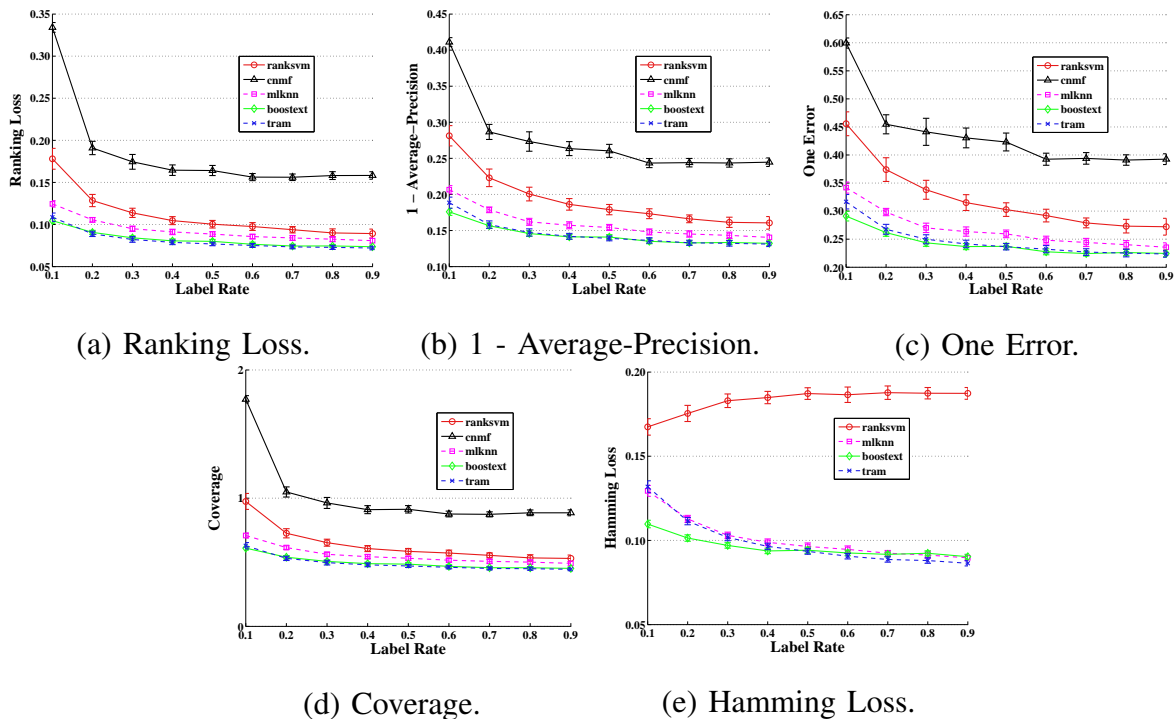


Fig. 10: Results on natural scene classification task with different label rates. The lower the curve, the better the performance. Along the curves, we also plot the 95% confidence intervals of t-test.

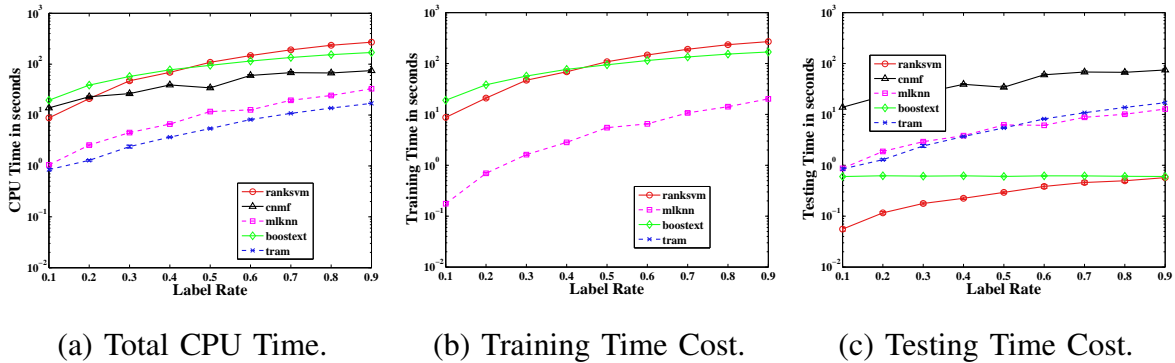


Fig. 11: Time costs on natural scene classification task with different label rates. Note that the time costs are in log scale. Along the curves, we also plot the 95% confidence intervals of t-test.

figure

As can be seen in Figure 10 and Figure 11, TRAM is still among the most accurate methods. However, since this data set is relatively small, when the label rate is less than 20%, the training data set are even smaller. The TRAM’s performances are still stable as the labeled instances decrease to 10% in label rate, thus TRAM does not overfit much when labeled instances are too few, although the performance BoosTexter is better than TRAM in some criteria e.g. Hamming Loss, when label rate is lower than 30%.

