<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Multi-label metric transfer learning jointly considering instance space and label space distribution divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Jiang, Siyu; Xu, Yonghui; Wang, Tengyun; Yang, Haizhi; Qiu, Shaojian; Yu, Han; Song, Hengjie</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Jiang, S., Xu, Y., Wang, T., Yang, H., Qiu, S., Yu, H., &amp; Song, H. (2019). Multi-label metric transfer learning jointly considering instance space and label space distribution divergence. IEEE Access, 7, 10362-10373. doi:10.1109/ACCESS.2018.2889572</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2019</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10220/48323">http://hdl.handle.net/10220/48323</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>© 2019 IEEE (Open Access). Translations and content mining are permitted for academic research only. Personal use is also permitted, but republishing/redistribution requires IEEE permission. See <a href="http://www.ieee.org/publications_standards/publications/rights/index.html">http://www.ieee.org/publications_standards/publications/rights/index.html</a> for more information.</td>
</tr>
</tbody>
</table>
Multi-Label Metric Transfer Learning Jointly Considering Instance Space and Label Space Distribution Divergence

SIYU JIANG1, YONGHUI XU1,2, TENGYUN WANG1, HAIZHI YANG1, SHAOJIAN QIU3, HAN YU2, AND HENJIE SONG1

1School of Software Engineering, South China University of Technology, Guangzhou 510006, China
2School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798
3School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
Corresponding author: Hengjie Song (sehjsong@scut.edu.cn)

ABSTRACT Multi-label learning deals with problems in which each instance is associated with a set of labels. Most multi-label learning algorithms ignore the potential distribution differences between the training domain and the test domain in the instance space and label space, as well as the intrinsic geometric information of the label space. These restrictive assumptions limit the ability of the existing multi-label learning algorithms to classify between domains. To solve this problem, in this paper, we propose a novel distribution-adaptation-based method, the multi-label metric transfer learning (MLMTL), to relax these two assumptions and handle more general multi-label learning tasks effectively. In particular, MLMTL extends the maximum mean discrepancy method into multi-label classification by learning and adjusting the weights for the multi-labeled training instances. In this way, MLMTL bridges the instance distribution and label distribution divergence between training and test datasets. In addition, based on the balanced multi-label training data, we explore the intrinsic geometric information of the label space by encoding it into a distance metric learning framework. Extensive experiments on five benchmark datasets show that the proposed approach significantly outperforms the state-of-the-art multi-label learning algorithms.

INDEX TERMS Transfer learning, metric learning, multi-label learning.

I. INTRODUCTION

Multi-label learning aims to learn a prediction model that can assign a set of relevant labels for unseen objects [1]. Formally, \( \mathcal{X} = \mathbb{R}^d \) denotes the \( d \)-dimensional feature space, and \( \mathcal{Y} = \{ l_1, l_2, \ldots, l_k \} \) denotes \( k \) possible class labels. The objective of multi-label classification is to learn a predictive function \( h: \mathcal{X} \rightarrow 2^\mathcal{Y} \) that assigns each instance \( x \in \mathcal{X} \) with a set of proper labels \( h(x) \subseteq \mathcal{Y} \).

Multi-label learning has been applied to a wide range of applications, such as text categorization [2, 3], image classification [4, 5], video processing [6, 7] and bioinformations annotation [8, 9]. For instance, in the text analysis domain, a single news document may cover several topics (Obamacare, politics, healthcare, and so on). In an image classification task, an image can be labeled with various tags (e.g., an image taken from beach may include beach, mountain, field, and sunset).

To leverage the advantages of multi-label representation, many multi-label learning algorithms have been proposed. Current studies on multi-label learning are based on statistical information derived from the label sets of an unseen object’s neighboring objects [10]. However, considering the potential association between paired labels, dual set multi-label learning (DSML) [11] exploits pairwise inter-set label relationships for assisting multi-label learning. Furthermore, multi-label learning with label specific features (LIFT) [12] suggests a promising direction by using label-specific features for multi-label learning, and other known algorithms [13]–[15].

Although many applications have successfully integrated multi-label learning approaches, existing approaches often assume that the instances and labels of training and test data are drawn from the same distribution. However, in many multi-label learning tasks, the label-distribution and instance-distribution may change [16]–[18]. For example, in the multi-label image classification task, as shown in Fig. 1, the scene dataset contains pictures of seaside scenes and includes multiple labels, e.g., beach, tree, mountain and water. The desert...
dataset contains pictures of desert and includes multiple labels, e.g., desert, mountain, sunset and tree. The forest dataset contains pictures of forest and includes multiple labels, e.g., mountain, tree, and water. From the figure, it can be observed that beach and water appear frequently in the seaside scenes. Desert appears more often in the desert scenes. Tree and mountain appear more often in the forest scenes. These observations show that the distributions of objects (labels) in different scenes are different: \( P_{\text{training}}(\mathbf{Y}) \neq P_{\text{test}}(\mathbf{Y}) \). This is referred to as label-level distribution divergence. From the figure, we also note that blue areas are more likely to appear in beach scenes. Yellow areas are more likely to appear in desert scenes. Green areas are more likely to appear in forest scenes. These observations indicate significant differences in the distribution of instances from different scenarios. This is called instance-level distribution divergence. When instance-level distribution divergence and label-level distribution (i.e., the intrinsic geometric information of the label space) divergence occur simultaneously, existing transfer learning methods or multi-label classification methods do not perform well.

