Label Distribution Learning with Label Correlations on Local Samples

Xiuyi Jia, *Member, IEEE*, Zechao Li, *Member, IEEE*, Xiang Zheng, Weiwei Li, and Sheng-Jun Huang, *Member, IEEE*

Abstract—Label distribution learning (LDL) is proposed for solving the label ambiguity problem in recent years, which can be seen as an extension of multi-label learning. To improve the performance of label distribution learning, some existing algorithms exploit label correlations in a global manner that assumes the label correlations are shared by all instances. However, the instances in different groups may share different label correlations, and few label correlations are globally applicable in real-world tasks. In this paper, two novel label distribution learning algorithms are proposed by exploiting label correlations on local samples, which are called GD-LDL-SCL and Adam-LDL-SCL, respectively. To utilize the label correlations on local samples, the influence of local samples is encoded, and a local correlation vector is designed as the additional features for each instance, which is based on the different clustered local samples. Then, the label distribution for an unseen instance can be predicted by exploiting the original features and the additional features simultaneously. Extensive experiments on some real-world data sets validate that our proposed methods can address the label distribution problems effectively and outperform state-of-the-art methods.

Index Terms—Multi-label learning, label distribution learning, label correlations

1 Introduction

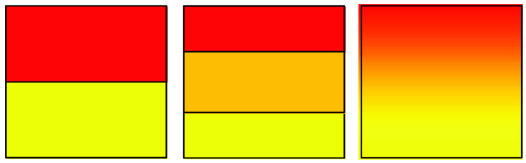
Learning with ambiguity has attracted lots of attention in recent years and has been a hot topic in machine learning area. Single-label learning and multi-label learning [1] are two existing sophisticated paradigms that can solve some label ambiguity problems at present. The former assumes that each instance can be only associated with a single predefined label, whereas the latter assumes that each instance may have a set of class labels. A great deal of research [2], [3], [4] has demonstrated that multi-label learning is a more widely applied paradigm and can address more complex label ambiguity problems. However, multi-label learning can only address the label ambiguity problem of "*what describes the instance*" , i.e., it can only output a set of predefined labels that are associated with the instance. However, the more general ambiguity scenario of "*how to describe the instance*" is needed to address once in a while. For example, in facial expression emotion recognition, it is more meaningful to know the extent of each emotion describes the instance. To solve such problems, a novel paradigm named label distribution learning (LDL) was proposed by Geng [5], which is a natural extension of multi-label learning. Different from traditional multi-label learning to decide the simple 0/1 label belongingness, LDL outputs a soft label belongingness. Moreover, the extent of each label describes the instance is denoted by the corresponding value of a label distribution, which is called the *description degree*.

LDL is a more widely used paradigm that can be applied to more scenarios, and it can address more problems with label ambiguity. Fig. 1 gives the reason why it is the case and shows the decision regions with two labels only. In single-label learning, there are two kinds of outputs, i.e., *red* or *yellow*. In multi-label learning, there are three kinds of outputs, i.e., except for the same two outputs of single-label learning, it can also outputs *red* and *yellow* simultaneously which is denoted by *orange*. However, in LDL, there are infinite outputs because each point in Fig. 1c can be viewed as an output of LDL. In addition, single-label learning and multi-label learning can be viewed as the particular cases of LDL. As shown in Fig. 2, single-label learning can be converted into LDL by setting the description degree of the correct label to 1 while the others are set to 0. For multi-label learning, assuming an instance has two relevant labels, it can be transformed to LDL by setting the description degrees of the two relevant labels to 0.5. Finally, Fig. 2c represents a general case of LDL.

