Pattern Recognition 135 (2023) 109189

Contents lists available at ScienceDirect

Pattern Recognition

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

A Novel Label Enhancement Algorithm Based on Manifold Learning

Chao Tan[a,*], Sheng Chen[b,c], Xin Geng[d], Genlin Ji[a]

[a] School of Computer and Electronic Information/School of Artificial Intelligence, Nanjing Normal University, Nanjing 210023, China
[b] School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK
[c] Department of Computer Science and Technology, Ocean University of China, Qingdao 266100 China
[d] School of Computer Science and Engineering, Southeast University, Nanjing 210096, China

A R T I C L E I N F O

Article history:
Received 2 June 2022
Revised 24 October 2022
Accepted 13 November 2022
Available online 20 November 2022

Keywords:
Multi-label learning
Label enhancement
Incremental subspace learning
Label propagation
Manifold learning
Conditional random field

A B S T R A C T

We propose a label enhancement model to solve the multi-label learning (MLL) problem by using the incremental subspace learning to enrich the label space and to improve the ability of label recognition. In particular, we use the incremental estimation of the feature function representing the manifold structure to guide the construction of the label space and to transform the local topology from the feature space to the label space. First, we build a recursive form for incremental estimation of the feature function representing the feature space information. Second, the label propagation is used to obtain the hidden supervisory information of labels in the data. Finally, an enhanced maximum entropy model based on conditional random field is established as the objective, to obtain the predicted label distribution. The enriched label information in the manifold space obtained in first step and the estimated label distributions provided in second step are employed to train this enhanced maximum entropy model by a gradient-descent iterative optimization to obtain the label distribution predictor's parameters with enhanced accuracy. We evaluate our method on 24 real-world datasets. Experimental results demonstrate that our label enhancement manifold learning model has advantages in predictive performance over the latest MLL methods.

1. Introduction

For many real-world multi-label problems, different labels have different importance. For example, a natural scene image is labeled with multiple labels such as 'sky', 'water', 'forest' and 'cloud', but the degree to which these labels describe the image is different [1]. There are many similar examples in diverse applications, where multiple labels related to an example do not have the same importance to the example. To reflect the different degrees of importance for the set of multiple labels, a more natural way to label an instance $\boldsymbol{x}$ is to assign a real number $d_{\boldsymbol{x}}^{y}$ to each possible label y, to represent the degree to which y describes $\boldsymbol{x}$. We can choose $d_{\boldsymbol{x}}^{y} \in [0,\ 1]$, and make the label set complete, i.e., using all the labels in the set always fully describes the instance. Then, $\sum_y d_{\boldsymbol{x}}^{y} = 1$. Such a $d_{\boldsymbol{x}}^{y}$ is called the description degree of y to $\boldsymbol{x}$. The description degrees of all the labels for an instance form a data structure conforming to the probability distribution called label distribution. The learning process on the instances labeled by label distributions is called label distribution learning (LDL).

Label distribution is more general in most supervised learning problems because the relevance or irrelevance of a label to an instance is essentially relative. When multiple labels are associated with an instance, the relative importance among them is more likely to be different than exactly equal. However, in practice, it is not realistic to directly obtain the descriptive degree of each label in many applications. This is because the process of quantifying the description degrees is costly, and there is often no objective quantitative standard for the descriptiveness of each label. The current common data labeling method is that an instance $\boldsymbol{x}$ is assigned with $l_{\boldsymbol{x}}^{y} \in \{0, 1\}$ to each possible label y. If $l_{\boldsymbol{x}}^{y} = 1$, it means that y is a relevant label of $\boldsymbol{x}$, and if $l_{\boldsymbol{x}}^{y} = 0$, it means that y is an irrelevant label of $\boldsymbol{x}$. $l_{\boldsymbol{x}}^{y}$ expresses the logical relationship of yes or no, and for an instance, the logical vector formed by the logical values $l_{\boldsymbol{x}}^{y}$ of all the labels is called logical label. Most of the existing data use logical labels as the supervision information of instances.

It can be visualized that the supervision information in these data essentially follows a certain label distribution. Although this label distribution is not explicitly given, it is possible to recover it through the analysis of the data set. This process is called label enhancement (LE). LE refers to the process of transforming the original logical labels of the training samples into the label distributions. Similar to the multi-label classification method based on

[*] Corresponding author.
E-mail addresses: tutu_tanchao@163.com (C. Tan), sqc@ecs.soton.ac.uk (S. Chen), xgeng@seu.edu.cn (X. Geng), glji@njnu.edu.cn (G. Ji).

https://doi.org/10.1016/j.patcog.2022.109189
0031-3203/© 2022 Elsevier Ltd. All rights reserved.

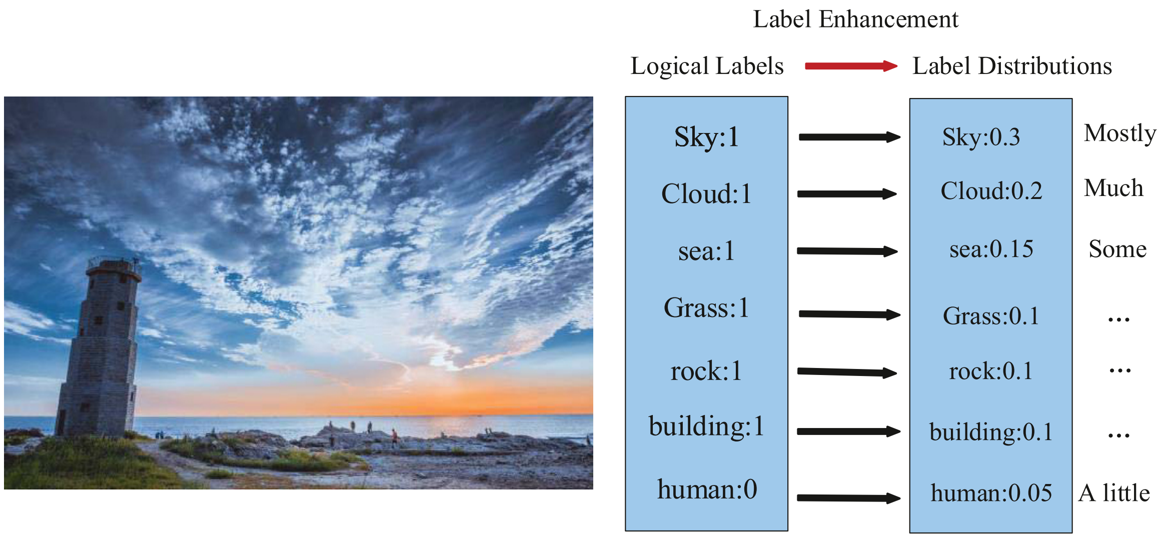

Fig. 1. An example of label enhancement.

embedding [2], LE also relies on the mining of label-related information hidden in the training set. Let $\mathcal{Y}$ be the original logical label space of the samples, and denote $\mathcal{D}$ as the label distribution space after LE. Then, LE expands the original label space $\mathcal{Y}=\{0,1\}^c$ to the label distribution space $\mathcal{D}=[0,\ 1]^c$, where c is the number of labels. In fact, $\mathcal{D}$ constitutes a hypercube in the c-dimensional Euclidean space, and $\mathcal{Y}$ is only located at the vertices of the hypercube. LE reinforces the supervision information in the training set via the correlation among the labels. After the label distributions are recovered, better prediction results can be obtained through LDL. An example of LE is illustrated in Fig. 1.

Motivated by the extensive review of the existing literature for LDL, manifold learning (ML) and LE given in the related work section, in this paper, we propose a label enhancement algorithm based on ML (LEAML) for multi label enhancement learning. The LEAML algorithm is also a probabilistic label learning model to solve the multi-label learning (MLL) problem. Our LEAML consists of the following three components.

1. Manifold space enhanced feature extraction: Based on the incremental semi-supervised subspace learning algorithm [3], we extract accurate and reduced-dimension features in the feature space.
2. The label distribution estimation via label propagation: We extend the label propagation technique to the problem of label distribution prediction, in order to obtain the hidden supervisory information of labels in the data.
3. Prediction from the conditional random field via estimated label distribution: To obtain the predicted label distribution, we construct an enhanced maximum entropy predictor model on a conditional random field. Specifically, we use the enhanced reduced-dimension features obtained in step 1), and we substitute the logical labels with the estimated enriched label distributions acquired in step 2). A gradient-descent optimization is then performed to obtain the maximum likelihood estimate for the parameters of the label distribution predictor.

The rest of this paper is organized as follows. The LDL, ML and LE are first reviewed in Section 2. Our proposed LEAML algorithm is detailed in Section 3, and we emphasize our novel contributions in comparison with the existing LE algorithms. In Section 4, extensive experimental evaluation is carried out to compare our LEAML algorithm with the existing state-of-the-art methods. Our conclusions are given in Section 5.

2. Related work

2.1. Label distribution learning

In the multi-label distribution learning framework, each label y_j of an instance $\boldsymbol{x}$ is assigned a real value $d_{\boldsymbol{x}}^{y_j}$, called the degree of description of $\boldsymbol{x}$ by label y_j, which represents the degree to which label y_j describes $\boldsymbol{x}$. Let the c labels $\boldsymbol{y}^i=\left[y_1^i\ y_2^i\cdots y_c^i\right]\in\mathcal{Y}$ be associated with instance $\boldsymbol{x}_i$. Then the data set of label distributions for $\boldsymbol{x}_i$ can be denoted as $\boldsymbol{d}_i=\boldsymbol{d}_{\boldsymbol{x}_i}^{\boldsymbol{y}^i}=\left[d_{\boldsymbol{x}_i}^{y_1^i}\ d_{\boldsymbol{x}_i}^{y_2^i}\cdots d_{\boldsymbol{x}_i}^{y_c^i}\right]$. Because the label distribution conforms to the probability distribution, much of the statistical theory can be directly applied to the LDL. First, $d_{\boldsymbol{x}}^{y_j}$ can be expressed as a conditional probability form, $d_{\boldsymbol{x}}^{y_j}=P(y_j|\boldsymbol{x})$. Assume that $P(y_j|\boldsymbol{x})$ is a parametric model, denoted as $P(y_j|\boldsymbol{x};\boldsymbol{\theta}_j)$, with the parameter vector $\boldsymbol{\theta}_j$. Then the LDL becomes the learning of the parameter vector $\boldsymbol{\theta}_j$, so that $P(y_j|\boldsymbol{x};\boldsymbol{\theta}_j)$ can output a distribution similar to $d_{\boldsymbol{x}}^{y_j}$. Denote $\boldsymbol{\theta}=\{\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_c\}$. If the Kullback-Leibler (KL) divergence is used as the measure of the similarity between the ground-truth label distribution and the predicted label distribution, the best parameter $\boldsymbol{\theta}^\star$ is determined by

$$\boldsymbol{\theta}^\star=\arg\max_{\boldsymbol{\theta}}\sum_i\sum_j d_{\boldsymbol{x}_i}^{y_j^i}\ln P\left(y_j^i|\boldsymbol{x}_i;\boldsymbol{\theta}_j\right). \tag{1}$$

Once $\boldsymbol{\theta}^\star$ is determined, the unknown label distribution for new sample $(\boldsymbol{x}',y_j')$ can be predicted as $P\left(y_j'|\boldsymbol{x}';\boldsymbol{\theta}_j^\star\right)$.

Using $P(y_j|\boldsymbol{x};\boldsymbol{\theta}_j)$ for classification is equivalent to the maximum posterior probability decision [4]. This shows that single-label learning (SLL) is a special case of LDL.

For MLL, the descriptive degree $d_{\boldsymbol{x}_i}^{y_j^i}$ of the relevant label of each instance $\boldsymbol{x}_i$ satisfies [5]

$$d_{\boldsymbol{x}_i}^{y_j^i}=\begin{cases}\frac{1}{|Y_i|}, & y_j^i\in Y_i,\\ 0, & y_j^i\notin Y_i,\end{cases} \tag{2}$$

where $Y_i=\{j:y_j^i=1,\forall j\}$ is the relevant label set of $\boldsymbol{x}_i$. By using (2), the optimization (1) can be rewritten as:

$$\boldsymbol{\theta}^\star=\arg\max_{\boldsymbol{\theta}}\sum_i\frac{1}{|Y_i|}\sum_{y_j^i\in Y_i}\ln P\left(y_j^i|\boldsymbol{x}_i;\boldsymbol{\theta}_j\right). \tag{3}$$

The above formula can be seen as first using entropy-based label assignment [6] to convert the multi-label data set into a weighted single-label data set, and then performing the maximum likelihood estimation of $\boldsymbol{\theta}$. Therefore, MLL is also a special case of LDL. Hence, LDL can be regarded as a more general learning framework than SLL and MLL.

There are three categories of LDL algorithms under this learning framework [7]. The first category includes PT-Bayes and PT-SVM [7], which transform the LDL problem into a traditional SLL or MLL problem. The second category transforms the SLL or MLL algorithm into a learning algorithm that can process label distribution data, such as AA-kNN and AA-BP [7]. The algorithms of the third category are specifically designed according to the inherent characteristics of LDL, and they include SA-IIS and SA-BFGS [7].

The above algorithms mostly apply the true label distributions and the predicted label distributions in the KL divergence metric to measure the distance between the two distributions, and then use the maximum entropy model to establish a parametric model of the label distribution. Since most real-world datasets do not have label distribution, we can only predict the label distributions $d_{\boldsymbol{x}_i}^{y_j^i}$ of the training set using the labeled samples $\left\{\boldsymbol{x}_i, \boldsymbol{y}^i\right\}$, that is, training the parametric model of the label distribution based on $\left\{\boldsymbol{x}_i, \boldsymbol{y}^i\right\}$ by a semi-supervised algorithm [7].