To address this problem we propose a novel approach, multi-label metric transfer learning (MLMTL), to transfer knowledge from multi-label instances in the training set to predict multi-labels of the instances in test sets where their distributions are different. The main contributions of this work are two-fold:

- Unlike most traditional multi-label learning approaches which assume that the training and the test data are derived from the same instance-level distribution and label-level distribution [1], we develop a MLMTL model which can effectively deal with differences in these distributions. Specifically, considering the difference in instance-level, we employ instance weights to match the distributions between the training and the test data.
- Unlike traditional multi-label transfer learning methods, which ignore differences in label class distributions between the training and test datasets, we exploit label-level data by considering different weights for the specific instances. After that, considering the advantages of preserving the intrinsic geometric information of the label space, MLMTL explores and encodes it into a distance metric learning framework by utilizing the desirable traits of the Mahalanobis distance (i.e., discovering label dependency such that an instance with different multiple labels will be moved further away.) [19]. In this way, the learned MLMTL model is more realistic and improves the accuracy of the test tasks under a more appropriate distance metric.

We organize the rest of this paper as follows. In Section II, we review related work. In Section III, we describe the proposed MLMTL approach in detail. In Section IV, we provide our experiment settings, design, and results to evaluate the performance of the proposed method. Section V concludes the paper and provides an outlook for future work.

## II. RELATED WORK

Previous studies related to our work can be classified into three parts: traditional multi-label learning, multi-label transfer learning, and multi-label metric learning.

### A. TRADITIONAL MULTI-LABEL LEARNING

Traditional multi-label learning divides multi-label classification methods into three major types [1] by the order of label correlations: first-order approaches, second-order approaches, and high-order approaches.

First-order approaches deal with multi-label classification problems by decomposing them into a number of independent binary classification problems. Boutell et al. [4] designed a framework that trains an independent classifier for each label to handle semantic scene classification. To make better use of the information in multi-label case, Zhang and Zhou [10] proposed an adaptation of the k-NN method, Multi-label k-Nearest Neighbors (ML-KNN), to perform multi-label learning tasks. ML-KNN finds the k nearest neighbors of an instance and applies the maximum likelihood approach to predict the labels. This method analyzes the multi-label k nearest neighbor. However, it ignores the co-existence of other labels.

Second-order approaches handle multi-label classification problems by considering pairwise relations between labels. Zhang and Zhou [20] put forward a neural network algorithm named Backpropagation for Multi-Label Learning (BP-MLL), which is derived from the backpropagation method by employing an error function to capture the characteristics of multi-label learning. Brinker [21] converted the multi-label learning problem into a pairwise label ranking task. The key idea is to separate the relevant labels from the irrelevant labels.

High-order approaches tackle multi-label classification problems by analyzing high-order relations between labels. Zhang et al. [22] proposed a multi-label learning approach with feature-induced labeling information enrichment (MLFE), which enriches the labeling information of multi-label examples by leveraging the structural information in the feature space. However, these traditional multi-label learning methods do not take into account the differences in the distributions of features and labels.

### B. MULTI-LABEL TRANSFER LEARNING

Transfer learning mainly focuses on how to learn a model using labeled data from a divergent distribution [16], [23]. Pan et al. [24] proposed an algorithm to learn a latent space for the training dataset and test dataset, which attempts to minimize the distributions between domains in a kernel Hilbert space. Long et al. [25] proposed a novel framework for deep adaptation networks, where deep convolutional
neural network tires to be extended to domain adaptation. Liu et al. [26] developed a shared fuzzy equivalence relations (SFER) method by integrating 1-dimensional fuzzy geometry and fuzzy equivalence relations for dealing with the heterogeneous unsupervised domain adaptation problems.

In term of multi-label transfer learning, Han et al. [27] proposed an algorithm to learn a new representation space from a related dataset, and projects the test data into the learned multi-label encoded sparse linear embedding space to achieve better performance. To handle the changes in the distribution of labels, Hou et al. [13] proposed a method called multi-label manifold learning, which reconstructs the label manifold automatically from the logical multi-label data. For all approaches, the common modeling strategy is to treat each instance in a crisp manner (i.e., being drawn from the same distribution). However, these method only consider the differences in the instance distributions, while label distribution divergence between training and test datasets is ignored.

C. MULTI-LABEL METRIC LEARNING

Our paper uses metric learning to mine potential geometric information between the data. Therefore, the most related works to this paper are multi-label metric learning methods. Gouk et al. [28] proposed a distance metric learning algorithm that maps instances into a feature space where the squared Euclidean distance provides an estimate of the Jaccard distance between the corresponding labels. Jin et al. [29] proposed an iterative method for multi-instance multi-label data by learning a metric from the estimation of instance-label association and updating it iteratively to improve the learning of distance metrics. Reyes et al. [30] exploited the similarity metric as a heuristic to estimate the feature weights in order to improve the performance of lazy multi-label ranking algorithms. Liu and Tsang [31] applied a large margin metric learning together with k nearest neighbors constraints to solve the projection matrix, which reduces the number of constraints. Megano et al. [32] used distance metric learning and evolutionary multi-objective optimization to evaluate inter- and intra- clusters by employing a cluster validity measure with a neighbor relation. Xu et al. [19] constructed a multi-instance multi-label distance metric learning framework (MIMLMDML) that applies Mahalanobis distance to preserve and utilize the intrinsic geometric information.

In summary, exiting approaches share the common assumption that the training and test data are from the same distribution, although discrepancies between training and test distributions are common situation. The next section proposes a novel approach called MLMTL. In contrast, to the best of our knowledge, we handle instance-level and label-level distribution differences for transfer learning in multi-label learning simultaneously, and this is currently not considered in other work. Moreover, MLMTL attempts to learn a distance metric by exploring the intrinsic geometric information in the label space.