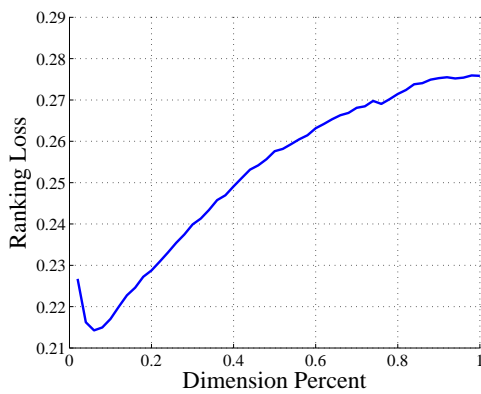
G. The Influence of Parameters

As observed in previous sections, when TRAM is used with the same parameters in all the multi-label tasks, it can all achieve satisfactory classification performances as accurate as the others. In this section, we analyze the influence of parameters in TRAM.

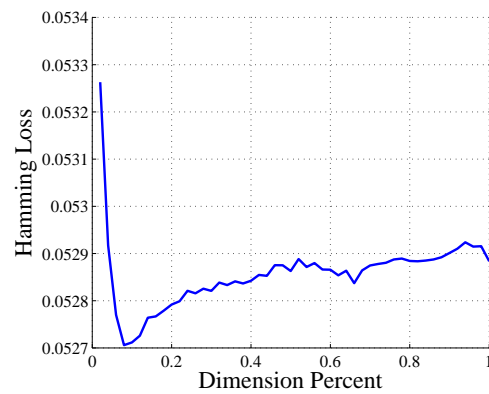
The first exploration is about the number of nearest neighbors during the instance graph construction. The experiment is based on automatic image annotation task. We randomly partition the dataset into labeled and unlabeled data with 20% label rate. The experiment result of TRAM is reported in Table VI, when the number of nearest neighbor during the graph construction varies from 8 to 12. The value following “ \pm ” gives the standard deviation and the best result on each metric is shown in bold face. With respect to above configurations, Table VI shows that the number of nearest neighbors used in graph construction step does not significantly affect TRAM’s performance. Therefore, all the results of TRAM shown in this paper are obtained with the parameter k set to be the moderate value of 10.

TABLE VI: Results (mean \pm std.) of TRAM with different number of nearest neighbors considered in the instance graph construction step on automatic image annotation task (“ \downarrow ” indicates “the smaller the better”, and “ \uparrow ” indicates “the larger the better”).

Evaluation Criterion	Number of Nearest Neighbors Considered				
	k=8	k=9	k=10	k=11	k=12
Ranking Loss \downarrow	0.220 \pm 0.006	0.219 \pm 0.006	0.217 \pm 0.006	0.216\pm0.006	0.216\pm0.006
Average Precision \uparrow	0.375 \pm 0.009	0.376 \pm 0.009	0.377 \pm 0.009	0.378 \pm 0.009	0.380\pm0.009
One-error \downarrow	0.695 \pm 0.012	0.694 \pm 0.013	0.693 \pm 0.014	0.691 \pm 0.013	0.688\pm0.014
Coverage \downarrow	15.1 \pm 0.4	15.0 \pm 0.4	14.9 \pm 0.3	14.9 \pm 0.4	14.8\pm0.4
Hamming Loss ($\times 10^{-2}$) \downarrow	5.272 \pm 0.060	5.272 \pm 0.060	5.270\pm0.062	5.271 \pm 0.062	5.271 \pm 0.063



(a) Ranking Loss.



(b) Hamming Loss.

Fig. 12: Performances of TRAM with different percentage of dimensions in MDDM step on automatic image annotation task.

table

TABLE VII: Dimensions of the subspace in MDDM step.