2.2. Manifold learning

ML has been widely combined with various state-or-the-art learning approaches, e.g., deep learning, to solve the challenging problems in many practical applications. Below we review some typical applications of ML.

Hong *et al.* [8] proposed a multi-task learning framework based on deep convolutional neural network (DCNN). In this framework, DCNN-based feature mapping and multi-task learning are connected to carry out DCNN-based regression for face-pose estimation. This framework unifies the multiview problem and multimodal problem in a single model. In addition, the authors adopted manifold regularization to form manifold regularized convolutional layers, so that the inner relationship of neurons can be utilized to preserve the local properties of neurons, leading to better feature representation.

Yu *et al.* [9] proposed a learning-to-rank model to jointly consider visual features and click features in image retrieval. A robust and accurate ranking model is built by using the click features, and the visual features are utilized to further enhance the model's performance. By integrating the visual features and click features, the authors designed an objective function, in which the hypergraph regularizer and linear model are adopted to respectively take these two features into consideration. An efficient algorithm based on fast alternating linearization was designed to solve the resulting optimization.

In [10], the authors devised a hierarchical deep word embedding (HDWE) model by integrating sparse constraints and an improved RELU operator to address click feature prediction from noisy and sparse visual features. Compared with traditional embedding models, HDWE can better predict the click features of images from coarse to fine through hierarchical semantics.

In [11], the authors proposed a pose recovery method using nonlinear mapping with deep neural network. Specifically, feature extraction is based on multimodal fusion and back-propagation deep learning. In multimodal fusion, a unified feature description is constructed using a hypergraph Laplacian with low-rank representation, while in back-propagation deep learning, the nonlinear mapping is learned from 2D image to 3D pose with parameter fine-tuning.

To address high dimensionality of image features and low efficiency of retrieving process in image-based 3D human pose recovery, the authors of [12] proposed an approach to recover 3D human poses from silhouettes by adopting locality sensitive sparse coding in the retrieving process. The method incorporates a local similarity preserving term into the objective of sparse coding, which groups similar silhouettes and alleviates the instability of sparse codes.

2.3. Label enhancement

Both traditional SLL and MLL treat each label as a logical indicator, with +1 indicating a relevant label and -1 (or 0) indicating an irrelevant label. This type of label is called logical label, which cannot provide the explicit relative importance of each label. By contrast, label distribution that assigns a real value from 0 to 1 to each label to indicate the relative importance of the label to the instance is called numerical label. However, for real-world applications, it is difficult to obtain the importance of a label directly. If the label importance can be obtained and used in training, it will greatly enhance MLL. Therefore, we need a way to reconstruct numerical labels from logical multi-label data. This is called LE. We now review the latest LE algorithms.

2.3.1. LE Algorithm based on label propagation

Graph-based LE algorithm represents the topological structure between instances with a graph model. Based on some reasonable assumptions, the relationship between inter-instance correlation and inter-label correlation can be established, and the logical labels of the instance are enhanced into the label distributions. Specifically, Zhang *et al.* [13] applied the label propagation (LP) method in semi-supervised learning to LE. This LE algorithm based on LP represents the topological structure between instances by using graph model. First, a label propagation matrix is constructed based on the correlation between instances. The algorithm utilizes the different path weights in the propagation process to make the description of different labels naturally different, to reflect the inter-label relationship embedding in the training data. This LP algorithm is in fact corresponds to the regularization framework of [14].

However, this LP process imposes high complexity due to the calculation of paired distances in the whole feature space, and some unnecessary information may be introduced, leading to the decrease of accuracy. More importantly, the LP algorithm is essentially the propagation of logical label, and the final normalization is used to force the logical label into the label distribution, which cannot reflect the essence of LE, namely, to predict the label distribution of unknown instances through the relationship between known instances.

2.3.2. LE Algorithm based on ML

Hou *et al.* [15] proposed an LE method based on ML. Similar to the LE based on LP, a fully connected graph is constructed using the training examples, and the algorithm establishes the relationship between instances correlation and label correlation based on smoothness hypothesis [16]. The underlying assumption is that the data are distributed on certain manifold in both feature space and label space. To explore the local topological structure in the training set, the local topological structure between the examples is obtained by solving the linear relationship according to the locally linear embedding [17]. By reconstructing the manifold of the feature space and the label space, the topological relationship of the feature space is transferred into the label space via the smoothness assumption. Hence, the topological relationship of the feature space manifold is used to guide the construction of the label space manifold, thereby the logical labels of the examples are enhanced to the label distributions.

This ML algorithm [15] enhances the logical labels into the label distribution via two separate steps. First, it reconstructs the structural information in the label space from the feature space, and second it uses the quadratic programming to solve the label prediction of unknown instances. These two steps require the two separate optimization processes with the two separate objective functions, which inevitably introduces error and reduces the prediction accuracy.

2.3.3. Graph laplacian LE

Xu *et al.* [18] proposed an LE algorithm called graph Laplacian LE (GLLE), to recover the label distributions from the logical labels by mining the hidden importance from training instances through the topological information of the feature space. GLLE is also a typical representative of LE based on graph model. Unlike the LP algorithm [13], which calculates the distance between samples in the whole feature space, GLLE selects the instance's k nearest neighbors and calculates the distance between the instance and its k nearest neighbors, which reduces the computational cost and improves the accuracy. GLLE also integrates the topological information of the feature space and the loss function that predicts the label distribution through logical label into a single combined objective function, thus, avoiding the need to construct the two separate objective functions, as in the case of the ML algorithm [15].

Like other LE algorithms, GLLE has a natural 'defect', as its first component loss function also models the difference between the logical labels and the predicted label distributions. Therefore, like other LE algorithms, it also choose the numerical label as close as possible to the original logical label. But this is not consistent with the 'physical' interpretation of the label distribution, namely, representing the degree to which the original label describes the instance.

2.3.4. LE With sample correlations

The LE with sample correlations (LESC) via low-rank representation algorithm [19] obtains the label distribution by exploiting the low-rank representation to excavate the global information in the feature space, which is different from the GLLE [18] that exploits the local similarity. The first component loss function for the LESC algorithm is the same as that of the GLLE, but its second component loss function is based on the low-rank representation, which is different from that of the GLLE.

The BFGS [20] is adopted to solve the optimization problem associated with the LESC, and hence to obtain the label distributions. However, the convergence of BFGS is hard to determine. The usual practice is to determine the number of iterations manually. This is time consuming particularly for large-size problems.

2.3.5. Multilabel distribution learning based on multi-output regression and ML

Recently, we proposed a multilabel distribution learning algorithm based on multi-output regression through ML, referred to as MDLRML [21]. By exploiting the samples' ML and the LDL, we link these two spaces' similar and smooth manifolds. This facilitates using the topological relationship of the manifolds in the feature space to guide the manifold construction of the label space. The smoothest regression function is used to fit the manifold data, and a locally constrained multi-output regression is designed to improve the data's local fitting. Based on the regression results, we enhance the logical labels into the label distributions, thereby mining and revealing the label's hidden information regarding importance or significance.

2.3.6. Probabilistic label enhancement algorithm

In our recent work [22], we proposed a very different probabilistic LE algorithm, called PLEA, which enhances the logical labels into the label distributions based on the principle that the label distribution represents the degree to which the original label describes the instance. Specifically, the supervised information in the label manifold is utilized in the feature manifold space construction to improve the accuracy of feature extraction, while dramatically reducing the feature dimension. Then the robust linear regression is employed to estimate the label distributions associated with the extracted reduced-dimension features. Using the enhanced reduced-dimension features and their associated estimated label distributions in the enhanced maximum entropy model, the unknown true label distributions are accurately estimated.

2.3.7. Label distribution manifold learning algorithm

Very recently, we developed a novel label distribution ML (LDML) method [23] to solve the LDL problem. We first extract the accurate and reduced-dimension features of the training data using ML. Then we estimate the unknown label distributions associated with the extracted features based on multi-output kernel regression. The extracted reduced-dimension features and their associated estimated label distributions are used to design an enhanced maximum entropy model, which enables us to estimate the unknown true label distributions for the training data accurately and efficiently. We also proposed to apply the tangent space alignment regression in the second stage, resulting in the LDML-R algorithm [23] that has better LDL performance at the cost of imposing higher complexity, compared with the LDML.

In the next section, after deriving our LEAML method, we will point out the difference or novelty of the LEAML, in comparison with the existing methods surveyed in this section. Also in Section 4, we will use the seven latest LE methods reviewed in this section, namely, the LP [13], the ML [15], the GLLE [18], the LESC [19], the MDLRML [21], the PLEA [22] and the LDML-R [23], as the benchmarks for the performance comparison with our proposed LEAML.

3. The proposed algorithm

The goal of LDL is to build a parametric model for the label distribution $d_{\boldsymbol{x}}^{y_j}=P(y_j|\boldsymbol{x};\boldsymbol{\theta}_j)$, $1\le j\le c$, for sample $\boldsymbol{x}$ and its c logical labels y_j. After obtaining the parameters $\{\boldsymbol{\theta}_j, 1\le j\le c\}$, the label distributions $d_{\boldsymbol{x}'}^{y'_j}$ of new data $(\boldsymbol{x}', y'_j)$ can be predicted. According to the smooth assumption[16], samples close to each other in the feature space are likely to have the same logical labels. Based on this property, it can be inferred that points close to each other in the feature space are likely to have similar numerical label vectors. This leads to the hypothesis that the label space and the feature space have similar local topological structures. The topology of the feature space can be represented as a graph $\mathcal{G}=(\mathcal{V},\mathcal{E},\mathcal{W})$, where $\mathcal{V}=\{\boldsymbol{x}_i, 1\le i\le n\}$ is the set of vertices consisting of all the training examples, and $\mathcal{E}$ is the set of all the edges in the graph, with edge $\boldsymbol{e}_{i,j}$ representing the relationship between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, while $\mathcal{W}$ is characterized by the weight matrix $\boldsymbol{W}=\left[w_{i,j}\right]_{n\times n}$, with $w_{i,j}$ being the weight of edge $\boldsymbol{e}_{i,j}$. In order to estimate the local topological structure of the feature space, the local neighbor information of each example is used to construct the graph $\mathcal{G}$. According to [15], each example can be reconstructed from its neighbors by a linear combination. Therefore, the local topological structure of the feature space can be obtained by solving the following optimization

$$\min_{\boldsymbol{W}}\sum_{i=1}^{n}\left\|\boldsymbol{x}_i-\sum_{j\ne i}w_{i,j}\boldsymbol{x}_j\right\|^2, \tag{4}$$

in which $w_{i,j}=0$ if $\boldsymbol{x}_j$ is not an k-nearest neighbor of $\boldsymbol{x}_i$. The above optimization can be transformed into a quadratic programming problem

$$\min_{\boldsymbol{W}}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{G}\boldsymbol{W}, \tag{5}$$

where $\boldsymbol{G}=\left[g_{i,j}\right]_{n\times n}$ with $g_{i,j}=(\boldsymbol{x}_i-\boldsymbol{x}_j)^{\mathrm{T}}(\boldsymbol{x}_i-\boldsymbol{x}_j)$.

Since the feature space and the label space should have a similar local topological structure, the following regularization term can be introduced [24]

$$R = \|\boldsymbol{U} - \boldsymbol{W}\boldsymbol{U}\|_F^2 = \mathrm{tr}\left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{U}\right), \tag{6}$$

where $\boldsymbol{M} = (\boldsymbol{I}-\boldsymbol{W})^{\mathrm{T}}(\boldsymbol{I}-\boldsymbol{W})$, and $\boldsymbol{U} \in \mathbb{R}^{n\times c}$ represents the numerical label space, composed of the eigenvectors corresponding to the c minimum eigenvalues of $\boldsymbol{M}$, while $\boldsymbol{I}$ is the identity matrix of appropriate dimension. To train a mapping from the logical label space to the numerical label space, the regression model can be obtained by solving the regularized optimization problem:

$$\min_{\boldsymbol{U}} \left\|\boldsymbol{U} - \boldsymbol{Y}^{\mathrm{T}}\right\|_F^2 + \mathrm{tr}\left(\boldsymbol{U}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{U}\right), \tag{7}$$

where $\boldsymbol{Y} = \left[\boldsymbol{y}^1\ \boldsymbol{y}^2 \cdots \boldsymbol{y}^n\right] \in \mathbb{R}^{c\times n}$ is the logical label matrix. This optimization can be solved by quadratic programming, to construct the label space $\boldsymbol{U}$.

Given the dataset of size n, the algorithm first constructs the graph, i.e., compute $\boldsymbol{W} \in \mathbb{R}^{n\times n}$. Then it needs to construct the label space $\boldsymbol{U} \in \mathbb{R}^{n\times c}$ by solving the optimization (7), which may impose high complexity, particularly for large n. Therefore, we employ incremental label space construction method to construct $\boldsymbol{U}$ as described in the following subsection.

3.1. Manifold space enhanced feature extraction

For the regularization term of the objective function, let $\boldsymbol{v}_k$ be the eigenvector of $\boldsymbol{M}_k$ associated with the eigenvalue λ_k at the kth iteration, namely, $\boldsymbol{M}_k\boldsymbol{v}_k = \lambda_k\boldsymbol{v}_k$. Define $\boldsymbol{u}_k = \boldsymbol{M}_k\boldsymbol{v}_k$. Then $\widehat{\boldsymbol{u}}_n = \frac{1}{n}\sum_{k=1}^{n}\boldsymbol{u}_k$ is the nth step estimate of $\boldsymbol{u}$, which can be expressed as

$$\begin{aligned}\widehat{\boldsymbol{u}}_n &= \frac{1}{n}\sum_{k=1}^{n}\boldsymbol{u}_k = \frac{1}{n}\sum_{k=1}^{n}\boldsymbol{M}_k\boldsymbol{v}_k = \frac{n-1}{n}\left(\frac{1}{n-1}\sum_{k=1}^{n-1}\boldsymbol{M}_k\boldsymbol{v}_k\right) + \frac{1}{n}\boldsymbol{M}_n\boldsymbol{v}_n \\ &= \frac{n-1}{n}\widehat{\boldsymbol{u}}_{n-1} + \frac{1}{n}\boldsymbol{M}_n\boldsymbol{v}_n,\end{aligned} \tag{8}$$

where $\boldsymbol{v}_n$ is the nth step estimate of $\boldsymbol{v}$. Based on the statistical efficiency [3], it is easy to get $\lambda = \left\|\widehat{\boldsymbol{u}}\right\|$, $\boldsymbol{v} = \frac{\widehat{\boldsymbol{u}}}{\left\|\widehat{\boldsymbol{u}}\right\|}$, and the following lemma.