III. THE PROPOSED MLMTL FRAMEWORK

A. PRELIMINARIES

Let $D_{SD} = \{(x_i, Y_i)|1 \leq i \leq m\}$ denote the multi-label source domain with m multi-label instances. Here, $x_i \in \mathcal{X}$ is an input feature vector with d-dimensions and associated with $k$ different labels $Y_i = \{l_1, \ldots, k\} \in \{0, 1\}^k$. The target multi-label domain with n multi-label instances is denoted as $D_{TD} = D_{TD}^T \cup D_{TD}^C = \{(x_i, Y_i)|x_{m+1}, \ldots, x_{m+n}^C\} \cup \{(x_{m+i}, c), \ldots, x_{m+n}\}$, and partially labeled data in the target dataset, where $n^C \ll m$. The training dataset contains the labeled instances of the source domain and the target domain, and the test dataset contains the unlabeled instances of the target domain. Formally, our objective is to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ from the training data $D_{SD}$ to predict the labels for the instances belonging to $D_{TD}$. To address the differences in the distributions between $D_{SD}$ and $D_{TD}$, we design the MLMTL approach which consists of three steps. In the first step, we apply kernel mean matching (KMM) [33] to obtain the instance weight $w_i$ for each $x_i$ in $D_{TD}$, which can then be used to adjust instance-level distributions. Meanwhile, we exploit label-level instances by considering different weights for the specific data. In the second step, we provide a new principle to learn a Mahalanobis distance metric. In the third step, we encode the structural information of the label space and propose a method for multi-label classification. Table 1 presents important notations used in this paper.

In the above problem setting, we focus on learning a Mahalanobis distance matrix for the multi-label test dataset. Given a positive semi-definite matrix $M \in \mathbb{R}^{d \times d}$, we denote the Mahalanobis distance metric between a pair of instances $x_i$ and $x_j$ as:

$$d_{ij} = \sqrt{(x_i - x_j)^T M (x_i - x_j)}.$$  \hspace{1cm} (1)

| TABLE 1. Summary of frequently used notations. |
|---------|---------------------------------|
| Symbols | Definitions                     |
| $D_{SD}$ | The source domain.              |
| $D_{TD}$ | The target domain.              |
| $D_{TD}^C$ | The few multi-label instances with labeled in target domain. |
| $D_{TD}^U$ | The multi-label instances with unlabeled in target domain. |
| $x_i$, $x_j$ | The i-th/j-th instance. |
| $m, n$ | Number of source/target domain instances. |
| $d$ | Number of feature space dimension. |
| $\text{MMD}(\cdot, \cdot)$ | Maximum Mean Discrepancy between distributions |
| $D(x_i, x_j)$ | The square of the Mahalanobis distance between instance $x_i$ and $x_j$. |
| $\omega$ | The weight vector of instances. |
| $M_{tr}, M_{te}$ | The marginal distribution of training/test domain. |
| $A$ | The learned Mahalanobis distance Metric. |
| $\delta$ | A margin factor to tune the margin between $x_i$ and $x_j$. |
| $\psi(Y_i, Y_j)$ | A margin between i-th instance and j-th instance. |
| $\ell_{ij}$ | A jaccard distance between $x_i$ and $x_j$. |
| $\zeta, \xi$ | Two slack vector to improve the robustness of the algorithm. |
Since \( M \) is a positive semi-definite matrix, we can decompose \( M \) in the following form: \( M = A^T A \), where \( A \in \mathbb{R}^{d \times d} \). Therefore, learning a Mahalanobis distance matrix is equivalent to learning a matrix \( A \).

Note that because the distributions of the training dataset and the test dataset are different, the labeled data in the training dataset cannot be used to learn a precise distance metric for the test dataset directly. The reweighted training dataset is of the lowest distribution discrepancy (MMD) with the test dataset. Thus, we assume that the labeled data in the training dataset after reweighting are still able to serve as discriminative information for the test dataset, and thus are useful for learning a metric for the test dataset [34].

### B. ADJUSTING INSTANCE-LEVEL AND LABEL-LEVEL DISTRIBUTIONS

In general, the labeled instances cannot be directly applied to learning the distance metric for the testing instances because the distributions of training dataset and test dataset are different: \( \Pr_{tr}(Y) \neq \Pr_{te}(Y) \). For adjusting instance-level distributions, we propose a KMM [33]-based method to obtain the weight \( \omega_i \) for each instance \( x_i \) in the training dataset to bridge the training and the test datasets.

The KMM algorithm introduces the maximum mean discrepancy (MMD), a nonparametric method that directly infers the weights for a given data source without requiring a distribution estimation [33]. This algorithm takes into account both the source and target data to build a transferring bridge. In this paper, KMM is used to measure the similarity between two distributions. The main purpose of KMM is to calculate appropriate weights for each instance from the source domain to minimize the MMD in a reproducing-kernel Hilbert space (RKHS).

Using the matching operation for the distributions as the example, after space mapping and instance weighting, we calculate the MMD for the distributions of the source and target domains as:

\[
\text{MMD}(M_{tr}, M_{te}) = \left\| \frac{1}{m} \sum_{i=1}^{m} \omega_i \phi(x_i) - \frac{1}{n} \sum_{j=m+1}^{m+n} \phi(x_j) \right\|_H^2,
\]

where \( \| \cdot \|_H^2 \) is the RKHS norm and \( \phi(\cdot) \) represents a feature mapping on the RKHS. \( \omega_i \) for all \( i \) make up a weight vector that we want to learn for the training data. In this way, MLMTL reduces the distance of the instances distributions.

Most transfer leaning algorithms assume that the source data have balanced class distributions and the same misclassification cost. Therefore, when complex imbalanced datasets appear, these algorithms cannot correctly represent the distribution characteristics of the data, resulting in reduced accuracies after transfer. Fig. 1 shows that significant differences..
can be observed in the label-level distributions. Unlike traditional transfer learning algorithms, we not only consider the differences between sample distributions, but also consider the label-level distribution differences. In our method, we give a weight to each class (i.e., we fix the weight of negative classes as 1, and set the weight of positive classes as $\beta$). With a weight $\beta$, it means we give more importance to the loss of the misclassified positive instances. In this case, the recall rate of the classification method can be improved even with a sparsely labeled dataset. Section IV discusses the effectiveness of $\beta$.