More and more attention is attracted by LDL because LDL can address the more general scenario of label ambiguity, and it can achieve good results by using LDL methods. To name a few, LDL is first proposed to deal with the problem of facial age estimation [6]. Due to the fact that the instances at neighboring ages are similar, it is more desirable to use a

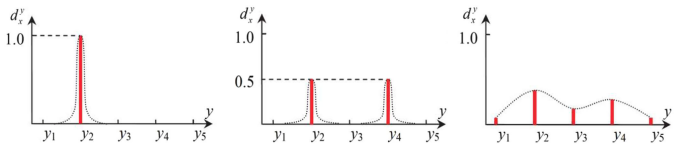
- *X. Jia, Z. Li, and X. Zheng are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210014, China. E-mail: {jiaxy, zechao.li}@njust.edu.cn, zhengxiang_mr@163.com.*
- *W. Li is with the College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China. E-mail: liweiwei@nuaa.edu.cn.*
- *S.-J. Huang is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China. E-mail: huangsj@nuaa.edu.cn.*

Manuscript received 13 Jan. 2019; revised 17 July 2019; accepted 15 Sept. 2019. Date of publication 24 Sept. 2019; date of current version 5 Mar. 2021. (Corresponding author: Zechao Li.)
Recommended for acceptance by B. Moseley.
Digital Object Identifier no. 10.1109/TKDE.2019.2943337



(a) Single-label learning (b) Multi-label learning (c) Label distribution learning

Fig. 1. The decision regions of three learning paradigms for a learning problem with two labels [5].



(a) Single-label learning (b) Multi-label learning (c) Label distribution learning

Fig. 2. Examples label distributions for single-label, multi-label and the general case [5].

distribution across all ages than a single age for a face, thus the IIS-LLD method was proposed [6]. Furthermore, to improve the performance of IIS-LLD, Geng's group also proposed a neural network based algorithm (CPNN) [7] and the BFGS-LLD algorithm [8]. Besides, Yang et al. [9] proposed a deep label distribution learning algorithm by combining LDL with deep learning. Yang et al. [10] applied LDL to visual sentiment analysis and got better performance than single-label learning. Jia et al. [11] studied the weakly supervised LDL problems based on transductive matrix completion.

However, label correlations are not considered in above works, i.e., a label can be helpful in learning another label under certain conditions in many real-world applications. For example, in the facial expression emotion recognition problem, when a face has a high description degree on label *anger*, the label *disgust* is more likely to have a higher description degree than label *happy*. Because *anger* and *disgust* are usually negative emotions while *happy* is a positive emotion. More data information can be utilized and a better performance can be achieved by taking into account the label correlations. Recently, label correlations were exploited in certain studies. In text emotion recognition [12], the prior knowledge based label correlations were captured by the Plutchik's wheel of emotions. Zhou et al. [13] and Jia et al. [14] applied Pearson's correlation to employ the label correlation. Although above works attempted to take into account label correlations, they usually exploited label correlations in a global manner under the assumption that the label correlations are shared by all instances. Nevertheless, in most real-world applications, label correlations are usually local, i.e., instances in different groups may share different label correlations. For example, Fig. 3 gives three pictures with different label ranks, and we assume *flower* and *butterfly* are two labels that have a correlation. In Fig. 3b, *butterfly* is difficult to be observed and thus could be tough to predict; but with the correlation between *flower* and *butterfly*, label *flower* can be helpful to predict label *butterfly* since the label *flower* is relatively easier to be observed in the image. However, in Fig. 3c, with the *flower* clearly presented, blind application of the correlation between *flower* and *butterfly* will result in an incorrect prediction because *butterfly* is

(a) *flower*, *butterfly*, *water*, *tree* (b) *flower*, *grass*, *cloud*, *butterfly* (c) *flower*, *cabin*, *tree*, *grass*

Fig. 3. Illustration of non-global label correlations.

usually not appeared in a *cabin*. The main reason for this case is that Fig. 3c is not in the same group as Figs. 3a and 3b. Therefore, some unnecessary or even misleading constraints may be enforced on some samples by exploiting such global correlations, and the performance may be hurt because some irrelevant labels will be predicted.

To solve above problem, the label correlations must be considered at a local level. In this paper, we propose two label distribution learning algorithms by exploiting label correlations locally. We assume that all instances can be divided into several clusters which are called local samples, and the correlations in different clusters are utilized. The instances in each cluster are as similar as possible, thus their label distributions should be as similar as possible, i.e., the same label correlations should be shared in each cluster. To encode the influence of different local samples and inspired by [15], a local correlation vector is constructed based on the different clustered local samples, and this vector is used as an additional feature for each instance. Different values in the additional feature vector represent the impacts of different clusters (local samples) on an instance. With the additional features combined original features, a new optimization problem is formulated. In addition, we develop an alternating minimization method based on gradient descent for the optimization. To address the challenges of vanilla gradient descent, adaptive moment estimation (Adam) method [16] is applied to solve the alternating minimization problem. In terms of different optimization methods, we propose two algorithms called GD-LDL-SCL and Adam-LDL-SCL, respectively. Extensive experiments validate that our proposed algorithms can obtain a promising performance when considering label correlations on local samples.