Lemma 1. *Let $\{\boldsymbol{M}_n\}$ be a sequence of real matrices. If $\lim_{n\to\infty}\boldsymbol{M}_n = \boldsymbol{M}$, then $\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{M}_i = \boldsymbol{M}$, where $\boldsymbol{M}_i$ is the estimate of $\boldsymbol{M}$ at the ith step.*

Since $\lim_{n\to\infty}\boldsymbol{v}_n = \lim_{n\to\infty}\frac{\widehat{\boldsymbol{u}}_{n-1}}{\left\|\widehat{\boldsymbol{u}}_{n-1}\right\|} = \boldsymbol{v}$, $\boldsymbol{v}_n$ can be estimated by $\frac{\widehat{\boldsymbol{u}}_{n-1}}{\left\|\widehat{\boldsymbol{u}}_{n-1}\right\|}$. This leads to a recursive formula for incremental estimation of the feature function representing the weight of the sample edge in the feature space

$$\widehat{\boldsymbol{u}}_n = \frac{n-1}{n}\widehat{\boldsymbol{u}}_{n-1} + \frac{1}{n}\boldsymbol{M}_n\frac{\widehat{\boldsymbol{u}}_{n-1}}{\left\|\widehat{\boldsymbol{u}}_{n-1}\right\|}, \tag{9}$$

where $\boldsymbol{M}_n = (\boldsymbol{I}-\boldsymbol{W}_n)^{\mathrm{T}}(\boldsymbol{I}-\boldsymbol{W}_n)$ with $\boldsymbol{W}_n$ been the nth-step estimation of $\boldsymbol{W}$. Initially, we set $\widehat{\boldsymbol{u}}_1 = \boldsymbol{u}_1 = \boldsymbol{M}_1\boldsymbol{v}_1$, the first direction of data spread.

The procedure (9) estimates the first major eigenvector. For the notational convenience, we denote this first eigenvector by $\widehat{\boldsymbol{u}}_{1,n}$. To compute the other subsequent eigenvectors $\widehat{\boldsymbol{u}}_{j,n}$, $2 \le j \le c$, we use the iterative procedure of [3]. According to [3], it helps to generate 'guiding observations' in a complementary space for computing the subsequent eigenvectors. Let $\boldsymbol{z}_j = \boldsymbol{M}_{[\ :j]}$, where $\boldsymbol{M}_{[\ :j]}$ is the jth column of $\boldsymbol{M}$. To compute the second eigenvector, for example, we first subtract from $\boldsymbol{z}_1$ its projection onto the estimated first eigenvector

$$\boldsymbol{z}_2 = \boldsymbol{z}_1 - \boldsymbol{z}_1^{\mathrm{T}}\frac{\widehat{\boldsymbol{u}}_{1,n}}{\left\|\widehat{\boldsymbol{u}}_{1,n}\right\|}\frac{\widehat{\boldsymbol{u}}_{1,n}}{\left\|\widehat{\boldsymbol{u}}_{1,n}\right\|}. \tag{10}$$

The residual $\boldsymbol{z}_2$ lies in the complementary space of $\widehat{\boldsymbol{u}}_{1,n}$, which serves as the input data to the second iteration step. In this way, the orthogonality is always enforced when the convergence is reached. This effectively exploits the sample available well and thus speeds up the convergence [3].

Combining the mechanism discussed above, we have the recursive form for incremental estimation of the c columns of $\boldsymbol{U}$, which will be used as the feature vectors in the conditional random field based LE of Subsection 3.3.

Procedure : Compute the first k, $1 \le k \le c$, dominant eigenvectors, $\widehat{\boldsymbol{u}}_{1,n}, \widehat{\boldsymbol{u}}_{2,n}, \cdots \widehat{\boldsymbol{u}}_{k,n}$, directly from $\boldsymbol{u}_n$, $n = 1, 2, \cdots$, where $\widehat{\boldsymbol{u}}_{j,n-1}$ is the $(n-1)$th-step estimation of the jth eigenvector. Define $\boldsymbol{M}_{n,n}$ as the nth-step estimation of $\boldsymbol{M}_n = (\boldsymbol{I}-\boldsymbol{W}_n)^{\mathrm{T}}(\boldsymbol{I}-\boldsymbol{W}_n)$ with $\boldsymbol{W}_n$ been the nth-step estimation of $\boldsymbol{W}$. Actually, $\boldsymbol{W}_n$ is computed via the local linear embedding [17], and $\boldsymbol{M}_n = (\boldsymbol{I}-\boldsymbol{W}_n)^{\mathrm{T}}(\boldsymbol{I}-\boldsymbol{W}_n)$ then leads to the first estimation or the 'initial' value of $\boldsymbol{M}_{n,n}$. Further give the initial conditions $\widehat{\boldsymbol{u}}_{j,0} = \boldsymbol{M}_{[:1]}$ and $\boldsymbol{z}_{0,n} = \boldsymbol{M}_{[:1]}$.

For $n = 1, 2, \cdots$ do

For $j = 1, 2, \cdots, \min\{k, n\}$ do

$$\widehat{\boldsymbol{u}}_{j,n} = \frac{n-1}{n}\widehat{\boldsymbol{u}}_{j,n-1} + \frac{1}{n}\boldsymbol{M}_{n,n}\frac{\widehat{\boldsymbol{u}}_{j,n-1}}{\left\|\widehat{\boldsymbol{u}}_{j,n-1}\right\|}, \tag{11}$$

$$\boldsymbol{z}_{j+1,n} = \boldsymbol{z}_{j,n} - \boldsymbol{z}_{j,n}^{\mathrm{T}}\frac{\widehat{\boldsymbol{u}}_{j,n}}{\left\|\widehat{\boldsymbol{u}}_{j,n}\right\|}\frac{\widehat{\boldsymbol{u}}_{j,n}}{\left\|\widehat{\boldsymbol{u}}_{j,n}\right\|}, \tag{12}$$

$$\boldsymbol{M}_{n,n} = \boldsymbol{z}_{j+1,n}\boldsymbol{z}_{j+1,n}^{\mathrm{T}}. \tag{13}$$

According to the statistical efficiency [3], the estimated mean $\widehat{\boldsymbol{u}}_n = \frac{1}{n}\sum_{k=1}^{n}\boldsymbol{u}_k$ is the efficient estimate of the mean of a Gaussian distribution [25]. Our method of using average is motivated its statistical efficiency. When $\widehat{\boldsymbol{u}}_n$ is drawn from a Gaussian distribution, the estimating $\widehat{\boldsymbol{u}}_{i,n}$ has a high statistical efficiency and a fairly low error variance. The c dominant eigenvectors, $\widehat{\boldsymbol{u}}_{j,n}$ for $1 \le j \le c$, incrementally constructed by the above procedure are used as the feature functions to train the label distribution prediction model based on the conditional random field via maximum likelihood estimation in Subsection 3.4.

3.2. Label distribution estimate via label propagation

The LP method [26] is widely used in semi-supervised learning. The core idea of this LP is very simple: the similar data should have the same label. In each iteration of the LP process, each node of the graph exchanges its label information with its connected neighbor nodes. The idea is to select the community label that is the most common label among the connected nodes. As the community label continues spreading, the nodes connected closely will have a common label eventually. Our proposed algorithm extends this LP to the problem of label distribution prediction, to obtain the hidden supervisory information of labels in the data. Specifically, inspired by the LP [26], the label is propagated via the edges between samples. The greater the weight of the edge is, the more similar the two samples are, and the label will propagate easier. As aforementioned, a hidden label distribution can be defined as $d_{\boldsymbol{x}_i}^{y_j^i}$ of the size $n \times c$:

$$d_{\boldsymbol{x}_i}^{y_j^i} = \frac{g_{i,j}}{\sum_{k=1}^{c} g_{i,k}}, \quad 1 \le i \le n,\ 1 \le j \le c. \tag{14}$$

Each element $g_{i,j}$ represents an estimated label distribution.

Give the multi-label training samples $\left\{\boldsymbol{x}_i, \boldsymbol{y}^i\right\}_{i=1}^{n}$, where $\boldsymbol{y}^i = \left[y_1^i\ y_2^i \cdots y_c^i\right]^{\mathrm{T}}$ denotes the c-dimensional logical label vector for the feature sample $\boldsymbol{x}_i$. We have the logical label matrix $\boldsymbol{Y} = \left[\boldsymbol{y}^1\ \boldsymbol{y}^2 \cdots \boldsymbol{y}^n\right]_{c\times n}$ storing all the binary labeling vectors. To estimate the label distribution for $\boldsymbol{x}_i$, we first model the local relationship among $\boldsymbol{x}_i$ and its k nearest neighbors by the linear least squares reconstruction (4) to yield the weight matrix $\boldsymbol{W} = \left[w_{i,j}\right]_{n\times n}$. Then the estimated label distribution $\boldsymbol{g}_i = \left[g_{i,1}\ g_{i,2}\cdots g_{i,c}\right]^{\mathrm{T}}$ is calculated using the formula:

$$\boldsymbol{g}_i = \rho \boldsymbol{y}^i + (1-\rho)\boldsymbol{YW}_{[\ :i]},\ \ 1 \le i \le n, \tag{15}$$

where ρ is the balancing parameter and $\boldsymbol{W}_{[\ :i]}$ is the ith column of $\boldsymbol{W}$. Finally, $\boldsymbol{g}_i$ is normalized according to

$$\widetilde{g}_{i,j} = \frac{g_{i,j}}{\sum_{l=1}^{c} g_{i,l}},\ \ 1 \le i \le n, 1 \le j \le c. \tag{16}$$

Here the well-established LP is used to estimate the label distributions associated with the extracted features. This is different from the PLEA [22] and DML-R [23]. In our proposed algorithm we use this estimated label distribution to train the prediction model based on the conditional random field and perform a maximum likelihood estimation of the prediction model in Subsection 3.4.

3.3. Prediction via estimated label distribution

As described in Subsection 2.1, the label distribution description degree $d_{\boldsymbol{x}}^{y}$ can be represented by the form of conditional probability $d_{\boldsymbol{x}}^{y} = P(y|\boldsymbol{x})$. This may be interpreted as that given an example $\boldsymbol{x}$, the goal of LDL is to learn a conditional probability mass function $P(y|\boldsymbol{x})$ from $\boldsymbol{x}$. Let $f_K(\boldsymbol{x}, y)$ be a feature function that depends on both the instance $\boldsymbol{x}$ and the label y. Then the expected value of f_K is given by the average of f_K over the training set

$$\widetilde{f}_K = \sum_{y} \int \widetilde{p}(\boldsymbol{x}, y) f_K(\boldsymbol{x}, y) d\boldsymbol{x}, \tag{17}$$

where $\widetilde{p}(\boldsymbol{x}, y)$ is the empirical joint distribution. We utilize the conditional random field model [27], which is an effective statistical learning method for labeling problem, to train the parametric model of the label distribution. In the learning phase, the training data set is used to obtain a conditional probability model $\widehat{P}(y|\boldsymbol{x})$ via maximum likelihood estimation. In the predicting phase, given an input sequence $\boldsymbol{x}$, the output sequence $\widehat{y}$ is found with the largest conditional probability $\widehat{P}(y|\boldsymbol{x})$. The main reason why we adopt the conditional random field model is that the maximum likelihood estimation used in the model is particularly suitable for the prediction of normal distribution. As discussed in Subsection 3.1, according to the statistical efficiency [3], $\widehat{\boldsymbol{u}}_{i,n}$ chosen from a Gaussian distribution has a high statistical efficiency.

By defining the probability of a particular label sequence y_j^i given observation sequence $\boldsymbol{x}_i$ to be a normalized product of potential functions [27–29], the conditional random field has the following parametrized form:

$$P\left(y_j^i|\boldsymbol{x}_i\right) = \frac{1}{Z_i}\exp\left(\sum_q \lambda_j^q t_q\left(y_j^{i-1}, y_j^i, \boldsymbol{x}_i\right)\right) + \frac{1}{Z_i}\exp\left(\sum_r \mu_j^r s_r\left(y_j^i, \boldsymbol{x}_i\right)\right), \tag{18}$$

where Z_i is the normalization factor given by

$$Z_i = \sum_{j=1}^{c}\exp\left(\sum_q \lambda_j^q t_q\left(y_j^{i-1}, y_j^i, \boldsymbol{x}_i\right) + \sum_r \mu_j^r s_r\left(y_j^i, \boldsymbol{x}_i\right)\right), \tag{19}$$

$t_q(\cdot,\cdot,\cdot)$ and $s_r(\cdot,\cdot)$ are the transition feature function and the state feature function that depend on both the instance $\boldsymbol{x}_i$ and the label y_j^i, respectively, while λ_j^q and μ_j^r are the corresponding weights. Both $t_q(\cdot,\cdot,\cdot)$ and $s_r(\cdot,\cdot)$ are local features, and they can be any real-valued functions.