**C. ENCODING STRUCTURAL INFORMATION OF THE LABEL SPACE**

In the previous section, we have learned instance weight $w_i$ for each $x_i$ in $D_{TD}$. To learn an appropriate Mahalanobis distance, we employ a principle to construct an objective function for multi-label distance metric transfer learning. Specifically, the principle is the larger the label vector distance between the instances is, the larger the margin between the instances should be. According to this principle, after combining the margins between the instances with distance $D(x_i, x_j)$ of instances larger than a margin $\psi(Y_i, Y_j)$, we note that

$$D(x_i, x_j) \geq \psi(Y_i, Y_j),$$

where $\psi(Y_i, Y_j)$ is the margin between $i$-th instance and $j$-th instance. Note that the margin $\psi(Y_i, Y_j)$ should increase significantly with increasing (squared) distance $D(x_i, x_j)$. Hence, we define $\psi(Y_i, Y_j)$ as follows:

$$\psi(Y_i, Y_j) = \sqrt{\exp(\delta \ell_{ij}) - 1},$$

where $\ell_{ij}$ denotes the Jaccard distance [1] between $x_i$ and $x_j$. Considering that the case where both variables are 1 (positive match) is more meaningful than the case where both are 0 (negative match), the Jaccard distance is chosen to measure the distance between $x_i$ and $x_j$ because it measures the comparison between the asymmetric information and the vector components of the binary variable. In addition, since we adjust the margin between $x_i$ and $x_j$, we create a margin factor $\delta > 0$. Therefore, considering that the larger margin between $x_i$ and $x_j$ is more advantageous for our proposed algorithm, MLMTL chooses to employ a larger $\delta$ to gain larger margins. Hence, we set $\delta=1$ for MLMTL on all datasets. In this way, we can encode much of the structural information of the feature space.

**D. MULTI-LABEL METRIC TRANSFER LEARNING**

In some situations, in addition to obtaining instances from the source domain, we can also obtain a few labeled instances from the target domain. Hence, based on the above analysis of multi-label learning and differences in distributions, we can use the labeled instances obtained from the source and the target domains to learn a distance metric $A$ and propose a method for multi-label classification:

$$\arg\min (A, \zeta, \xi)$$

$$= tr(A) - \log(det(A)) + \lambda \left( \sum_{Y_i \neq Y_j} \omega_{ij} \zeta_{ij} + \sum_{Y_i = Y_j} \omega_{ij} \xi_{ij} \right)$$

$$\text{s.t.} \begin{cases} tr(v_i^T A^T A v_j) \geq \exp(\delta \ell_{ij}) - \zeta_{ij}, (Y_i \neq Y_j, 1 \leq i, j \leq n) \\ tr(v_i^T A^T A v_j) \leq 1 + \xi_{ij}, (Y_i = Y_j, 1 \leq i, j \leq n) \\ \zeta_{ij} \geq 0, \xi_{ij} \geq 0, \end{cases}$$

where $D_{TD}$ is defined as the set of multi-label instances labeled in the test dataset. We set $v_i = (x_i - x_j)$ and $\omega_{ij} = \omega_i \times \omega_j$. For handling noise in the datasets in order to improve the robustness of our algorithm, we set two slack vectors $\zeta \in \mathbb{R}^d$ and $\xi \in \mathbb{R}^d$.

Since we have reweighted the instances, (5) can generate an optimal solution for the multi-label learning task.

**E. OPTIMIZATION**

In this section, we show the approach to solving the optimization problem constructed in (5). Firstly, by adding penalty functions, we convert the constrained problem into an unconstrained problem. The resulting optimization problem becomes,

$$g(A, \zeta, \xi) = f(A, \zeta, \xi)$$

$$= \arg\min \left[ \sum_{Y_i \neq Y_j} \max\{\exp(\delta \ell_{ij}) - \zeta_{ij} - d_{ij}, 0\}^2 \right]$$

$$\text{s.t.} \begin{cases} \sum_{Y_i = Y_j} \max(0 - \zeta_{ij})^2 \\ \sum_{Y_i \neq Y_j} \max(0 - \xi_{ij})^2 \end{cases}$$

In this equation, $d_{ij}$ is denoted as $tr(v_i^T A^T A v_j)$ and $\delta$ is the penalty coefficient.

Using the gradient-projection function [35], we initialize $A_0, \zeta_0, \xi_0$, and $A$ through gradient descent. We compute $\xi$ and $\zeta$ based on the following rules:

$$A^{t+1} = A^t - \gamma_1 \frac{\partial g(A, \zeta, \xi)}{\partial A} |_{A^t},$$

$$\xi_{ij}^{t+1} = \xi_{ij}^t - \gamma_2 \frac{\partial g(A, \zeta, \xi)}{\partial \xi_{ij}} |_{\xi_{ij}^t},$$

$$\zeta_{ij}^{t+1} = \zeta_{ij}^t - \gamma_2 \frac{\partial g(A, \zeta, \xi)}{\partial \zeta_{ij}} |_{\zeta_{ij}^t}.$$
\[ I - A^{-1} - 2\delta \left\{ \sum_{Y_i \neq Y_j} \max(A(v_i^T v_j^T)^T + A(v_i^T v_j^T), 0) \right\} \]
\[ - \sum_{Y_i = Y_j} \max(A(v_i^T v_j^T)^T + A(v_i^T v_j^T), 0), \]
\[ \partial g(A, \xi, \gamma) \]
\[ \frac{\partial g(A, \xi, \gamma)}{\partial \xi} = w_j - 2\delta \left\{ \sum_{Y_i = Y_j} \max(\delta_{ij} - 1 - \xi_{ij}, 0)v \right\} \]
\[ + \sum_{Y_i \neq Y_j} \max(0 - \xi_{ij}), \]
\[ \frac{\partial g(A, \xi, \gamma)}{\partial \xi} = w_j - 2\delta \left\{ \sum_{Y_i \neq Y_j} \max(\exp(\delta_{ij}) - \xi_{ij} - d_{ij}, 0) \right\} \]
\[ + \sum_{Y_i \neq Y_j} \max(0 - \xi_{ij}). \]

We summarize the proposed MLMTL method in Algorithm 1.