The main contributions can be summarized as follows:

- To encode the influence of clustered local samples, a local correlation vector is constructed based on the local samples. The experimental results suggests that it is more reasonable to exploit label correlations on local samples.
- To obtain and utilize the reasonable local correlation vectors, a novel objective function and a novel output model of LDL are proposed, which lead to a better performance.
- To solve the proposed formulation, an Adam based optimization method is applied, which has never been utilized in the LDL area. The theoretical and empirical analysis validate that the Adam based optimization algorithm is efficient and effective.

This paper extends our previous work [17] in the following aspects: 1) more knowledge and related works about LDL are introduced, including the insides of LDL and some technical details; 2) an Adam based optimization method is provided in the alternating optimization process, which is more efficient

and effective than the vanilla gradient descent based optimazation method; 3) to verify the performance of our proposed methods, five additional real-world data sets and one recently released LDL algorithm are provided; 4) more discussions are provided about the experimental results, including some statistic analysis.

The remainder of the paper is organized as follows. In Section 2, we briefly discuss existing works related to our proposed approaches. Our proposed methods will be presented in Section 3. Finally, we report the experimental results in Section 4, and followed by the conclusion in Section 5.

2 Related Work

In recent years, a number of algorithms, which can be divided into three groups, have been proposed for LDL. One group is based on the problem transformation (PT) strategy, which is a straightforward way to transform an LDL problem into a single-label learning problem. To be specific, each LDL training example is transformed into L single-label examples with the weights (description degree), where L denotes the number of predefined affective labels. Then, according to the weight of each example, the LDL training set with n examples is resampled to a standard single-label training set that includes $n \times L$ examples. Finally, any single-label learning method can be applied to the resampled training set. For example, SVM and Bayes are transformed to PT-SVM and PT-Bayes based on the PT strategy, respectively.

The second group is based on the algorithm adaptation (AA) strategy. Certain existing single-label learning approaches can be naturally extended to address the problem of label distribution, such as AA-kNN based on k-NN and AA-BP based on backpropagation neural network [5].

The final group of LDL algorithms consists of those based on the specialized algorithm (SA), which matches the LDL problem directly. Generally speaking, a specialized algorithm is composed of three parts, i.e., an output model, an objective function and an optimization method. Existing LDL approaches were designed from these three aspects. For instance, the specialized LDL algorithms mainly use the maximum entropy model [18] as the output model, such as the SA-IIS algorithm and the SA-BFGS algorithm [5]. In addition, some metrics used to evaluate the similarity of two distributions are usually applied as the objective function of LDL. In existing works, many divergence functions including the balanced divergence function [12], the Kullback-Leibler (KL) divergence [19], Jeffery divergence [20], and weighted Jeffery divergence [13], have been discussed in corresponding LDL algorithms. For the optimization problem, different methods were also developed in LDL. To name a few, improved iterative scaling (IIS) method and a more efficient method named BFGS were used in SA-IIS algorithm and SA-BFGS algorithm, respectively [5]; L-BFGS was applied to solve the emotion distribution learning problem [12].

To improve the performance of LDL, some algorithms attemp to exploit label correlations in different ways. In detail, the correlations were captured based on the Plutchik's wheel of emotions [12]; the label correlations were exploited by seeking the Pearson's correlation coefficients between two labels [13]; global label correlations were exploited for incomplete label distribution learning [21]; a distance-mapping function was employed to encode the global label correlations [14]; and the label correlations were exploited based on the optimal transport [22]. However, these approaches exploited label correlations at a global level, and we explained that it is more reasonable to use the label correlations locally in the introduction.

Recently, the related work has demonstrated that the third strategy is more effective than the previous two strategies [5]. Therefore, our proposed algorithms are designed based on the SA strategy, and they are constructed from above three aspects as well. In this paper, the output model and the objective function are improved by exploiting the influence of clustered local samples. Moreover, two alternating solutions based on vanilla gradient descent and Adam are employed for the optimization.