As y_j^i is binary, the transition feature function can be represented as:

$$t_q\left(y_j^{i-1}, y_j^i, \boldsymbol{x}_i\right) = \begin{cases} f_q(\boldsymbol{x}_i, y_j^i), & y_j^{i-1} = y_j^i = 1, \\ 0, & y_j^{i-1} = y_j^i = 0. \end{cases} \tag{20}$$

That is, the feature of $\boldsymbol{x}_i$ is extracted only when the label value is 1. Similarly the state feature function can be represented as:

$$s_r\left(y_j^i, \boldsymbol{x}_i\right) = \begin{cases} f_r(\boldsymbol{x}_i, y_j^i), & y_j^i = 1, \\ 0, & y_j^i = 0. \end{cases} \tag{21}$$

It can be seen that both the transition feature functions and the state feature functions can be represented by the 'unified' feature functions of $\boldsymbol{x}_i$, denoted as $f_l(\boldsymbol{x}_i, y_j^i)$, $1 \le l \le k$. In the same way, all the corresponding weights, λ_j^q and μ_j^r, can be unified as θ_j^l, $1 \le l \le k = c$. According to [4,7], the features are further expressed as $f_l(\boldsymbol{x}_i, y_j^i) = y_j^i \widehat{f}_l(\boldsymbol{x}_i)$, where $\widehat{f}_l(\boldsymbol{x}_i)$ is the class-independent lth feature function. Thus, our model (18) can be re-expressed as

$$P\left(y_i^j|\boldsymbol{x}_i; \boldsymbol{\theta}_j\right) = \frac{1}{Z_i}\exp\left(\sum_{l=1}^{c}(\theta_j^l \cdot y_j^i)\widehat{f}_l(\boldsymbol{x}_i)\right), \tag{22}$$

$$Z_i = \sum_{j=1}^{c}\exp\left(\sum_{l=1}^{c}(\theta_j^l \cdot y_j^i)\widehat{f}_l(\boldsymbol{x}_i)\right), \tag{23}$$

where $1 \le j \le c$, $1 \le i \le n$, and $\boldsymbol{\theta}_j = \left[\theta_j^1\ \theta_j^2\cdots\theta_j^c\right]^{\mathrm{T}}$.

The features extracted lies in the feature space of $\boldsymbol{x}_i$, which in fact can be given by the k eigenvectors $\widehat{\boldsymbol{u}}_{l,n}$, $1 \le l \le k = c$, incrementally extracted in Subsection 3.1. Specifically, arrange these c eigenvectors into the matrix form

$$\widehat{\boldsymbol{U}}_n = [\widehat{\boldsymbol{u}}_{1,n}\ \widehat{\boldsymbol{u}}_{2,n}\cdots\widehat{\boldsymbol{u}}_{c,n}] \in \mathbb{R}^{n\times c}. \tag{24}$$

Then the ith row of $\widehat{\boldsymbol{U}}_n$

$$\widehat{\boldsymbol{U}}_{n[i:\,]} = [\widehat{u}_{1,n}[i]\ \widehat{u}_{2,n}[i]\cdots\widehat{u}_{c,n}[i]], \tag{25}$$

where $\widehat{u}_{l,n}[i]$ denotes the ith element of $\widehat{\boldsymbol{u}}_{l,n}$, forms the feature vector $\left[\widehat{f}_1(\boldsymbol{x}_i)\ \widehat{f}_2(\boldsymbol{x}_i)\ \cdots\widehat{f}_c(\boldsymbol{x}_i)\right]$ extracted for $\boldsymbol{x}_i$. Therefore, our model becomes

$$P\left(y_i^j|\boldsymbol{x}_i; \boldsymbol{\theta}_j\right) = \frac{1}{Z_i}\exp\left(\sum_{l=1}^{c}(\theta_j^l \cdot y_j^i)\widehat{u}_{l,n}[i]\right), \tag{26}$$

$$Z_i = \sum_{j=1}^{c}\exp\left(\sum_{l=1}^{c}(\theta_j^l \cdot y_j^i)\widehat{u}_{l,n}[i]\right). \tag{27}$$

3.4. Optimization model

Substituting (26) into (1) and recognizing $\sum_{j=1}^{c} d_{\boldsymbol{x}_i}^{y_j^i} = 1$ yields the target function of $\boldsymbol{\theta} = \left\{\boldsymbol{\theta}_j, 1 \le j \le c\right\}$

$$T(\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{j=1}^{c} d_{\boldsymbol{x}_i}^{y_j^i} \ln P\left(y_i^j|\boldsymbol{x}_i; \boldsymbol{\theta}_j\right) = \sum_{i=1}^{n}\sum_{j=1}^{c} d_{\boldsymbol{x}_i}^{y_j^i}\sum_{l=1}^{c}(\theta_j^l \cdot y_j^i)\widehat{u}_{l,n}[i] - \sum_{i=1}^{n}\ln\left(\sum_{j=1}^{c}\exp\left(\sum_{l=1}^{c}(\theta_j^l \cdot y_j^i)\widehat{u}_{l,n}[i]\right)\right). \tag{28}$$

The target function (28) contains the unknown true label distributions $d_{\boldsymbol{x}_i}^{y_j^i}$. In Subsection 3.2, we have estimated the label distributions $\widetilde{g}_{i,j}$ via LP, which contains more supervisory information than the logical label set. We can use this estimated label distribution via LP to train the prediction model to obtain the label distribution predictor. Specifically, we can substitute the estimated $\widetilde{g}_{i,j}$ for $d_{\boldsymbol{x}_i}^{y_j^i}$ in (28) to arrive at the empirical target function

$$\widetilde{T}(\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{j=1}^{c}\widetilde{g}_{i,j}\sum_{l=1}^{c}(\theta_j^l \cdot y_j^i)\widehat{u}_{l,n}[i] - \sum_{i=1}^{n}\ln\left(\sum_{j=1}^{c}\exp\left(\sum_{l=1}^{c}(\theta_j^l \cdot y_j^i)\widehat{u}_{l,n}[i]\right)\right). \tag{29}$$

A gradient-descent iterative optimization algorithm, called the improved iterative scaling (IIS) [30], can be applied to find the parameters $\boldsymbol{\theta}$ by solving the nonlinear equation associated with the lower bound of $\widetilde{T}(\boldsymbol{\theta}+\Delta\boldsymbol{\theta})-\widetilde{T}(\boldsymbol{\theta})$. Once $\boldsymbol{\theta}_j$, $1 \le j \le c$, are obtained, $P\left(y'_j|\boldsymbol{x}';\boldsymbol{\theta}_j\right)$ can be used to predict for the unknown true label distributions $d_{\boldsymbol{x}'}^{y'_j}$ for new sample $(\boldsymbol{x}', y'_j)$.

The proposed LEAML is summarized in Algorithm 1. The computational complexity of this LEAML consists of three parts as summarized below.

Algorithm 1 Label enhancement algorithm based on manifold learning.

```
Input: Multi-label training sample set $\{\boldsymbol{x}_i \in \mathbb{R}^q, \boldsymbol{y}_i = [y_i^1 \cdots y_i^c]^{\mathrm{T}} \in \{0,1\}^c\}_{i=1}^n$.
Output: Label distribution estimates $\widehat{d}_{\boldsymbol{x}_i}^{y_i^j} = P(y_i^j|\boldsymbol{x}_i;\boldsymbol{\theta}_j)$, $1 \le j \le c$, $1 \le i \le n$.
1: Step 1. Manifold space enhanced feature extraction:
2: After preprocessing, eigenvectors $\widehat{\boldsymbol{u}}_{j,n}$, $\boldsymbol{z}_{j,n}$ obtainedfrom manifold space are enhanced in iteration procedure (11) to (13) ofSubsection 3.1, to get $k$ eigenvectors $\widehat{\boldsymbol{u}}_{l,n}$ incrementally, for$1 \le l \le k = c$.
3: Step 2. Label distribution estimation via LP:
4: Obtain estimated label distribution $\widetilde{\boldsymbol{g}}_i$ via label propagation(15) and (16) of Subsection 3.2.
5: Step 3. Prediction from conditional random field:
6: Form empirical target function (29) with $\widetilde{g}_{i,j}$ estimatedin Step 2 and $\widehat{\boldsymbol{u}}_{l,n}$ extracted in Step 1.
7: Use IIS iterative algorithm to optimize target function (29)to find label distributions' parameters $\boldsymbol{\theta}$.
8: return $d_{\boldsymbol{x}'}^{y'_j} \leftarrow P(y'_j|\boldsymbol{x}';\boldsymbol{\theta}_j)$, $1 \le j \le c$,for new sample $(\boldsymbol{x}', y'_j)$.
```

Step 1. The procedure of the incremental feature extraction has the complexity on the order of $\mathrm{O}\left(n \times c^3\right)$.

Step 2. According to [26], the label propagation has the linear complexity $\mathrm{O}\left(n\right)$. Hence, the complexity of **Step 2** is $\mathrm{O}\left(n\right)$.

Step 3. Let the number of iterations for the IIS algorithm [30] be upper bounded by I_{iis}. Since the complexity per iteration of the IIS optimization is $\mathrm{O}\left(c^2 \times n^2\right)$, the complexity of **Step 3** is $\mathrm{O}\left(I_{\text{iis}} \times c^2 \times n^2\right)$.

3.5. Comparison with existing state-of-the-arts

We now compare our LEAML algorithm with the latest LE approaches to emphasize its novel contributions.

With the exception of the MDLRML [21], PLEA [22] and LDML-R [23], most existing label learning methods assume that the numerical label should be sufficiently close to the original logical label, and choose the difference between the logical labels and the predicted label distributions as the loss function. This principle however is not consistent with the 'physics' of the label distribution, which represents the degree to which the original label describes the instance. Similar to the MDLRML, PLEA and LDML-R, our LEAML directly utilizes the physical interpretation of the label distribution, namely, the conditional probability of the label given the instance, to build the label distribution model, specifically, a conditional random field based prediction model.

Graph semi-supervised learning methods, such as the LP [13] and the GLLE [18], are conceptually appearing, and it is easy to explore the properties of these algorithms through the analysis of the matrix operations involved. However, in the label propagation from the logical label space to the numerical label space, these algorithms need to solve the regularized optimization involving the label matrix variable. This imposes considerable computational complexity, particularly for large-scale data. Therefore, we introduce incremental label learning in our LEAML to reduce the complexity in the label propagation stage.

Many existing label learning methods, such as the ML [15], the MDLRML [21], the PLEA [22] and the LDML-R [23], typically utilize the hypothesis of manifold that the data in the same manifold structure should have the same labels. Our proposed method also exploits the local topology of the feature space to obtain the relative label importance, and uses the distribution to train the prediction model of label distribution. With the exception of the PLEA and LDML-R, the existing multi-label distribution learning algorithms either directly fit the label distribution model or use the maximum entropy model to train the parameters of the label distribution model. By contrast, our algorithm build an enhanced maximum entropy model with the enriched label information in the manifold space and the estimated enhanced distribution information of labels, to train the label distribution prediction model's parameters.

Next we emphasize the differences between the proposed LEAML algorithm and our previous MDLRML, PLEA and LDML-R. The MDLRML [21] is very different from the PLEA, LDML-R and LEAML. It first performs the feature extraction based on the local tangent space alignment algorithm (LTSA) [31]. Then based on the extracted features and their corresponding logical labels, it enhances the logical labels into the label distributions using multi-output regression with sigmoid function. The PLEA [22] also first extracts accurate and reduced-dimension features based on the LTSA. However, it next estimates the label distributions associated with the extracted features based on regression. Then using the extracted reduced-dimensional features and their associated label distribution estimates to form the enhanced maximum entropy model, the unknown label distributions are estimated with enhanced accuracy. The LDML-R first extracts reduced-dimension features in the feature manifold space using the locally linear embedding ML algorithm [17]. It next uses the LTSA based regression to learn the unknown label distributions associated with the extracted reduced-dimension features. Like the PLEA, it then forms the enhanced maximum entropy model to estimate the unknown label distributions. The proposed LEAML has similar three-step algorithmic procedure as the PLEA and LDML-R. However, from Subsection 3.1, the LEAML extracts the features in a different way with incremental learning for reducing complexity. Moreover, from Subsection 3.2, the LEAML estimates the label distributions using the LP method, which imposes lower complexity. Basically, the PLEA and LDML-R focus on LDL, not on label enhancement as our LEAML approach. In the next section, the extensive experimental results demonstrate that the proposed LEAML outperforms the PLEA and LDML-R.

Table 1 compares the complexity of our LEAML with those of the seven benchmarks. It can be seen that the complexity of our LEAML is lower than those of the PLEA, LDML-R, LP and ML. It is less straightforward from Table 1 to draw the conclusion whether our LEAML has lower or higher complexity than the MDLRML, GLLE and LESC. The MDLRML [21] has a two-step algorithmic procedure and our past experience suggests that it is computationally very efficient. It is known that the BFGS algorithm used by GLLE and LESC for nonlinear gradient descent optimization converges very slowly. In the next section, we will use the runtime performance to measure the complexity.

4. Experimental evaluation

All the experiments are carried out on Matlab 2019b, running on a PC with i5-6200 2.30 GHz processor of 4 cores and 8GB of

Table 1
Complexity comparison of various LE algorithms, where k is the number of nearest neighbors, f is the number of features, I_{iis} is the number of iterations for the IIS [22], I_{irwls} is the number of iterations for the iterative reweighed least squares [32], $I_{\text{iteration}}$ is the number of iterations of LP [13] and I_{bfgs} is the number of iterations for the BFGS algorithm [20].