Algorithm 1 MLMTL

Require: training set \( D \), a margin factor \( \delta \), step size's \( \gamma^1, \gamma^2, \gamma^3 \), a penalty coefficient \( \delta \), and a threshold \( \epsilon \).

Ensure:

\( (A, \xi, \gamma) = \arg \min f(A, \xi, \gamma) \)

1: Initialize \( A^0, \xi^0 \), and \( \gamma^0 \)

2: procedure MLMTL, \( \xi, \gamma \)

3: while true do

4: Update \( A \) by \( A^{t+1} = A^t - \gamma t \frac{\partial g(A, \xi, \gamma)}{\partial A} \)

5: Update each \( \xi_{ij} \) by \( \xi_{ij}^{t+1} = \xi_{ij}^t - \gamma t \frac{\partial g(A, \xi, \gamma)}{\partial \xi_{ij}} \)

6: Update each \( \xi_{ij} \) by \( \xi_{ij}^{t+1} = \xi_{ij}^t - \gamma t \frac{\partial g(A, \xi, \gamma)}{\partial \xi_{ij}} \)

7: if \( f(A^{t+1}, \xi^{t+1}, \gamma^{t+1}) - f(A^t, \xi^t, \gamma^t) < \epsilon \) then

8: \( A = A^{t+1}, \xi = \xi^{t+1}, \gamma = \gamma^{t+1} \)

9: Break loop

10: end if

11: end while

12: end procedure

IV. EXPERIMENTAL EVALUATION

In this section, we evaluate the effectiveness of the proposed method using 5 publicly available multi-label datasets.

A. DATASETS

We adopt five benchmark multi-label datasets in our experiments, including scene [4] (image), emotions [36] (music), enron1 (text), medical2 (text), and bibtex [37]. All the datasets are publicly available and cover a broad range of cases with different multi-label properties. Table 2 shows that we use \textit{Num}, \textit{Sub}, \textit{F(S)}, \textit{D}, \textit{L}, \textit{Cardinality}, and \textit{Domain} to represent the number of examples, the number of examples in different sub-dataset, feature type, feature dimensionality, size of label space, the average number of relevant labels, and domain type, respectively. The detailed information of the datasets is as follows:

- Scene: The scene dataset provides 2,407 images tagged with six labels such as sunset, beach, and mountain.
- Emotions: The emotions dataset provides 593 songs categorized into 6 labels evoking emotions such as sad-ly and happy-pleased etc.
- Medical: The medical dataset provides 978 chest radiograph text reports annotated with one or more out of the 45 disease codes.
- Enron: The Enron dataset provides 1,702 email messages categorized into 53 topic views, such as company strategy, humor, and legal advice.
- Bibtex: The bibtext dataset provides 7,395 text annotated with 159 labels such as the title of the paper, the authors, etc.

To evaluate the performance of the algorithm on the datasets with different instance-level distributions and label-level distributions, we adopted the k-means algorithm to divide each dataset into four sub-datasets. Because of the significant differences in the centers of the sample belonging to these sub-datasets, their data distributions are different. We selected one sub-dataset as the source domain in turn, and then treated the other three sub-datasets as target domain for prediction. For example, \textit{Sub1} dataset (source domain) predicts \textit{Sub2} (target domain). In this way, in each dataset, we constructed 12 combinations of training and test datasets. We ran each model 10 times on each of the 12 combinations and used the average outputs as the final results.

B. EVALUATION METRICS

Let \( D_t = \{ (x_i, Y_i) | 1 \leq i \leq n \} \) be the test set and \( h(\cdot) \) be the target multi-label classifier. To consider various aspects of the multi-label predictor’s performance, in this paper, we applied both example-based evaluation metrics [2], [38], [39] (Average-Precision, Average-Recall) and label-based evaluation metrics [40] (Average-F1, Macro-average F1, Micro-average F1). The evaluation metrics are widely adopted to evaluate multi-label learning [1], [41].


1) EXAMPLE-BASED METRICS

Average-Precision evaluates the rank of average fraction of relevant labels ranked lower than particular label \( y \in Y_i \). A larger Average-Precision value indicates better performance:

\[
\text{Average - Precision} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|Y_i|} \sum_{y \in Y_i} \frac{1}{\text{rank}_f(x, y)} \sum_{y' \in Y_i} \text{rank}_f(x, y') \leq \text{rank}_f(x, y) \leq |Y_i|.
\]  

(13)

Average-Recall evaluates the average fraction of correct labels that have been predicted. A larger Average-Recall value indicates better performance:

\[
\text{Average - Recall} = \frac{1}{n} \sum_{i=1}^{n} \frac{|\{y| \text{rank}_f(x, y) \leq |h(x_i)|, y \in Y_i\}|}{|Y_i|}.
\]  

(14)

2) LABEL-BASED METRICS

The Average-F1 value evaluates the average precision and the average recall. A larger Average-F1 value indicates better performance:

\[
\text{Average - F1} = \frac{2 \times \text{avgPrec} \times \text{avgRecall}}{\text{avgPrec} + \text{avgRecall}} \approx \frac{2TP}{2TP + FP + FN}.
\]

Macro-average F1 value evaluates the arithmetic average of F1-measure of all output labels. A larger Macro-average F1 value indicates better performance:

\[
\text{Micro - F1} = \frac{\sum_{i=1}^{k} 2FP_i}{\sum_{i=1}^{k} (2TP_i + FP_i + FN_i)}.
\]

Micro-average F1 value evaluates the weighted average of F1-measure over all \( k \) labels. A larger Macro-average F1 indicates better performance:

\[
\text{Maro - F1} = \frac{1}{k} \sum_{i=1}^{k} \frac{2FP_i}{(2TP_i + FP_i + FN_i)}.
\]

where \( TP_i, FP_i \) and \( FN_i \) indicate the number of true positives, false positives, and false negatives in the \( i \)-th class label, respectively.