Data Set	Average # Dim	Original # Attributes	Percentage (%)
annotation	43	500	8.60
yeast	14	103	13.59
RCV1-v2(Set A)	54	4,238	1.27
RCV1-v2(Set B)	64	9,479	0.68
scene	6	294	2.04

Besides the number of nearest neighbor, another parameter is about the number of dimensions in the subspace used by MDDM. Note that due to the curse of the dimensionality, the similarities directly calculated based on distances between instances in the input space may be unreliable, especially when these similarities are the key parameters for the TRAM model. A simple, but often very effective, way of dealing with high-dimensional data is to reduce the number of dimensions, by finding a subspace from the input features that is most relevant to label information. Therefore, we need to utilize MDDM before the graph construction among instances. In order to verify this assumption, the results under different percentage of dimensions in the pre-process stage are reported in Figure 12. The experiment is based on automatic image annotation task, and results on other tasks are similar to the case in this task.

Figure 12 shows that on automatic image annotation task, the *Ranking Loss* and *Hamming Loss* of TRAM are significantly improved by introducing the dimensionality reduction (MDDM) before constructing the instance graph. TRAM’s best performance are more likely to appear at the relatively low percentage of dimensions. Nonetheless, the number of dimensions does not have to be pre-specified, which can automatically be determined by setting MDDM’s threshold parameter *thr* as preserving 99.99% of the eigenvalues. Table VII shows the number of dimensions automatically preserved in each task. And it can be seen that the percentage of the preserved dimensions on automatic image annotation task (8.60%) is among the percentages where TRAM shows the best performance in Figure 12. Therefore, in this paper, we set the MDDM’s *thr* as 99.99% to automatic determine dimensions before the instance graph construction.

H. The Influence of Unlabeled Data

Since TRAM proposed in this paper is a transductive learning algorithm, it is also necessary to further analyze the influence of unlabeled data on the performance. In order to separate the influence of labeled data, in this experiment, we fix the size of labeled training set, and increase the size of test set to analyze the influence of unlabeled data. The experiment is based on automatic image annotation task.

We randomly partition the dataset into labeled and unlabeled data according to certain ratios. In detail, we randomly draw 25% of the data as labeled training examples and use the remaining 75% of the data to generate testing examples under different *test rates* from 10% to 90%. For instance, assuming the data set contains 4,800 examples and the test rate is 80%, we randomly

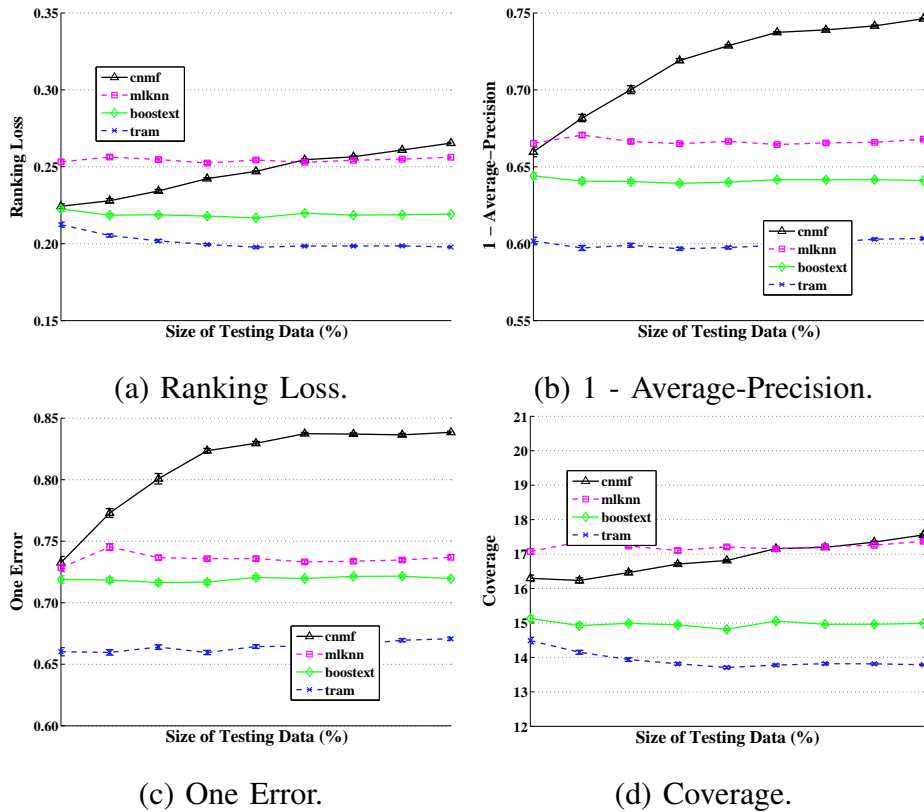


Fig. 13: Results with increasing size of test data on automatic image annotation task. Along the curves, we also plot the 95% confidence intervals of t-test. Note that ML-KNN, BoosTExTER are supervised learning algorithms and only use the training data set with fixed size.