Our LEAML	$O(c^3 \times n + n + I_{\text{iis}} \times c^2 \times n^2)$
PLEA [22]	$O(n \times k^3 + I_{\text{irwls}} \times n^3 + I_{\text{iis}} \times c \times k \times n^2)$
MDLRML [21]	$O(c \times n \times k^3 + I_{\text{irwls}} \times n^3)$
LDML-R [23]	$O(d \times n^3 + I_{\text{irwls}} \times n^3 + I_{\text{iis}} \times c \times k \times n^2)$
LP [13]	$O(n^2 \times f + n^3 + I_{\text{iteration}} \times n^2 \times f)$
ML [15]	$O(n^2 + I_{\text{irwls}} \times n^3)$
GLLE [18]	$O(n^2 \times f + I_{\text{bfgs}} \times n^2)$
LESC [19]	$O(n^2 \times f + I_{\text{bfgs}} \times n^2)$

Table 2
Fourteen multi-label datasets with known ground-truth label distributions from [7] used in label enhancement experiments .

No	Dataset	Examples n	Features q	Labels c
1	SJAFFE	213	243	6
2	Natural_scene	2000	294	9
3	SBU_3DFE	2500	243	6
4	Yeast_spoem	2465	24	2
5	Yeast_alpha	2465	24	18
6	Yeast_cdc	2465	24	15
7	Yeast_cold	2465	24	4
8	Yeast_diau	2465	24	7
9	Yeast_dtt	2465	24	4
10	Yeast_elu	2465	24	14
11	Yeast_heat	2465	24	6
12	Yeast_spo	2465	24	6
13	Yeast_spo5	2465	24	3
14	Movie	7755	1869	5

RAM. Two sets of experiments, LE experiments and MLL experiments, are performed.

4.1. Label enhancement experiments

4.1.1. LE Datasets

We use 14 real-world datasets from [7] in the LE experiments. Basic attributes of these datasets are given in Table 2. The datasets of Yeast-spoem to Yeast-spo5 are collected from the records of 10 biological experiments on the budding yeast genes. The rest of the datasets are collected from facial expression images, natural scene images and movies, respectively. These 14 datasets are labeled with the ground-truth label distributions, and hence they are suitable for evaluating the accuracy of the label distribution prediction.

4.1.2. LE Evaluation measures

The output of an LE algorithm is the label distribution estimate. In order to compare the estimated label distribution with the ground-truth label distribution of the datasets, a natural evaluation measure is the average distance or similarity between the predicted label distribution and the ground-truth label distribution. As suggested in [7], we select the six measures to reflect an LE algorithm's performance from different aspects in semantics [33], and they are: Chebyshev distance (Cheb) ↓, Clark distance (Clark) ↓, Kullback-Leibler divergence (KL) ↓, Canberra distance (Canber) ↓, cosine correlation coefficient (Cosine) ↑, and intersection similarity (Inters) ↑. The first four metrics are distance metrics and hence 'the smaller the better': '↓', while the last two are similarity metrics and therefore 'the larger the better': "↑".

4.1.3. Benchmarks for LE experiments and algorithmic settings

We test the LE performance of our LEAML algorithm with the seven existing state-of-the-art benchmarks reviewed in Section 2, which are the LP [13], ML [15], GLLE [18], LESC [19], MDLRML [21], PLEA [22] and LDML-R [23]

We list the algorithmic parameter settings here. For our LEAML, the balancing parameter ρ in (15) is empirically chosen to be 0.5, and the number of nearest neighbors for (4) is set to $k = 10$. This value is chosen simply to be consistent with the value of k used in the benchmark MDLRML [21]. Other algorithmic parameters of the MDLRML are: the number of low-dimensional embeddings $\widehat{d} = 8$, two loss weightings $\alpha = \beta = 0.5$ to maintain the balance of the two loss terms, and the termination threshold $\xi = 10^{-5}$. For the PLEA [22], the number of extracted principal features is set to $k = 10$, which is the same as the number of nearest neighbors. As for the LDML-R [23], the number of nearest neighbors in feature extraction is set to k=10, and the dimension of the embedded coordinates is given by d=8, which are based on the experience. For the other LE algorithms, we use the original algorithmic settings provided by the authors in their publications. Specifically, we choose the parameter of the LP [13] to be $\alpha = 0.5$. For the ML [15], we have the number of nearest neighbors $k = c + 1$, and its other parameters are $\lambda = 1$, $\epsilon = 0.01$, $C_1 = 1$ and $C_2 = 10$. For the GLLE [18], we choose the parameter λ from $\{0.01, 0.1, \cdots, 100\}$ and set the number of nearest neighbors to $k = c + 1$. As for the LESC [19], the parameter λ is selected among $\{0.0001, 0.001, \cdots, 10\}$.

4.1.4. Label distribution recovery performance

Fig. 2 depicts the flowchart of the label distribution recovery experiment. As a basic task of LE is to predict the label distributions from the logical labels, we need the true logical labels. The true logical labels of the datasets are obtained from the ground-truth label distributions via a binarization process. We recover the

Table 3
Label distribution recovery performance measured by Chebyshev distance ↓.

Dataset	LP	ML	GLLE	LESC	MDLRML	PLEA	LDML-R	LEAML
SJAFFE	0.1070(7)	0.2188(8)	0.0845(6)	0.0692(5)	0.0566(4)	0.0412(2)	0.0375(1)	0.0453(3)
Natural_scene	0.2750(5)	0.2990(6)	0.3353(7)	0.3417(8)	0.1317(4)	0.1106(1)	0.1257(2)	0.1316(3)
SBU_3DFE	0.1230(5.5)	0.1868(8)	0.1230(5.5)	0.1231(7)	0.0794(4)	0.0684(1)	0.0697(2)	0.0727(3)
Yeast_spoem	0.1630(8)	0.1319(7)	0.0870(5.5)	0.0870(5.5)	0.0096(3)	0.0099(4)	0.0046(1)	0.0077(2)
Yeast_alpha	0.0400(8)	0.0387(7)	0.0192(6)	0.0169(5)	0.0073(2)	0.0150(4)	0.0124(3)	0.0054(1)
Yeast_cdc	0.0420(7)	0.0475(8)	0.0217(6)	0.0198(5)	0.0083(2)	0.0081(1)	0.0091(4)	0.0085(3)
Yeast_cold	0.1370(8)	0.1207(7)	0.0650(6)	0.0572(5)	0.0174(3)	0.0180(4)	0.0164(2)	0.0153(1)
Yeast_diau	0.0990(7)	0.2011(8)	0.0530(6)	0.0419(5)	0.0205(4)	0.0194(2)	0.0195(3)	0.0183(1)
Yeast_dtt	0.1280(8)	0.1073(7)	0.0518(6)	0.0466(5)	0.0080(2)	0.0088(4)	0.0070(1)	0.0081(3)
Yeast_elu	0.0440(7)	0.0499(8)	0.0221(6)	0.0208(5)	0.0076(1)	0.0098(4)	0.0079(2)	0.0096(3)
Yeast_heat	0.0860(7)	0.0915(8)	0.0478(6)	0.0466(5)	0.0111(1)	0.0299(4)	0.0121(3)	0.0118(2)
Yeast_spo	0.0900(7)	0.0953(8)	0.0608(5)	0.0609(6)	0.0277(3)	0.0338(4)	0.0274(2)	0.0170(1)
Yeast_spo5	0.1140(7)	0.1514(8)	0.0980(6)	0.0933(5)	0.0531(4)	0.0506(3)	0.0498(2)	0.0162(1)
Movie	0.1517(7)	0.1933(8)	0.1211(4)	0.1395(5)	0.1418(6)	0.0750(1)	0.0967(2)	0.1095(3)
Avg.Rank	7.0357(7)	7.5714(8)	5.7857(6)	5.4643(5)	3.0714(4)	2.7857(3)	2.1429(1.5)	2.1429(1.5)

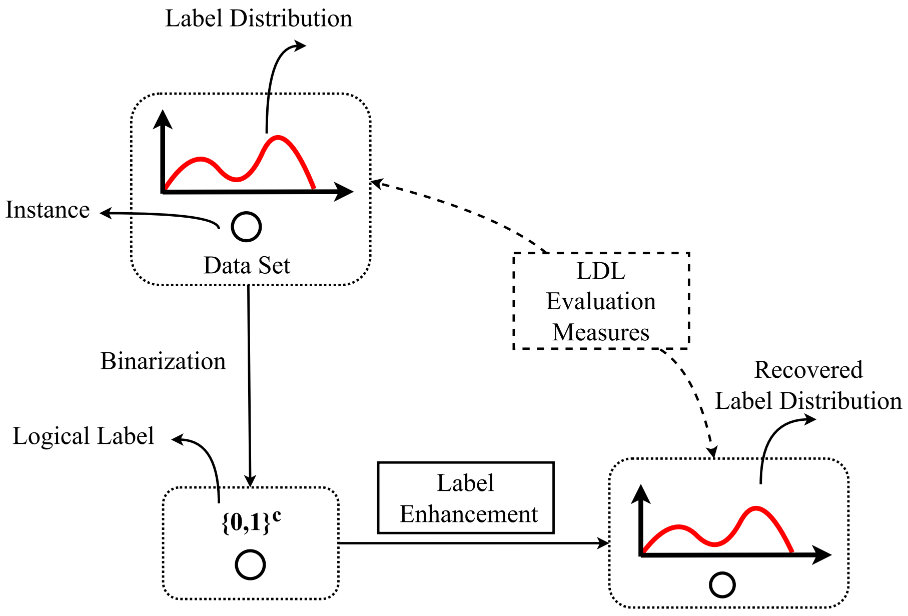

Fig. 2. The flowchart of the label distribution recovery experiment.

Table 4
Label distribution recovery performance measured by Clark distance ↓.

Dataset	LP	ML	GLLE	LESC	MDLRML	PLEA	LDML-R	LEAML
SJAFFE	0.3140(6)	0.8055(8)	0.3633(7)	0.2763(5)	0.2365(4)	0.1620(1)	0.1847(3)	0.1749(2)
Natural_scene	2.4828(8)	2.4520(5)	2.4609(6)	2.4649(7)	2.0868(3)	2.0568(1)	2.0674(2)	2.0915(4)
SBU_3DFE	0.5810(7)	0.7861(8)	0.3818(6)	0.3785(5)	0.2560(3)	0.2401(1)	0.2967(4)	0.2558(2)
Yeast_spoem	0.2718(8)	0.2036(7)	0.1321(6)	0.1295(5)	0.0136(3)	0.0140(4)	0.0065(1)	0.0125(2)
Yeast_alpha	0.4322(7)	0.6025(8)	0.3304(6)	0.2823(5)	0.1027(2)	0.2597(4)	0.1805(3)	0.1003(1)
Yeast_cdc	0.3803(7)	0.5593(8)	0.3018(6)	0.2727(5)	0.1041(2)	0.1174(3)	0.1186(4)	0.0873(1)
Yeast_cold	0.1805(7)	0.3224(8)	0.1738(6)	0.1552(5)	0.0424(3)	0.0453(4)	0.0402(1)	0.0405(2)
Yeast_diau	0.2841(6)	0.7276(8)	0.2964(7)	0.2302(5)	0.0913(2)	0.1114(4)	0.0991(3)	0.0583(1)
Yeast_dtt	0.1902(7)	0.2953(8)	0.1413(6)	0.1278(5)	0.0191(3)	0.0244(4)	0.0168(2)	0.0154(1)
Yeast_elu	0.3642(7)	0.5340(8)	0.2845(6)	0.2617(5)	0.1013(1)	0.1245(4)	0.1087(3)	0.1033(2)
Yeast_heat	0.4886(5)	0.5121(8)	0.4933(6)	0.4941(7)	0.2467(4)	0.2326(3)	0.0556(1)	0.0561(2)
Yeast_spo	0.5585(8)	0.4030(7)	0.2618(6)	0.2596(5)	0.1221(3)	0.1503(4)	0.1215(2)	0.0757(1)
Yeast_spo5	0.2741(7)	0.3015(8)	0.1943(6)	0.1871(5)	0.1023(3)	0.1105(4)	0.0961(2)	0.0229(1)
Movie	0.5220(5)	0.7422(8)	0.5654(6)	0.6266(7)	0.4354(3)	0.3499(2)	0.5201(4)	0.3039(1)
Avg.Rank	6.7857(7)	7.6429(8)	6.1429(6)	5.4286(5)	2.7857(3)	3.0714(4)	2.5000(2)	1.6429(1)

label distributions from the logical labels of the 14 multi-label datasets of Table 2 using the 8 LE algorithms, and compare the estimated label distributions with the ground-truth label distributions. Quantitative results of the 8 algorithms applied to these 14 datasets are compared in Table 3,4,5,6, Table 8 for the six metrics measuring the distance or similarity between the truth label distributions and the recovered label distributions, respectively. Each row of the table presents the metric values attained by the 8 LE algorithms together with the rankings in brackets for the corresponding dataset. We also calculate the algorithms" average ranking performance over the 14 datasets in the last row of the table, where the numerical value before the bracket is the average ranking value over the 14 datasets, and the number in the bracket is again the rank.

As shown in Table 3,4,5,6, Table 8, the results clearly demonstrate the superior LE recovery performance of our LEAML algorithm over the other 7 state-of-art LE benchmark algorithms. Specifically, our LEAML ranks the first in more than 83.33% of the experiments across all the 14 datasets and the 6 evaluation measures, and it ranks the second in 29.63% of the 84 cases. On average, the proposed LEAML achieves the best label distribution recovery performance, and the LDML-R achieves the second best performance.

4.1.5. Computational complexity comparison

To compare the computational complexity imposed by the 8 algorithms in recovering the label distributions, we record the runtimes of these algorithms to complete the LE learning task for each dataset measured in second [s]. The runtime experimental results are listed in Table 9. It can be seen that the MDLRML algorithm is the clear winner, in terms of computational complexity. The proposed LEAML algorithm has the second best runtime performance, slightly lower than those of the PLEA and LDML-R algorithms.