C. COMPARISON METHODS

In this paper, we compare the performance of our proposed method with the following multi-label classification approaches:

- ML-KNN [10]: This is a first-order approach, which uses the maximum a posteriori principle to determine the label set for the test instance.
- BP-MLL [20]: This is a second-order approach and also is a neural network style multi-label learning algorithm. It compares the rankings of relevant labels and irrelevant labels to consider pairwise relations between labels.
- MLFE [22]: This is a high-order approach, which conveys the structural information modeled by sparse reconstruction in feature space to facilitate generating enriched labeling information.
- MLL [13]: This approach transfers the topological structure from the feature space to the label space by exploring the manifold in the label space, and it differs from transfer learning [16], which deals with two problem spaces of different fields or distributions.

For the comparison approaches, we select the best reported parameters for these four algorithms. Specifically, for ML-KNN, the number of nearest neighbors is set to 10 [10]. For BP-MLL, the number of hidden units of the network is set to 20%, and the number of training epochs is set to 100 [20]. For MLFE, parameters \( \beta_1, \beta_2, \) and \( \beta_3 \) are selected from \{1, 2, 3, ..., 10\}, \{1, 10, 15\}, and \{1, 10\} respectively [22]. For MLL, the balance parameter is set to 1, and cost parameters are selected from \{1, 2, 3, ..., 10\} [13].

D. SENSITIVITY ANALYSIS

We use parameter \( \beta \) to reweight the importance of the positive class for handling class classification and label-level distribution changing problems. To illustrate the effect of this parameter, in Fig. 2, we show the performances of the proposed algorithm with different \( \beta \) values on five datasets. The figure shows that as \( \beta \) increases, the performance of the five measures on the four datasets (scene, emotions, enron, bibtext) improved slightly, but declined for the other dataset (medical). This is because a larger weight for positive instances may influence the accuracy of the negative instances. This result proves that \( \beta \) has different impacts on different datasets: a larger \( \beta \) is suitable for scene, emotions and enron datasets, and a small \( \beta \) is suitable for medical datasets. Therefore,

---

**TABLE 2. Detailed information of the datasets.**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Num</th>
<th>Sub1</th>
<th>Sub2</th>
<th>Sub3</th>
<th>Sub4</th>
<th>F(S)</th>
<th>D</th>
<th>L</th>
<th>Cardinality</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>scene</td>
<td>2407</td>
<td>729</td>
<td>512</td>
<td>632</td>
<td>534</td>
<td>numeric</td>
<td>294</td>
<td>6</td>
<td>1.074</td>
<td>images</td>
</tr>
<tr>
<td>emotions</td>
<td>593</td>
<td>133</td>
<td>245</td>
<td>139</td>
<td>76</td>
<td>numeric</td>
<td>72</td>
<td>6</td>
<td>1.869</td>
<td>music</td>
</tr>
<tr>
<td>enron</td>
<td>1702</td>
<td>794</td>
<td>71</td>
<td>735</td>
<td>101</td>
<td>nominal</td>
<td>1001</td>
<td>53</td>
<td>3.378</td>
<td>text</td>
</tr>
<tr>
<td>medical</td>
<td>978</td>
<td>243</td>
<td>248</td>
<td>297</td>
<td>190</td>
<td>nominal</td>
<td>1449</td>
<td>45</td>
<td>1.245</td>
<td>text</td>
</tr>
<tr>
<td>bibtex</td>
<td>7395</td>
<td>1370</td>
<td>4977</td>
<td>1047</td>
<td>1</td>
<td>nominal</td>
<td>1836</td>
<td>159</td>
<td>2.402</td>
<td>text</td>
</tr>
</tbody>
</table>
FIGURE 2. The effect of beta $\beta$ to MLMTL on the five datasets based on different evaluation measures.

FIGURE 3. Comparison results with MLMTL and MLMTL-without KMM on five datasets.
FIGURE 4. Influence of each comparing algorithm with increasing percentages of labeled instances in test data.

we set $\beta = 0.3$, which is an intermediate value for MLMTL on all datasets.

E. PERFORMANCE COMPARISON

Firstly, we randomly selected 5% of labeled instances in the test dataset for training, and compared our model MLMTL with the other state-of-art multi-label algorithms. Secondly, we investigated the performances of different algorithms with respect to the various percentages of labeled instances in the test dataset. For each experiment, the performance of evaluation criteria was introduced in Section IV-B, where “↑” indicates “the larger the better.” In these comparison algorithms, we conducted $m$-fold cross-validation on MLMTL and other multi-label methods. Here, $m = 100%/l\%$, where $l \in \{3, 4, 5, 6, 7\}$ is the ratio of labeled data in the test data. For example, if $l = 5$, $m = 100%/10\% = 10$, it means that we conducted 10-fold cross-validation experiments.

1) COMPARISON OF MLMTL AND MLMTL-WITHOUT KMM

Fig. 3 shows the comparison results of MLMTL and MLMTL-without KMM with four evaluation metrics on five real-world datasets. As shown in the diagram, the Average-Precision and Average-Recall, Macro-F1 and Micro-F1 of MLMTL-without KMM on all five datasets is significant lower than that of MLMTL. This is because, MLMTL considers instance-level and label-level distribution divergence simultaneously by utilizing KMM-based method.