3 Our Proposed Approaches

3.1 The Framework

Let $\mathcal{X} = \mathcal{R}^q$ denote the q-dimensional input space, and let $\mathcal{Y} = \{y_1, y_2, \ldots, y_L\}$ denote the L predefined labels. Given a training set $S = \{(x_1, D_1), (x_2, D_2), \ldots, (x_n, D_n)\}$, where $D_i = \{d_i^1, d_i^2, \ldots, d_i^L\}$ is the label distribution associated with the instance x_i. We assign a value d_i^j called the description degree to an instance x_i for a particular label y_j, where $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$. All labels with non-zero d_i^j-s are the correct labels to describe the instance and satisfy $\sum_{j=1}^{L} d_i^j = 1$, which means that all labels in the set can fully describe the instance. The goal of LDL is to learn a mapping function $f : \mathcal{X} \rightarrow D$ to predict the label distributions for unseen instances.

As discussed in the Section 1, we construct an additional feature for each instance to exploit the label correlations locally. In detail, a local correlation vector c_i is designed for each instance x_i to encode the influence of different local samples, thus the original feature representation is expanded with the vector. Combined with the previous discussion, it can be discovered easily that the length of c_i is equal to the number of clustered local samples. Generally, similar instances always share similar label correlations, i.e., the instances that belong to the same cluster will have the similar local correlation vectors. Notice the similarity between different instances is measured in the label space rather than the feature space, because 1) similar label correlations are usually shared by the instances with similar label distribution; 2) a data set usually has a smaller label space compared to the feature space, which can reduce the clustering time. By combining the original features and the additional features, the following LDL output model is obtained based on the maximum entropy model:

$$p(y_l|x_i;\theta,w,c) = \frac{1}{\mathcal{Z}_i}\exp\left(\sum_k \theta_{l,k}x_i^k + \sum_t w_{l,t}c_i^t\right), \tag{1}$$

where $\mathcal{Z}_i = \sum_l \exp(\sum_k \theta_{l,k}x_i^k + \sum_t w_{l,t}c_i^t)$ is a normalization term, which is used to satisfy the requirement that the sum of all label description degrees of an instance equals 1. To simplify the formula, $p(y_l|x_i)$ is used to represent $p(y_l|x_i;\theta,w,c)$

in the remainder of this paper. In detail, the information of original features is denoted by the first term in the exponent part, while the information of additional features is represented by the second term, i.e., the information of local sample correlations. θ denotes the coefficient matrix of original features, in which $\theta_{l,k}$ is an element in θ. x_i^k is the kth original feature of instance x_i. w denotes the coefficient matrix of additional feature, in which $w_{l,t}$ is an element of w. c_i^t represents the influence of the tth cluster (local samples) on x_i. Aiming to combine the local correlation influence with global discrimination fitting into a unified framework, the following objective function is optimized:

$$\min_{\theta,w,c} V(\theta,w,c) + \lambda_1\Omega_1(\theta) + \lambda_2\Omega_2(w) + \lambda_3 Z(\theta,w,c), \tag{2}$$

where V is the loss function. Ω_1 and Ω_2 are two regularizers on θ and w, respectively. Z is a regularizer to enhance the local characteristic of the correlation vector, and λ_1, λ_2 and λ_3 are three parameters to balance the four terms.

As mentioned above, the purpose of LDL is to make the real distribution and predicted distribution as similar as possible, therefore V should be a function that can measure the similarity between different distributions. Many functions that can measure the similarity between two distributions were analyzed [23], such as euclidean distance, K-L divergence, Jeffery divergence, etc. Zhao and Geng [24] reported that the performance of K-L divergence is the most stable in the field of LDL according to several comparison experiments. Therefore, we adopt the K-L divergence as the basic loss function:

$$D_J(Q_a\|Q_b) = \sum_j Q_a^j \ln\frac{Q_a^j}{Q_b^j}, \tag{3}$$

where Q_a^j and Q_b^j denote the jth element of the two distributions Q_a and Q_b, respectively. In this paper, specifically, we define the expression for V based on K-L divergence as follows:

$$V(\theta,w,c) = \sum_{i=1}^{n}\sum_{l=1}^{L}\left(d_i^l \ln\left(\frac{d_i^l}{p(y_l|x_i)}\right)\right), \tag{4}$$

where d_i^l denotes the real description degree and $p(y_l|x_i)$ denotes the predicted description degree of label y_l to instance x_i.