4.1.6. Statistical validation of label distribution recovery performance

Friedman test statistically compares relative performance among multiple algorithms over multiple datasets [34]. We use this test to validate the statistical significance of the performance of various algorithms given in Table 3,4,5,6, Table 8. Table 10 lists the Friedman statistics F_F for the label distribution recovery performance of Table 3,4,5,6, Table 8, with the critical value at a significance level of 0.05, among the 8 comparing algorithms and 14

Table 5
Label distribution recovery performance measured by Canberra metric ↓.

Dataset	LP	ML	GLLE	LESC	MDLRML	PLEA	LDML-R	LEAML
SJAFFE	1.0708(7)	1.6894(8)	0.7518(6)	0.5606(5)	0.4796(4)	0.3720(1)	0.3822(2)	0.3989(3)
Natural_scene	6.7810(6)	6.7217(5)	6.8511(7)	6.8780(8)	5.3370(3)	4.9419(1)	5.0009(2)	5.3472(4)
SBU_3DFE	1.2463(7)	1.6593(8)	0.8409(6)	0.8039(5)	0.5488(3)	0.4815(1)	0.6410(4)	0.5034(2)
Yeast_spoem	0.3655(8)	0.2800(7)	0.1840(6)	0.1801(5)	0.0192(3)	0.0198(4)	0.0093(1)	0.0152(2)
Yeast_alpha	1.7068(7)	2.0181(8)	1.1135(6)	0.9514(5)	0.3328(1)	0.8726(4)	0.4740(3)	0.3677(2)
Yeast_cdc	1.3532(7)	1.7591(8)	0.9442(6)	0.8405(5)	0.3179(2)	0.3610(4)	0.3219(3)	0.2470(1)
Yeast_cold	0.3241(7)	0.5598(8)	0.3016(6)	0.2680(5)	0.0707(3)	0.0723(4)	0.0665(2)	0.0653(1)
Yeast_diau	0.6425(6)	1.6538(8)	0.6734(7)	0.5021(5)	0.1637(2)	0.2429(4)	0.1991(3)	0.1134(1)
Yeast_dtt	0.3560(7)	0.5070(8)	0.2458(6)	0.2229(5)	0.0319(3)	0.0453(4)	0.0277(1)	0.0307(2)
Yeast_elu	1.2612(7)	1.6263(8)	0.8692(6)	0.7906(5)	0.3028(2)	0.3189(4)	0.3167(3)	0.3021(1)
Yeast_heat	0.4706(7)	0.7826(8)	0.4203(6)	0.4110(5)	0.1124(1)	0.3349(4)	0.1181(3)	0.1174(2)
Yeast_spo	1.2341(8)	0.8440(7)	0.5422(6)	0.5329(5)	0.2344(2)	0.3379(4)	0.2365(3)	0.1578(1)
Yeast_spo5	0.4013(7)	0.4664(8)	0.3018(6)	0.2884(5)	0.1623(4)	0.1590(3)	0.0886(2)	0.0324(1)
Movie	0.9260(4)	1.4409(8)	1.0372(6)	1.1474(7)	0.7995(3)	0.6933(2)	0.9982(5)	0.6369(1)
Avg.Rank	6.7857(7)	7.6429(8)	6.1429(6)	5.3571(5)	2.5714(2)	3.1429(4)	2.6429(3)	1.7143(1)

Table 6
Label distribution recovery performance measured by Kullback-Leibler divergence ↓.

Dataset	LP	ML	GLLE	LESC	MDLRML	PLEA	LDML-R	LEAML
SJAFFE	0.0770(7)	0.2513(8)	0.0500(6)	0.0290(5)	0.0200(4)	0.0195(3)	0.0115(2)	0.0109(1)
Natural_scene	1.5950(6)	2.2757(7)	2.6630(8)	1.1663(5)	0.5689(4)	0.3268(1)	0.3367(2)	0.5620(3)
SBU_3DFE	0.1050(7)	0.2489(8)	0.0690(5)	0.0692(6)	0.0250(3)	0.0217(1)	0.0326(4)	0.0232(2)
Yeast_spoem	0.0670(7)	0.5030(8)	0.0270(5.5)	0.0270(5.5)	0.0001(3.0)	0.0001(3)	0.00003(1)	0.0001(3.0)
Yeast_alpha	0.1210(8)	0.0550(7)	0.0130(6)	0.0080(5)	0.0012(2)	0.0075(4)	0.0044(3)	0.0011(1)
Yeast_cdc	0.1110(8)	0.0609(7)	0.0140(6)	0.0100(5)	0.0014(2)	0.0018(3.5)	0.0018(3.5)	0.0010(1)
Yeast_cold	0.1030(7)	0.5560(8)	0.0190(6)	0.0150(5)	0.0018(4)	0.0010(3)	0.0007(1)	0.0008(2)
Yeast_diau	0.1270(7)	0.1934(8)	0.0270(6)	0.0170(5)	0.0022(2)	0.0036(4)	0.0028(3)	0.0017(1)
Yeast_dtt	0.1030(8)	0.0648(7)	0.0130(6)	0.0100(5)	0.0002(3)	0.0002(3)	0.0001(1)	0.0002(3)
Yeast_elu	0.1090(8)	0.0567(7)	0.0130(6)	0.0090(5)	0.0015(2)	0.0022(4)	0.0017(3)	0.0013(1)
Yeast_heat	0.0890(8)	0.0656(7)	0.0170(6)	0.0155(5)	0.0090(4)	0.0074(3)	0.0010(1.5)	0.0010(1.5)
Yeast_spo	0.0840(7)	0.5320(8)	0.0290(6)	0.0280(5)	0.0060(3)	0.0077(4)	0.0049(2)	0.0019(1)
Yeast_spo5	0.0420(7)	0.0811(8)	0.0340(6)	0.0310(5)	0.0066(3)	0.0078(4)	0.0058(2)	0.0005(1)
Movie	0.1358(6)	0.1268(5)	0.2239(7)	0.2310(8)	0.0756(4)	0.0419(1)	0.0736(3)	0.0587(2)
Avg.Rank	7.2143(7)	7.3571(8)	6.1071(6)	5.3214(5)	3.0714(4)	2.9643(3)	2.2857(2)	1.6786(1)

Table 7
Label distribution recovery performance measured by cosine coefficient ↑.

Dataset	LP	ML	GLLE	LESC	MDLRML	PLEA	LDML-R	LEAML
SJAFFE	0.9410(7)	0.8231(8)	0.9594(6)	0.9731(5)	0.9797(4)	0.9901(1)	0.9890(2)	0.9888(3)
Natural_scene	0.7264(7)	0.6610(8)	0.7789(3)	0.7602(4)	0.7555(5)	0.8920(1)	0.8905(2)	0.7446(6)
SBU_3DFE	0.9220(7)	0.8435(8)	0.9304(6)	0.9319(5)	0.9740(3)	0.9773(1)	0.9657(4)	0.9768(2)
Yeast_spoem	0.9503(7)	0.8530(8)	0.9780(5.5)	0.9780(5.5)	0.9998(3)	0.9998(3)	0.9999(1)	0.9998(3)
Yeast_alpha	0.9814(7)	0.9530(8)	0.9876(6)	0.9905(5)	0.9988(2)	0.9928(4)	0.9943(3)	0.9989(1)
Yeast_cdc	0.9828(7)	0.9468(8)	0.9875(6)	0.9896(5)	0.9982(3)	0.9982(3)	0.9982(3)	0.9989(1)
Yeast_cold	0.9847(6)	0.9429(8)	0.9827(7)	0.9859(5)	0.9991(3)	0.9990(4)	0.9992(2)	0.9993(1)
Yeast_diau	0.9805(6)	0.8427(8)	0.9750(7)	0.9844(5)	0.9978(2)	0.9963(4)	0.9973(3)	0.9995(1)
Yeast_dtt	0.9835(7)	0.9515(8)	0.9884(6)	0.9901(5)	0.9998(2)	0.9986(4)	0.9999(1)	0.9997(3)
Yeast_elu	0.9829(7)	0.9489(8)	0.9879(6)	0.9896(5)	0.9985(2)	0.9978(4)	0.9984(3)	0.9986(1)
Yeast_heat	0.9861(5)	0.9454(8)	0.9845(7)	0.9851(6)	0.9978(3)	0.9930(4)	0.9990(1)	0.9989(2)
Yeast_spo	0.9386(7)	0.8397(8)	0.9747(5.5)	0.9747(5.5)	0.9950(3)	0.9920(4)	0.9951(2)	0.9981(1)
Yeast_spo5	0.9686(7)	0.9359(8)	0.9713(6)	0.9732(5)	0.9935(3)	0.9927(4)	0.9943(2)	0.9995(1)
Movie	0.9589(4)	0.8765(8)	0.9369(5)	0.9200(7)	0.9301(6)	0.9750(2)	0.9723(3)	0.9860(1)
Avg.Rank	6.5000(7)	8.0000(8)	5.8571(6)	5.2143(5)	3.1429(4)	3.0714(3)	2.2857(2)	1.9286(1)

datasets. As can be seen from Table 10, at 0.05 significance level, all the F_F values for the six metrics are greater than the critical value, and therefore the null hypothesis of indistinguishable performance among the learning approaches is clearly rejected for all the evaluation metrics. In other words, the average performance rankings for the 8 algorithms given in Table 3,4,5,6, Table 8 are statistically significant.

Bayesian signed-rank test [35] is employed as the statistical test to show whether the LEAML performs significantly better than the other LE algorithms, in terms of each evaluation metric. Table 11 summarizes the statistical test results, where the associated probabilities for the corresponding tests are given respectively in the brackets. Specifically, a, b and c in [a, b, c] respectively represent the probabilities of [WIN, TIE, LOSE]. The prior default is that the performance of the two algorithms is the same. Prior strength is the strength of this null hypothesis, which means that this null hypothesis is established with a probability of 0.6. The performance of two algorithms are similar if the difference between two algorithms' results is less than rope $= 0.01$. The test results of Table 11 clearly validate the superior performance of our LEAML over the existing state-of-art LE algorithms, in terms of LE learning accuracy.

Table 8
Label distribution recovery performance measured by intersection similarity ↑.

Dataset	LP	ML	GLLE	LESC	MDLRML	PLEA	LDML-R	LEAML
SJAFFE	0.8361(7)	0.7251(8)	0.8757(6)	0.9050(5)	0.9141(4)	0.9355(1)	0.9344(2)	0.9320(3)
Natural_scene	0.4512(7)	0.3307(8)	0.5226(5)	0.5107(6)	0.5751(4)	0.7278(2)	0.7257(3)	0.7561(1)
SBU_3DFE	0.8096(7)	0.7414(8)	0.8531(6)	0.8542(5)	0.9047(3)	0.9159(1)	0.8885(4)	0.9148(2)
Yeast_spoem	0.8367(7)	0.7681(8)	0.9109(6)	0.9130(5)	0.9904(3)	0.9901(4)	0.9954(2)	0.9955(1)
Yeast_alpha	0.9074(7)	0.8898(8)	0.9386(6)	0.9473(5)	0.9815(1)	0.9519(4)	0.9710(3)	0.9796(2)
Yeast_cdc	0.9122(7)	0.8836(8)	0.9376(6)	0.9445(5)	0.9786(3)	0.9758(4)	0.9787(2)	0.9834(1)
Yeast_cold	0.9213(7)	0.8646(8)	0.9250(6)	0.9338(5)	0.9826(3)	0.9820(4)	0.9836(2)	0.9880(1)
Yeast_diau	0.9128(6)	0.7557(8)	0.9052(7)	0.9301(5)	0.9771(2)	0.9648(4)	0.9720(3)	0.9858(1)
Yeast_dtt	0.9134(7)	0.8779(8)	0.9393(6)	0.9448(5)	0.9920(3)	0.9887(4)	0.9930(2)	0.9933(1)
Yeast_elu	0.9120(7)	0.8839(8)	0.9383(6)	0.9439(5)	0.9783(2)	0.9771(4)	0.9773(3)	0.9851(1)
Yeast_heat	0.9237(7)	0.8718(8)	0.9310(6)	0.9324(5)	0.9812(2)	0.9451(4)	0.9803(3)	0.9866(1)
Yeast_spo	0.8184(7)	0.7614(8)	0.9105(6)	0.9121(5)	0.9610(3)	0.9428(4)	0.9906(1)	0.9738(2)
Yeast_spo5	0.8855(7)	0.7486(8)	0.9020(6)	0.9067(5)	0.9469(4)	0.9494(3)	0.9700(2)	0.9838(1)
Movie	0.5848(7)	0.5509(8)	0.5872(6)	0.5953(5)	0.8406(4)	0.8881(2)	0.8501(3)	0.8889(1)
Avg.Rank	6.9286(7)	8.0000(8)	6.0000(6)	5.0714(5)	2.9286(3)	3.2143(4)	2.5000(2)	1.3571(1)

Table 9
Computational complexity of 8 LE algorithms imposed on LE learning of 14 datasets with ground-truth label distributions measured by runtime [s] ↓.