2) COMPARISON OF ALGORITHM PERFORMANCE

Table 3 shows the comparison results of MLMTL with four methods on five real-world datasets. It shows that MLMTL algorithm outperforms the other multi-label algorithms in most of the cases. This is because, ML-KNN, BP-MLL, MLL, and MLL are designed for the traditional multi-label problem where the distributions of the training and the test datasets are assumed to be identical. Unlike the other state-of-the-art methods, MLMTL considers instance-level and label-level distribution divergence simultaneously and uses the Mahalanobis distance to explore the intrinsic geometric information in the label space.
### TABLE 3. Predictive performance of each comparing algorithm (mean ± std. deviation).

<table>
<thead>
<tr>
<th>Comparing algorithms</th>
<th>Average-Precision ↑</th>
<th>Average-Recall ↑</th>
<th>Average-F1 ↑</th>
<th>Macro-F1 ↑</th>
<th>Micro-F1 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>scene</td>
<td>emotions</td>
<td>enron</td>
<td>medical</td>
<td>bitbex</td>
</tr>
<tr>
<td>MLMTL</td>
<td>0.823 ± 0.006</td>
<td>0.599 ± 0.014</td>
<td>0.611 ± 0.016</td>
<td>0.457 ± 0.013</td>
<td>0.382 ± 0.003</td>
</tr>
<tr>
<td>ML-KNN</td>
<td>0.819 ± 0.010</td>
<td>0.585 ± 0.010</td>
<td><strong>0.623 ± 0.009</strong></td>
<td>0.479 ± 0.010</td>
<td>0.347 ± 0.005</td>
</tr>
<tr>
<td>BP-MLL</td>
<td>0.749 ± 0.014</td>
<td>0.574 ± 0.002</td>
<td>0.619 ± 0.009</td>
<td>0.466 ± 0.008</td>
<td>0.304 ± 0.006</td>
</tr>
<tr>
<td>MLPE</td>
<td>0.814 ± 0.007</td>
<td><strong>0.621 ± 0.004</strong></td>
<td>0.619 ± 0.010</td>
<td><strong>0.508 ± 0.025</strong></td>
<td><strong>0.396 ± 0.002</strong></td>
</tr>
<tr>
<td>MLL</td>
<td>0.703 ± 0.013</td>
<td>0.585 ± 0.008</td>
<td>0.451 ± 0.016</td>
<td>0.464 ± 0.025</td>
<td>0.375 ± 0.003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparing algorithms</th>
<th>scene</th>
<th>emotions</th>
<th>enron</th>
<th>medical</th>
<th>bitbex</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLMTL</td>
<td>0.735 ± 0.006</td>
<td><strong>0.621 ± 0.025</strong></td>
<td>0.463 ± 0.018</td>
<td><strong>0.540 ± 0.017</strong></td>
<td><strong>0.318 ± 0.004</strong></td>
</tr>
<tr>
<td>ML-KNN</td>
<td>0.594 ± 0.012</td>
<td>0.245 ± 0.020</td>
<td>0.379 ± 0.040</td>
<td>0.133 ± 0.035</td>
<td>0.194 ± 0.005</td>
</tr>
<tr>
<td>BP-MLL</td>
<td>0.710 ± 0.034</td>
<td>0.320 ± 0.043</td>
<td><strong>0.471 ± 0.040</strong></td>
<td>0.210 ± 0.014</td>
<td>0.177 ± 0.004</td>
</tr>
<tr>
<td>MLPE</td>
<td>0.600 ± 0.012</td>
<td>0.210 ± 0.022</td>
<td>0.424 ± 0.012</td>
<td>0.219 ± 0.045</td>
<td>0.206 ± 0.003</td>
</tr>
<tr>
<td>MLL</td>
<td>0.432 ± 0.030</td>
<td>0.261 ± 0.021</td>
<td>0.401 ± 0.025</td>
<td>0.260 ± 0.035</td>
<td>0.190 ± 0.007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparing algorithms</th>
<th>scene</th>
<th>emotions</th>
<th>enron</th>
<th>medical</th>
<th>bitbex</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLMTL</td>
<td>0.776 ± 0.005</td>
<td>0.606 ± 0.015</td>
<td><strong>0.523 ± 0.017</strong></td>
<td><strong>0.492 ± 0.011</strong></td>
<td><strong>0.339 ± 0.003</strong></td>
</tr>
<tr>
<td>ML-KNN</td>
<td>0.683 ± 0.014</td>
<td>0.331 ± 0.022</td>
<td>0.464 ± 0.029</td>
<td>0.185 ± 0.043</td>
<td>0.234 ± 0.005</td>
</tr>
<tr>
<td>BP-MLL</td>
<td>0.723 ± 0.024</td>
<td>0.394 ± 0.041</td>
<td>0.520 ± 0.029</td>
<td>0.267 ± 0.015</td>
<td>0.204 ± 0.004</td>
</tr>
<tr>
<td>MLPE</td>
<td>0.684 ± 0.013</td>
<td>0.293 ± 0.027</td>
<td>0.486 ± 0.011</td>
<td>0.292 ± 0.048</td>
<td>0.254 ± 0.003</td>
</tr>
<tr>
<td>MLL</td>
<td>0.527 ± 0.029</td>
<td>0.343 ± 0.023</td>
<td>0.420 ± 0.019</td>
<td>0.326 ± 0.037</td>
<td>0.238 ± 0.008</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparing algorithms</th>
<th>scene</th>
<th>emotions</th>
<th>enron</th>
<th>medical</th>
<th>bitbex</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLMTL</td>
<td>0.694 ± 0.008</td>
<td>0.468 ± 0.012</td>
<td>0.491 ± 0.019</td>
<td>0.317 ± 0.010</td>
<td>0.310 ± 0.002</td>
</tr>
<tr>
<td>ML-KNN</td>
<td>0.585 ± 0.010</td>
<td>0.262 ± 0.015</td>
<td>0.445 ± 0.018</td>
<td>0.140 ± 0.036</td>
<td>0.240 ± 0.005</td>
</tr>
<tr>
<td>BP-MLL</td>
<td>0.558 ± 0.017</td>
<td>0.330 ± 0.031</td>
<td>0.357 ± 0.018</td>
<td>0.173 ± 0.011</td>
<td>0.135 ± 0.003</td>
</tr>
<tr>
<td>MLPE</td>
<td>0.569 ± 0.014</td>
<td>0.224 ± 0.024</td>
<td>0.459 ± 0.012</td>
<td>0.224 ± 0.044</td>
<td>0.252 ± 0.003</td>
</tr>
<tr>
<td>MLL</td>
<td>0.419 ± 0.029</td>
<td>0.269 ± 0.019</td>
<td>0.438 ± 0.020</td>
<td>0.251 ± 0.034</td>
<td>0.230 ± 0.007</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Comparing algorithms</th>
<th>scene</th>
<th>emotions</th>
<th>enron</th>
<th>medical</th>
<th>bitbex</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLMTL</td>
<td>0.688 ± 0.008</td>
<td>0.483 ± 0.014</td>
<td><strong>0.492 ± 0.019</strong></td>
<td><strong>0.314 ± 0.012</strong></td>
<td><strong>0.289 ± 0.003</strong></td>
</tr>
<tr>
<td>ML-KNN</td>
<td>0.648 ± 0.012</td>
<td>0.290 ± 0.014</td>
<td>0.459 ± 0.013</td>
<td>0.179 ± 0.041</td>
<td>0.228 ± 0.005</td>
</tr>
<tr>
<td>BP-MLL</td>
<td>0.578 ± 0.013</td>
<td>0.340 ± 0.031</td>
<td>0.384 ± 0.013</td>
<td>0.181 ± 0.022</td>
<td>0.127 ± 0.005</td>
</tr>
<tr>
<td>MLPE</td>
<td>0.634 ± 0.015</td>
<td>0.276 ± 0.021</td>
<td>0.491 ± 0.012</td>
<td>0.279 ± 0.044</td>
<td>0.245 ± 0.004</td>
</tr>
<tr>
<td>MLL</td>
<td>0.507 ± 0.025</td>
<td>0.325 ± 0.016</td>
<td>0.449 ± 0.010</td>
<td>0.299 ± 0.039</td>
<td>0.229 ± 0.008</td>
</tr>
</tbody>
</table>