figure

draw 1,200 examples as training examples; and 3, 600 \times 80% = 2880 examples from the remaining data set as test examples. Thirty runs of experiments are conducted under every test rate; in each run, algorithms are evaluated on random data set partitions and the average performance is recorded.

Figure 13 shows that TRAM can explore the unlabeled data and obtain much better results on all evaluation criteria. The performances are very stable with increasing size of unlabeled data. Moreover, the *ranking loss* and *coverage* of TRAM are significantly improved before becoming stable as the size unlabeled data increases. This is because that in transductive learning, TRAM can benefit from the unlabeled data, and this benefit is expected to be saturated when the unlabeled data are more than enough for building an accurate model. Since ML-KNN and BoosTExTER are both supervised learning algorithms, their performances cannot be improved by the size of the

test data. Results on other tasks are also similar to the case shown in this task.

VII. CONCLUSION

In this paper, we propose a new transductive multi-label learning algorithm by alpha matting. In detail, the proposed method TRAM is first formulated as an optimization problem for transductive multi-label learning, which is able to exploit unlabeled data to obtain an effective model for assigning appropriate multiple labels to instances. Then we develop an efficient algorithm which has a closed form solution to solve this optimization problem. Empirical studies on many real-world multi-label learning tasks show that TRAM can effectively make use of unlabeled data information to achieve performance as good as existing state-of-the-art multi-label learning algorithms, moreover it is much faster and can handle larger data sets.

APPENDIX

In this part, we study the properties of the linear systems solutions for Eq.9 and Eq.10. For convenience of study, we combine the Eq.9 with the constrains for labeled data as:

$$A_{\mathcal{U}\mathcal{U}}\alpha_{\mathcal{U}}^j + A_{\mathcal{U}\mathcal{L}}\alpha_{\mathcal{L}}^j = \mathbf{0} \quad (13)$$

$$\alpha_{\mathcal{L}}^j = \bar{\alpha}_{\mathcal{L}}^j \quad (14)$$

which is equivalent to:

$$\tilde{A}\alpha^j = \mathbf{b}^j, \quad j = 1, \dots, m \quad (15)$$

where

$$\tilde{A} = \begin{bmatrix} A_{\mathcal{U}\mathcal{U}} & A_{\mathcal{U}\mathcal{L}} \\ \mathbf{0} & I \end{bmatrix} \text{ and } \mathbf{b}^j = \begin{bmatrix} \mathbf{0} \\ \bar{\alpha}_{\mathcal{L}}^j \end{bmatrix}$$

Then we show that the solution of $\tilde{A}\alpha^j = \mathbf{b}^j$ automatically satisfies the bilateral constrains $\mathbf{0} \leq \alpha^j \leq \mathbf{1}$.

Suppose \mathcal{U} is a discrete connected domain and \mathcal{L} is its boundary ($\mathcal{U} \cap \mathcal{L} = \emptyset$). The connectedness of \mathcal{U} is defined on instances' neighbor \mathcal{N}_i , *i.e.* instance i and z are connected if and only if they are a neighbor of each other, and W_{iz} and W_{zi} are both positive. Let $\alpha = (\alpha_i)$ be a discrete function defined on $\mathcal{U} \cup \mathcal{L}$, then the (*strong*) *discrete maximum principle* says that α can only attain its maximum in \mathcal{L} , unless α is constant in $\mathcal{U} \cup \mathcal{L}$. It is similar for the minimum principle. If there are more than one connected components in \mathcal{U} , we can apply the principle

to each component independently. We also assume that each point in \mathcal{L} is a neighbor of some instance in \mathcal{U} .

THEOREM 1: The solution to $\tilde{A}\alpha = \mathbf{b}$ satisfies the *discrete maximum principle*.