Algorithms	LP	ML	GLLE	LESC	MDLRML	PLEA	LDML-R	LEAML
Yeast-alpha	96.7089(7)	10.4143(3)	2101.2288(8)	0.6749(1)	3.3217(2)	21.2857(6)	19.0666(5)	15.7086(4)
Yeast-cdc	91.9976(6)	28.0560(5)	2058.3899(7)	2620.3463(8)	2.6802(1)	16.2703(3)	18.1193(4)	12.1933(2)
Yeast-cold	85.7640(6)	25.9546(5)	2098.2072(7)	2592.1138(8)	0.6458(1)	16.4845(3)	17.2280(4)	12.9326(2)
Yeast-diau	90.4889(6)	22.2115(5)	2164.7457(7)	2587.7626(8)	2.3170(1)	17.9098(4)	17.2450(3)	16.5481(2)
Yeast-dtt	85.0445(6)	24.5773(5)	2062.2618(7)	2568.7960(8)	2.5661(1)	17.8935(4)	17.1093(3)	12.8461(2)
Yeast-elu	91.1003(6)	31.2115(5)	1949.0746(7)	2244.6699(8)	3.1854(1)	17.3791(3)	18.2103(4)	12.2265(2)
Yeast-heat	88.0539(6)	33.0071(5)	2170.0425(7)	2324.9983(8)	1.8472(1)	17.9099(4)	17.2924(3)	14.5135(2)
Yeast-spo	88.1222(6)	21.0123(5)	2113.8297(7)	2491.1317(8)	0.6969(1)	17.8956(4)	17.1597(3)	14.8390(2)
Yeast-spo5	87.4979(6)	23.8147(5)	2234.4134(7)	2566.9491(8)	2.5439(1)	18.1495(4)	17.2504(3)	12.1442(2)
Yeast-spoem	100.8651(6)	7.5451(2)	2053.6103(7)	2654.3258(8)	0.7449(1)	17.6775(5)	17.0937(4)	16.4845(3)
Natural Scene	71.9972(3)	15.5970(2)	1098.1414(7)	2654.3258(8)	6.4686(1)	157.9247(4)	167.5837(5)	186.6003(6)
Movie	2223.9369(6)	340.6105(3)	59379.0917(8)	23081.6436(7)	251.4561(2)	216.8764(1)	446.2196(4)	448.2214(5)
SJAFFE	0.6692(2)	2.2597(3)	49.4767(8)	40.8202(7)	0.0251(1)	3.2242(4)	5.9159(5)	24.0298(6)
SBU_3DFE	109.8430(6)	13.2224(3)	2068.8012(7)	2726.1515(8)	0.8892(1)	23.9204(4)	8.8391(2)	24.0299(5)
Average rank	5.5714(6)	4.0000(5)	7.2143(7)	7.3571(8)	1.1429(1)	3.7857(4)	3.7143(3)	3.2143(2)

Table 10
Friedman statistics F_F for the label distribution recovery results in terms of each evaluation metric, with the critical value at a significance level of 0.05 (comparing algorithms: 8, datasets: 14) .

Evaluation metric	F_F	Critical value
Chebyshev distance	60.1633	2.112
Clark distance	75.1611	
Canberra distance	68.5722	
Kullback-Leibler divergence	78.7523	
cosine coefficient	55.2440	
intersectional similarity	153.6761	

4.2. Multilabel classification experiments

Having establishing that statistically, the LE learning accuracy of the proposed LEAML algorithm is better than the existing state-of-the-art LE algorithms, we next evaluate the multilabel classification capability of our LEAML.

4.2.1. Multilabel datasets, MLL metrics and MLL benchmarks

For this set of multilabel classification experiments, we employ the 10 real-word multilabel datasets without ground-truth label distributions from [36]. These 10 datasets are summarized in Table 12, where we have S: the number of examples, T: the number of testing samples, $dim(S)$: the feature dimensions, $L(S)$: the number of class labels, $LCard(S)$: the label cardinality, $LDen(S)$: the label density, $DL(S)$: the distinct label sets, and $F(S)$: the feature type.

Table 11
Bayesian signed-rank test on the label distribution recovery performance among 8 algorithms in terms of six evaluation metrics (rope = 0.01, and default prior strength is 0.6) .

LEAML versus	Evaluation metric					
	Chebyshev	Clark	Canberra	KL divergence	cosine	Inters
LP	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]
ML	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]
GLLE	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[0.99588,0.0,0.00412]	[1.0,0.0,0.0]
LESC	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[1.0,0.0,0.0]	[0.99942,0.0,0.00058]	[1.0,0.0,0.0]
MDLRML	[0.99822,4e-05,0.00174]	[0.99924,0.0,0.00076]	[0.99466,0.0,0.00534]	[0.99986,0.00014,0.0]	[0.98294,0.00452,0.01254]	[1.0,0.0,0.0]
PLEA	[0.7511,0.0,0.2489]	[0.99484,0.0,0.00516]	[0.97136,0.0,0.02864]	[0.8643,0.00038,0.13532]	[0.97044,0.0,0.02956]	[0.99998,0.0,2e-05]
LDML-R	[0.37264,0.0,0.62736]	[0.99634,0.0,0.00366]	[0.95998,0.0,0.04002]	[0.98192,0.00114,0.01694]	[0.95404,0.0057,0.04026]	[0.99616,0.0,0.00384]

Table 12
Characteristics of 10 real-world datasets from [36] with unknown ground-truth label distributions used in multilabel classification experiments with MLL metrics .

Dataset	S	T	$dim(S)$	$L(S)$	$LCard(S)$	$LDen(S)$	$DL(S)$	$F(S)$
Emotions	415	178	72	6	1.869	0.311	27	numeric
Medical	645	333	1449	45	1.245	0.028	94	nominal
Cal500	250	252	68	174	26.044	0.150	502	numeric
Birds	320	325	260	19	1.014	0.053	133	numeric
Enron	1123	579	1001	53	3.378	0.064	753	nominal
Yeast	1200	1217	103	14	4.237	0.303	198	numeric
Image	1000	1000	294	5	1.236	0.247	20	numeric
Scene	1211	1196	294	6	1.074	0.179	15	numeric
Corel5k	2500	2500	499	374	3.522	0.009	3175	nominal
Bibtex	3700	3695	1836	159	2.402	0.015	2856	nominal

Table 13
MLL performance comparison of 9 algorithms on 10 real-world datasets of Table 12.

	Yeast	Emotions	Medical	Cal500	Birds	Image	Scene	Enron	Corel5k	Bibtex
Algorithm					Hamming Loss↓					
ML	0.2073	0.2388	**0.0114**	0.1578	0.0636	0.1642	**0.0847**	**0.0546**	0.0098	0.0126
ML-kNN	0.1980	0.2706	0.0153	0.1416	0.0546	0.1862	0.0989	0.0620	**0.0094**	0.0136
MLNB	0.2166	0.2804	0.0339	**0.1395**	0.0779	0.2300	0.1299	0.1145	0.0145	0.0824
MLFE	0.2038	0.2434	**0.0112**	0.1549	0.0615	**0.1616**	0.0903	**0.0543**	0.0101	**0.0124**
BP-MLL	0.4500	0.2987	0.0290	0.1472	0.0683	0.3056	0.2904	0.0682	**0.0094**	0.0160
MDLRML	**0.1910**	**0.2247**	0.0116	**0.1412**	**0.0514**	0.1644	**0.0872**	0.0570	0.0156	**0.0094**
PLEA	0.1945	0.2406	0.0115	0.1596	0.0645	0.1654	**0.0847**	**0.0546**	0.0098	0.0126
LDML-R	0.1950	**0.2350**	0.0277	0.1489	**0.0510**	0.2484	0.1809	0.0668	**0.0092**	0.0125
LEAML	**0.1914**	0.2491	0.2440	0.1413	0.0552	**0.1610**	**0.0872**	0.0554	**0.0094**	0.0127
Algorithm					ranking loss↓					
ML	0.3022	0.2228	0.1084	0.4721	0.3288	0.1467	0.0580	0.3210	0.4177	**0.0897**
ML-kNN	**0.1715**	0.2724	**0.0540**	0.1928	0.3070	0.1927	0.0931	**0.1220**	**0.2663**	0.2234
MLNB	0.2323	0.2150	0.0599	**0.1927**	**0.2157**	0.2420	0.1124	0.1768	**0.1267**	0.1584
MLFE	**0.1777**	**0.2061**	**0.0209**	0.2089	0.3210	**0.1443**	0.0713	**0.0958**	0.3156	0.0914
BP-MLL	0.4450	0.4803	0.2445	0.1996	0.3964	0.7956	0.5992	0.3738	0.2695	0.4764
MDLRML	0.2937	**0.1812**	0.1440	**0.1769**	**0.2916**	**0.1352**	**0.0577**	0.3076	0.4303	0.1017
PLEA	0.2904	0.2166	0.1093	0.4749	0.3865	0.1456	**0.0575**	0.3229	0.4439	**0.0897**
LDML-R	0.3038	0.2345	0.4970	0.4836	0.4858	0.4623	0.4766	0.3124	0.4982	0.4994
LEAML	0.2877	0.2222	0.1435	0.4654	0.2922	**0.1352**	0.0580	0.2951	0.4268	**0.0886**
Algorithm					one error↓					
ML	0.2857	0.5000	0.3421	0.0805	0.7895	**0.2000**	**0.0000**	0.6731	0.9360	0.3899
ML-kNN	0.2345	0.4213	0.2492	0.1190	0.7356	0.3600	0.2425	0.3921	0.7892	0.6225
MLNB	0.4170	0.4848	0.4234	0.1190	0.5517	0.4390	0.2851	0.5233	0.8804	0.5876
MLFE	0.2356	0.3708	0.1471	0.1984	0.7471	0.2680	0.2157	0.2608	0.7832	0.3710
BP-MLL	0.7034	0.7022	0.4024	0.1071	0.7989	0.6710	0.8269	0.2642	0.9716	0.4547
MDLRML	**0.0714**	**0.3333**	0.3684	**0.0747**	**0.3396**	**0.2000**	0.3000	0.0639	0.7680	0.3322
PLEA	**0.1429**	0.5000	0.3421	**0.0776**	0.4474	**0.0000**	**0.0000**	0.6923	0.9302	0.3459
LDML-R	0.2857	0.5000	**0.1421**	0.1494	**0.0526**	**0.2000**	**0.1667**	**0.0566**	**0.0116**	**0.0063**
LEAML	**0.0714**	**0.1502**	**0.0000**	0.1034	**0.0526**	**0.0000**	**0.0000**	**0.0385**	**0.0000**	**0.0000**
Algorithm					coverage↓					
ML	0.8749	0.1600	0.5236	0.2302	0.2836	0.9510	0.9282	0.4523	0.1813	0.2472
ML-kNN	**0.6414**	0.2247	0.3441	**0.1319**	0.3606	1.0420	0.5686	**0.1631**	0.1978	0.5723
MLNB	**0.2499**	0.2871	0.1925	**0.1346**	0.2695	1.2450	0.6564	0.2313	0.2102	0.3819
MLFE	0.6503	0.1887	0.1475	0.1354	0.3763	**0.8410**	0.4582	**0.1495**	0.2238	0.2586
BP-MLL	0.8990	0.3089	0.2955	1.3386	0.4415	2.1460	2.0761	0.2369	0.1980	0.7356
MDLRML	0.8620	**0.1480**	0.0590	0.2270	**0.2660**	0.9350	**0.1870**	0.4390	0.1830	**0.1980**
PLEA	0.8816	0.1570	**0.0520**	0.2281	0.2828	0.9512	0.9330	0.4579	**0.1806**	0.2457
LDML-R	0.8629	0.1723	0.3398	0.2302	0.2704	0.9608	1.0690	0.7529	0.1866	0.3477
LEAML	0.8636	**0.1502**	**0.0340**	0.2276	**0.2649**	**0.9030**	**0.1199**	0.4329	**0.1748**	**0.2438**
Algorithm					average precision↑					
ML	**0.8228**	0.7764	**0.9806**	0.7758	0.6430	0.8555	**0.9329**	0.9056	0.7059	**0.9261**
ML-kNN	0.6642	0.7142	0.7695	0.5054	0.6173	0.8187	0.9108	0.5512	0.5233	0.6528
MLNB	0.6936	0.6807	0.5227	0.5120	0.6955	0.7788	0.8993	0.5569	0.3559	0.8209
MLFE	0.6996	**0.7901**	0.8745	0.5377	0.7047	**0.8617**	**0.9385**	0.6581	0.5549	0.8672
BP-MLL	0.4297	0.5161	0.2081	0.4783	0.2460	0.5111	0.4200	0.2057	0.2012	0.0659
MDLRML	**0.8316**	**0.7831**	0.5748	0.7883	0.6841	0.7258	0.8394	0.2130	0.3470	0.3637
PLEA	0.5085	0.6217	0.5783	0.1815	0.1255	0.7073	0.8407	0.1890	0.0339	0.3871
LDML-R	0.6910	0.6743	**0.9597**	**0.8416**	**0.9348**	0.7271	0.7892	**0.9217**	**0.9873**	**0.9824**
LEAML	0.5802	0.6515	**0.9597**	**0.8363**	**0.8849**	**0.9277**	0.8395	**0.9168**	**0.9652**	0.8821

To evaluate multilabel classification performance, we choose the five widely used MLL metrics [37], and they are: Hamming loss ↓, ranking loss ↓, one error ↓, coverage ↓, and average precision ↑.

The benchmark algorithms used in this set of multilabel classification experiments include our previous MDLRML [21], PLEA [22] and LDML-R [23] as well as five existing state-of-the-art MLL algorithms, specifically, the BP-MLL [38], the ML [15], the ML-kNN [39], the MLNB [40], and MLFE the [41].

4.2.2. *Multilabel classification performance*

The experimental results are listed in Table 13, where bold black number indicates the best performance and bold blue num-

Table 14
Bayesian signed-rank test on the multilabel classification performance among the 9 algorithms in terms of five evaluation metrics (rope = 0.01, and default prior strength is 0.6) .