3) IMPACT OF THE PERCENTAGES OF LABELED INSTANCE IN TEST DATA

To study the variations in MLMTL’s performance with increasing labeled instances in the test data, we selected 3% (4%, 5%, 6%, 7%) data of test dataset with training data to learn a predictive classifier for labeling remaining 97% (96%, 95%, 94%, 93%) of test instances. To illustrate parameter sensitivity, we show the performance of the proposed algorithm with under ratios on all datasets. Fig. 4 summarizes the experimental results of MLMTL in terms of Average-Precision, Average-Recall, Macro-F1, and Micro-F1 in all datasets. As the results show, we also notice that as this ratio increases, the performance of MLMTL improved slightly for those evaluation measures. In summary, the experimental results demonstrate the effectiveness of the proposed MLMTL method. Hence, when compared with other algorithms, MLMTL chose the least amount of labeled instances, 3% in the test dataset.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel MLMTL approach to perform multi-label learning, which simultaneously considers instance-level distribution divergence and label-level distribution divergence between training and test datasets. Compared with the previous work, MLMTL is the first to relax these distribution divergence in multi-label learning, which employs the instance weights to bridge the training and test datasets. Moreover, it provides a solution for more efficiently preserving and adopting the intrinsic geometric information among the label spaces. We have compared our method with four state-of-the-art multi-label learning approaches. The results show that MLMTL outperforms these methods on five real-world datasets.

Several problems remain to be investigated in future work. One example is how to extend our MLMTL method with a state-of-the-art solution for the class imbalance problem. In addition, we plan to apply the proposed MLMTL approach to perform multi-label learning tasks across multiple domains.

REFERENCES


TENGYUN WANG is currently pursuing the Ph.D. degree with the School of Software Engineering, South China University of Technology. His research interests include data mining, machine learning, and applied data science, including computational advertising.

HAIZHI YANG is currently pursuing the Ph.D. degree in software engineering with the South China University of Technology. His research interests include machine learning, data mining, and applied data science, including computational advertising.

SHAQIJAN QIU received the M.Sc. degree from The Chinese University of Hong Kong, in 2013. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His research interests include transfer learning, software metric, defect prediction, and fault localization.

HAN YU received the B.Eng. (Hons.) and Ph.D. degrees from the School of Computer Engineering, Nanyang Technological University (NTU), Singapore, in 2007 and 2014, respectively. He is currently a Lee Kuan Yew Postdoctoral Fellow with the Joint NTU–UBC Research Centre of Excellence in Active Living for the Elderly (LILY). His research focus is on artificial intelligence–powered crowd work. Specifically, he designs stochastic optimization techniques to enhance crowd workers’ experience through data-driven algorithmic management.

HENGJIE SONG received the B.Sc. degree in electrical engineering and the M.Sc. degree in aerospace computer design from the Harbin Institute of Technology, in 1998 and 2002, respectively, and the Ph.D. degree from Nanyang Technological University, Singapore. He is currently a Professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. His research interests include data mining and artificial intelligence. He has served as a PC-Reviewer for the 21th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

***