Proof: Suppose that the maximum of α can be attained at an interior point $i_0 \in \mathcal{U}$. Then the i_0 -th equation of Eq.15 is $(\tilde{A}\alpha)_{i_0} = 0$ since $b_{i_0} = 0$. Notice that the i_0 -th row of \tilde{A} is the same as the i_0 -th row of $A = I - W$. Therefore,

$$(\tilde{A}\alpha)_{i_0} = \alpha_{i_0} - \sum_{z \in \mathcal{N}_{i_0}} W_{i_0 z} \alpha_z = 0$$

or

$$\alpha_{i_0} = \sum_{z \in \mathcal{N}_{i_0}} W_{i_0 z} \alpha_z$$

Note that $W_{i_0 z} > 0$ for $z \in \mathcal{N}_{i_0}$ and $\sum_{z \in \mathcal{N}_{i_0}} W_{i_0 z} = 1$, which means the maximum value α_{i_0} equals a weighted average of $\{\alpha_z : z \in \mathcal{N}_{i_0}\}$, thus for all $z \in \mathcal{N}_{i_0}$, α_z is also the maximum. Similarly, since the domain \mathcal{U} is connected, we can conclude that the values of α in \mathcal{U} and the neighbor of \mathcal{U} which covers \mathcal{L} are all maximum. This shows that if α has an interior maximum, then α is constant in $\mathcal{U} \cup \mathcal{L}$. ■

COROLLARY 1: The solution to $\tilde{A}\alpha = \mathbf{b}$ satisfies the the bilateral constraints $0 \leq \alpha \leq 1$, if $\{\alpha_i = 0 : i \in \mathcal{L}\}$ and $\{\alpha_i = 1 : i \in \mathcal{L}\}$ are non-empty sets.

Proof: According to maximum principle, $\alpha_z \leq \max_{i \in \mathcal{L}} \alpha_i = 1$ for all $z \in \mathcal{U}$. Similarly, we have $\alpha \geq \min_{i \in \mathcal{L}} \alpha_i = 0$. Therefore, $0 \leq \alpha_z \leq 1$ for all $z \in \mathcal{U}$. ■

ACKNOWLEDGMENT

The authors wish to express their gratitude to the anonymous reviewers for their valuable comments and suggestions. The authors want to thank Yu-Feng Li, Jieping Ye, Yang Yu, De-Chuan Zhan and Yin Zhang for reading a draft of the paper. This work was supported by the National Science Foundation of China (60635030, 60721002), the Jiangsu Science Foundation (BK2008018), the Jiangsu 333 High-Level Talent Cultivation Program, the National High Technology Research and Development Program of China (2007AA01Z169), the National Fundamental Research Program of China (2010CB327900), Hong Kong Baptist University Faculty Research Grants and Hong Kong Research Grant Council Grant Number 201508.

REFERENCES

- [1] N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the 15th International Conference on Machine Learning*, pages 1–9, Madison, WI, 1998.
- [2] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: a geometric framework for learning from examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.
- [3] N. Biggs. *Algebraic Graph Theory*. Cambridge University Press, Cambridge, UK, 1974.
- [4] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning*, pages 19–26, Williamstown, MA, 2001.
- [5] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.
- [6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, 2004.
- [7] G. Carneiro, A. Chan, P. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):394–410, 2007.
- [8] O. Chapelle, B. Scholkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
- [9] Y. Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–271, Kauai, HI, 2001.
- [10] F. D. Comité, R. Gilleron, and M. Tommasi. Learning multi-label alternating decision tree from texts and data. In *Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 35–49, Leipzig, Germany, 2003.
- [11] K. Crammer and Y. Singer. A new family of online algorithms for category ranking. In *Proceedings of the 25th International Conference on Research and Development in Information Retrieval*, pages 151–158, Tampere, Finland, 2002.
- [12] P. Duygulu, K. Barnard, N. de Freitas, and D. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the 7th European Conference on Computer Vision*, pages 97–112, Copenhagen, Denmark, 2002.
- [13] 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. Cambridge, MA: MIT Press, 2002.
- [14] 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.
- [15] I. Vlahavas G. Tsoumakas. Random k-labelsets: An ensemble method for multilabel classification. In *Proceedings of the 18th European Conference on Machine Learning*, pages 406–417, Warsaw, Poland, 2007.
- [16] 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 21th International Conference on Machine Learning*, pages 329–336, Banff, Canada, 2004.
- [17] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th International Conference on Information and Knowledge Management*, pages 195–200, Bremen, Germany, 2005.
- [18] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 22–30, Sydney, Australia, 2004.
- [19] L. Grady. Random walks for interactive alpha-matting. In *Proceedings of the 5th International Conference on Visualization, Imaging and Image Processing*, pages 423–429, Benidorm, Spain, 2005.