LEAML versus	Evaluation metric				
	Hamming loss↓	ranking loss↓	one error↓	coverage↓	average precision↑
ML	[0.62732, 0.0007, 0.37198]	[0.92448, 0.0003, 0.07522]	[0.99968, 0.00016, 0.00016]	[1.0, 0.0, 0.0]	[0.51918, 0.0, 0.48082]
ML-kNN	[0.93258, 0.0003, 0.06712]	[0.14964, 4e-05, 0.85032]	[1.0, 0.0, 0.0]	[0.8928, 2e-05, 0.10718]	[0.99456, 2e-05, 0.00542]
MLNB	[0.95454, 2e-05, 0.04544]	[0.05134, 2e-05, 0.94864]	[1.0, 0.0, 0.0]	[0.79686, 0.0, 0.20314]	[0.99228, 0.0, 0.00772]
MLFE	[0.75654, 0.0, 0.24346]	[0.038, 2e-05, 0.96198]	[1.0, 0.0, 0.0]	[0.57846, 2e-05, 0.42152]	[0.91942, 0.0, 0.08058]
BP-MLL	[0.97216, 0.00018, 0.02766]	[0.9674, 4e-05, 0.03256]	[1.0, 0.0, 0.0]	[0.99764, 0.0, 0.00236]	[1.0, 0.0, 0.0]
MDLRML	[0.16516, 0.00216, 0.83268]	[0.5875, 0.00088, 0.41162]	[0.99932, 0.00012, 0.00056]	[0.86442, 0.0, 0.13558]	[0.98304, 0.00032, 0.01664]
PLEA	[0.63194, 0.0008, 0.36726]	[0.92612, 2e-05, 0.07386]	[0.99772, 0.0015, 0.00078]	[1.0, 0.0, 0.0]	[0.99996, 0.0, 4e-05]
LDML-R	[0.64654, 0.0001, 0.35336]	[1.0, 0.0, 0.0]	[0.99998, 2e-05, 0.0]	[0.99994, 0.0, 6e-05]	[0.14304, 0.00036, 0.8566]

ber indicates the second best performance. The top four ranking algorithms are as follows. Our LEAML achieves the best performance in 17 cases and the second best performance in 11 cases. The MDLRML attains the best performance in 10 cases and the second best performance in 11 cases. The LDML-R has 8 cases of the best performance and 8 cases of the second best performance. The fourth ranking MLFE achieves the best performance in 8 cases and the second best performance in 6 cases.

4.2.3. Statistical validation of multilabel classification performance

To test the statistical relationship between LEAML and the other algorithms, the results of Bayesian signed-rank test [35] are given in Table 14. It can be seen that the LEAML algorithm wins over the ML, BP-MLL and PLEA in all the five metrics, while it wins over ML-kNN, MLBN, MLFE, MDLRML and LDML-R in four metrics but loses to each of these competitors in one metric. The results of Bayesian signed-rank test thus suggest that our LEAML has statistically significant advantages over these benchmarks in MLL performance.

5. Conclusions

In this paper, we have proposed a novel label enhancement manifold learning algorithm to solve the multi-label learning problem. Our proposed LEAML consists of three interconnected components. First we excavate the underlying label information contained in the feature space through an incremental semi-supervised subspace learning. Then we estimate the label distribution via label propagation. Afterward we use the estimated label distributions and extracted features to train the label distribution prediction model via the conditional random field, and the maximum likelihood estimation of the label distribution predictor's parameters are obtained based on a gradient-descent iterative optimization algorithm. Extensive experimental results involving 14 real-word multilabel datasets with the ground-truth label distributions have convincingly demonstrated the superior label distribution recovery performance of our proposed LEAML algorithm over the well-established state-of-the-art LE algorithms. Experimental results involving 10 real-life datasets without ground-truth label distributions have demonstrated the excellent multilabel classification performance of our LEAML algorithm compared with the state-of-the-art MLL algorithms.

Declaration of competing interest

The authors have no conflict of interest to declare.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

Data will be made available on request.

Acknowledgment

This work was supported by National Natural Science Foundation of China under Grant 61702270, 41971343 and 62076063. Dr Tan would like to thank the sponsorship of Chinese Scholarship Council for funding her research at School of Electronics and Computer Science, University of Southampton, UK.

References

[1] X. Geng, L. Luo, Multilabel Ranking with Inconsistent Rankers, in: Proc. CVPR 2014, 2014, pp. 3742–3747. Jun.23–28. Columbus, OH, USA
[2] W.-J. Zhou, Y. Yu, M.L. Zhang, Binary Linear Compression for Multi-label Classification, in: Proc. IJCAI 2017, 2017, pp. 3546–3552. Aug. 19–25.Melbourne, Australia
[3] J.-Y. Weng, Y.-L. Zhang, W.-S. Hwang, Candid covariance-free incremental principal component analysis, IEEE Trans. Pattern Anal. Mach. Intell. 25 (8) (2003) 1034–1040.
[4] X. Geng, C. Yin, Z.H. Zhou, Facial age estimation by learning from label distributions, IEEE Trans. Pattern Anal. Mach. Intell. 35 (10) (2013) 2401–2412.
[5] M.-L. Zhang, J.P. Fang, Partial multi-label learning via credible label elicitation, IEEE Trans. Pattern Anal. Mach. Intell. 43 (10) (2021) 3587–3599.
[6] G. Tsoumakas, I. KatakisI, Multi-label classification: an overview, Int. J. Data Warehous. Mining 3 (3) (2007) 1–13.
[7] X. Geng, Label distribution learning, IEEE Trans. Knowl. Data Eng. 28 (7) (2016) 1734–1748.
[8] C. Hong, Multimodal face-pose estimation with multitask manifold deep learning, IEEE Trans. Ind. Inform. 15 (7) (2019) 3952–3961.
[9] J. Yu, D. Tao, M. Wang, Y. Rui, Learning to rank using user clicks and visual features for image retrieval, IEEE Trans. Cybernetics 45 (4) (2015) 767–779.
[10] J. Yu, Hierarchical deep click feature prediction for fine-grained image recognition, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2) (2022) 563–578.
[11] C. Hong, Multimodal deep autoencoder for human pose recovery, IEEE Trans. Image Process. 24 (12) (2015) 5659–5670.
[12] C. Hong, J. Yu, X. Chen, Image-based 3D Human Pose Recovery with Locality Sensitive Sparse Retrieval, in: Proc. 2013 IEEE Int. Conf. Systems, Man, and Cybernetics, 2013, pp. 2103–2108. (Manchester, UK), Oct. 13-16
[13] M.L. Zhang, Leveraging implicit relative labeling-importance information for effective multi-label learning, IEEE Trans. Knowl. Data Eng. 33 (5) (2021) 2057–2070.
[14] F.-P. Nie, D. Xu, I.-W. Tsang, C.S. Zhang, Flexible manifold embedding: a framework for semi-supervised and unsupervised dimension reduction, IEEE Trans. Image Process. 19 (7) (2010) 1921–1932.
[15] P. Hou, X. Geng, M.L. Zhang, Multi-label Manifold Learning, in: Proc. AAAI 2016, 2016, pp. 1680–1686. (Phoenix, AZ, USA), Feb. 12-17
[16] X.J. Zhu, Semi-supervised Learning with Graphs, Language Technologies Institute, School of Computer Science, 2005. Carnegie Mellon University
[17] S.-T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[18] N. Xu, A. Tao, X. Geng, Label Enhancement for Label Distribution Learning, in: Proc. IJCAI 2018, 2018, pp. 2926–2932. (Stockholm, Sweden), Jul. 13-19
[19] H.Y. Tang, Label Enhancement with Sample Correlations via Low-rank Representation, in: Proc. AAAI 2020, 2020, pp. 5932–5939. (New York, USA), Feb. 7-12
[20] J. Nocedal, S.J. Wright, Numerical Optimization (2006). Springer, New York
[21] C. Tan, S. Chen, G.-L. Ji, X. Geng, Multilabel distribution learning based on multioutput regression and manifold learning, IEEE Trans. Cybernetics 52 (6) (2022) 5064–5078.

[22] C. Tan, S. Chen, G.-L. Ji, X. Geng, A novel probabilistic label enhancement algorithm for multi-label distribution learning, IEEE Trans. Knowl. Data Eng. 34 (11) (2022) 5098–5113.
[23] C. Tan, S. Chen, X. Geng, G.L. Ji, A label distribution manifold learning algorithm, Pattern Recognit. (2022), doi:10.1016/j.patcog.2022.109112. Early access, Oct
[24] R. Shao, N. Xu, X. Geng, Multi-label Learning with Label Enhancement, in: Proc. ICDM 2018, 2018, pp. 437–446. (Singapore), Nov. 17-20
[25] M. Fisz, Probability theory and mathematical statistics (3rd edition)963. John Wiley and Sons, New York
[26] D.Y. Zhou, Learning with Local and Global Consistency, in: Proc. NIPS 2003, 2003, pp. 321–328. (Vancouver, BC, Canada), Dec. 8-13
[27] J. Lafferty, A. Mccallum, F.C.N. Pereira, Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data, in: Proc. ICML 2001, Williamstown, MA, USA, 2001, pp. 282–289. Jun. 28-Jul. 1
[28] H.M. Wallach, Conditional random fields: an introduction, Technical Report No.MS-CIS-04-21 Department of Computer and Information Science (2004). University of Pennsylvania
[29] C. Sutton, A. McCallum, An Introduction to Conditional Random Fields for Relational Learning, in: L. Getoor, B. Taskar (Eds.), Introduction to Statistical Relational Learning, MIT Press, 2007, pp. 93–127.
[30] S.-D. Pietra, V.-J.-D. Pietra, J.D. Lafferty, Inducing features of random fields, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 380–393.
[31] Z.Y. Zhang, H.Y. Zha, Principal manifolds and nonlinear dimensionality reduction via tangent space alignment, SIAM J. Sci. Comput. 26 (1) (2004) 313–338.
[32] F. Párez-Cruz, A. Navia-Vázquez, P.L. Alarcón-Diana, A. Artés-Rodríguez, An IRWLS Procedure for SVR, in: Proc. 10th European Signal Processing Conf., Tampere, Finland, 2000, pp. 1–4. Sep. 4-8
[33] S.H. Cha, Comprehensive survey on distance/similarity measures between probability density functions, Int. J. Math. Models Method. Appl. Sci. 1 (4) (2007) 300–307.
[34] J. Demsăr, Statistical comparisons of classifiers over multiple datasets, J. Mach. Learn. Res. 7 (1) (2006) 1–30.
[35] A. Benavoli, G. Corani, J. Demsăr, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, J. Mach. Learn. Res. 18 (77) (2017) 1–36.
[36] Mulan: A java library for multi-label learning. http://www.mulan.sourceforge.net/datasets-mlc.html,2018-03-01.
[37] M.-L. Zhang, Z.H. Zhou, A review on multi-label learning algorithms, IEEE Trans. Knowl. Data Eng. 26 (8) (2014) 1819–1837.
[38] M.-L. Zhang, Z.H. Zhou, Multilabel neural networks with applications to functional genomics and text categorization, IEEE Trans. Knowl. Data Eng. 18 (10) (2006) 1338–1351.
[39] M.-L. Zhang, Z.H. Zhou, ML-KNN: a lazy learning approach to multi-label learning, Pattern Recognit. 40 (7) (2007) 2038–2048.
[40] M.-L. Zhang, J.M. Peña, V. Robles, Feature selection for multi-label naive bayes classification, Inf. Sci. (Ny) 179 (19) (2009) 3218–3229.
[41] Q.-W. Zhang, Y. Zhong, M.L. Zhang, Feature-induced Labeling Information Enrichment for Multi-label Learning, in: Proc. AAAI 2018, New Orleans, LA, USA, 2018, pp. 4446–4453. Feb. 2-7

1. Chao Tan received the B.E. and M.E. degree in Computer Science and Technology from Southeast University in 2005 and 2009, respectively, and received the PhD degree in Computer Science and Technology from Tongji University in 2015. She joined the Nanjing Normal University as a lec- turer in 2015 and is an associate professor in the School of Computer and Electronic Information/School of Artificial Intelligence at present. She has worked as a postdoctoral researcher in Southeast University. Her re- search interests generally focus on machine learning, multi-label manifold learning and data mining.

Sheng Chen received his BEng degree from the East China Petroleum Institute, Dongying, China, in 1982, and his PhD degree from the City University, London, in 1986, both in control engineering. In 2005, he was awarded the higher doctoral degree, Doctor of Sciences (DSc), from the U- niversity of Southampton, Southampton, UK. From 1986 to 1999, He held research and academic appointments at the Universities of Shefield, Edin- burgh and Portsmouth, all in UK. Since 1999, he has been with the School of Electronics and Computer Science, the University of Southampton, UK, where he holds the post of Professor in Intelligent Systems and Signal Processing. Dr Chen's research interests include neural network and machine learning, adaptive signal processing, wireless communications, modelling and identification of nonlinear systems, evolutionary computation meth- ods and optimisation. He has published over 700 research papers. Dr. Chen is a Fellow of the United Kingdom Royal Academy of Engineer- ing, a Fellow of IEEE, a fellow of IET, a Distinguished Adjunct Professor at King Abdulaziz University, Jeddah, Saudi Arabia, and an original ISI highly cited researcher in engineering (March 2004). Professor Chen has 15,100+ Web of Science citations with h-index 54 and 20,500+ Google Scholar citations with h-index 75.

Xin Geng received the BSc and MSc degrees in Computer Science from Nanjing University, China, in 2001 and 2004, respectively, and the PhD degree from Deakin University, Australia in 2008. He is currently a pro- fessor in the school of Computer Science and Engineering and the dean of the graduate school at Southeast University. His research interests in- clude pattern recognition, machine learning, and computer vision. He has published more than 40 refereed papers and holds four patents in these areas. He is member of the IEEE.

Genlin Ji received the B.E. and M.E. degree in Computer Science and Technology from Nanjing University of Aeronautics and Astronautics in 1986 and 1989, respectively, and received the PhD degree in Computer Science and Technology from Southeast University in 2004. He is now a professor in the School of Computer and Electronic Information/School of Artificial Intelligence at Nanjing Normal University. His research interests generally focus on data mining and its application.