- [20] M. Opper, H. Seung, and H. Sompolinsky. Query by committee. In *Proceedings of the 5th ACM Workshop on Computational Learning Theory*, pages 287–294, Pittsburgh, PA, 1992.
- [21] W. Hackbusch. Iterative solution of large sparse systems of equations. *Mathematics of Computation*, 64(212):1759–1761, 1995.
- [22] T. Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pages 200–209, Bled, Slovenia, 1999.
- [23] T. Joachims. Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 290–297, Washington, DC, 2003.
- [24] J. Sun, J. Jia, C. K. Tang, and H. Y. Shum. Poisson matting. In *Proceedings of the 31st Annual Conference on Computer Graphics and Interactive Techniques*, pages 315–321, Los Angeles, CA, 2004.
- [25] 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. Cambridge, MA: MIT Press, 2005.
- [26] S.-W. Ji, L. Sun, and J.-P. Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 668–676, Las Vegas, Nevada, 2008.
- [27] A. Levin, D. Lischinsky, and Y. Weiss. A closed form solution to natural image matting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 228–242, New York, NY, 2006.
- [28] A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, 2007.
- [29] D. D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.
- [30] 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.
- [31] P. Matstoms. Sparse QR factorization in MATLAB. *ACM Transactions on Mathematical Software*, 20(1):136 – 159, 1994.
- [32] 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.
- [33] M. Ng, G. Qiu, and A. Yip. A study of interactive multiple class image segmentation problems. Technical Report 07-51, UCLA CAM, 2007.
- [34] P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy. Combining microarray expression data and phylogenetic profiles to learn functional categories using support vector machines. In *Proceedings of the 5th International Conference on Computational Biology*, pages 242–248, Montréal, Canada, 2001.
- [35] 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 International Conference on Multimedia*, pages 17–26, Augsburg, Germany, 2007.
- [36] R. Rak, L. Kurgan, and M. Reformat. Multi-label associative classification of medical documents from MEDLINE. In *Proceedings of the 4th International Conference on Machine Learning and Applications*, pages 177–186, Los Angeles, CA, 2005.
- [37] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 18–25, Hilton Head, SC, 2000.
- [38] R. E. Schapire and Y. Singer. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168, 2000.

- [39] F. A. Thabtah, P. I. Cowling, and Y. Peng. MMAC: A new multi-class, multi-label associative classification approach. In *Proceedings of the 4th International Conference on Data Mining*, pages 217–224, Brighton, UK, 2004.
- [40] N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 721–728. Cambridge, MA: MIT Press, 2003.
- [41] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.
- [42] J. Wang and M. Cohen. An iterative optimization approach for unified image segmentation and matting. In *Proceedings of the 10th IEEE International Conference on Computer Vision*, pages 936–943, Beijing, China, 2005.
- [43] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similaritysearch methods in high-dimensional spaces. In *Proceedings of 24th International Conference on Very Large Data Bases*, pages 194–205, New York, NY, 1998.
- [44] K. Yu, S. Yu, and V. Tresp. Multi-label informed latent semantic indexing. In *Proceedings of the 28th International Conference on Research and Development in Information Retrieval*, pages 258–265, Salvador, Brazil, 2005.
- [45] 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):1479–1493, 2006.
- [46] M.-L. Zhang and Z.-H. Zhou. MI-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.
- [47] Y. Zhang and Z.-H. Zhou. Multi-label dimensionality reduction via dependency maximization. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1053–1055, Chicago, IL, 2008.
- [48] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 321–328. Cambridge, MA: MIT Press, 2003.
- [49] Z.-H. Zhou and M. Li. Semi-supervised regression with co-training style algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 19(11):1338–1351, 2007.
- [50] Z.-H. Zhou and M.-L. Zhang. Multi-instance multi-label learning with application to scene classification. In Y. Weiss, B. Scholkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1609–1616. Cambridge, MA: MIT Press, 2006.
- [51] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li. MIML: A framework for learning with ambiguous objects. In *CORR abs/0808.3231*, 2008.
- [52] S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labelled classification using maximum entropy method. In *Proceedings of the 28th International Conference on Research and Development in Information Retrieval*, pages 274–281, Salvador, Brazil, 2005.
- [53] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, 2007.