For the regularizers on θ and w, we simply implement them as follows:

$$\Omega_1(\theta) = \|\theta\|_F^2, \tag{5}$$

$$\Omega_2(w) = \|w\|_F^2, \tag{6}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, it can be used to improve the generalization performance of the algorithm.

The final term of Eq. (2) is employed to enhance the local structure of the correlation vector, which exploits the sample correlations locally. Assume that all training instances can be divided into m different clusters $\{G_1, G_2, \ldots, G_m\}$ based on the label space, and each cluster is a local sample, in which the same label correlations are shared by all instances. This partitioning can be implemented by clustering or some domain knowledge, such as gene pathways [25] and networks [26] in bioinformatics applications. For easy implementation, k-means [27] is utilized as the clustering method. Moreover, we use the cluster center to represent the cluster, i.e., the local sample. The following formula is defined to compute the label distribution of corresponding cluster center:

$$p_j = \frac{1}{|G_j|}\sum_{x_k\in G_j} D_k, \tag{7}$$

where $|G_j|$ denote the number of instances in G_j. For given instance x_i, c_i^j is defined to measure the local influence of G_j on x_i. It can be easily observed that the more similar D_i and p_j, the more likely that x_i shares the same correlation with instances in G_j, which suggests a larger value of c_i^j. To exploit the influence of different local samples, a local correlation vector $c_i = [c_i^1, c_i^2, \ldots, c_i^m]$ is constructed for each instance x_i. Moreover, the euclidean distance is adopted to measure the correlation between a cluster center and an instance, and the penalty term is defined as follows:

$$Z(\theta,w,c) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_i^j \|p(y|x_i) - p_j\|_2^2, \tag{8}$$

where $p(y|x_i)$ denotes the predicted label distribution of instance x_i. By substituting Eqs. (4), (5), (6), and (8) into Eq. (2), the following optimization problem can be obtained:

$$\begin{aligned}\min_{\theta,w,c} &\sum_{i=1}^{n}\sum_{l=1}^{L}\left(d_i^l \ln\left(\frac{d_i^l}{p(y_l|x_i)}\right)\right) + \lambda_1\|\theta\|_F^2 + \lambda_2\|w\|_F^2 \\ &+ \lambda_3\sum_{i=1}^{n}\sum_{j=1}^{m} c_i^j\|p(y|x_i) - p_j\|_2^2 \quad s.t. \quad c_i^j \geq 0,\end{aligned} \tag{9}$$

where c_i^j represents the similarity of G_j and x_i, thus c_i^j is constrained to be a non-negative one.

3.2 Learning by Alternating Minimization

Eq. (9) has inequality constraints, therefore we use an interior point method to transform it into an unconstrained problem:

$$\begin{aligned}\min_{\theta,w,c} &\sum_{i=1}^{n}\sum_{l=1}^{L}\left(d_i^l \ln\left(\frac{d_i^l}{p(y_l|x_i)}\right)\right) + \lambda_1\|\theta\|_F^2 + \lambda_2\|w\|_F^2 \\ &+ \lambda_3\sum_{i=1}^{n}\sum_{j=1}^{m} c_i^j\|p(y|x_i) - p_j\|_2^2 + \mu\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{1}{c_i^j},\end{aligned} \tag{10}$$

where μ is the penalty factor and $\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{1}{c_i^j}$ is the barrier function.

We utilize the alternating minimization method to solve Eq. (10). To be specific, we update one of the variables in $\{\theta, w, c\}$ with vanilla gradient descent, and leave the others fixed. Vanilla gradient descent, also known as batch gradient descent, computes the gradient of the objective function w.r.t. parameters $\{\theta, w, c\}$ using the entire training datasets:

$$\begin{aligned}\theta_{t+1} &= \theta_t - \eta\cdot\nabla_{\theta_t}J(\theta_t) \\ w_{t+1} &= w_t - \eta\cdot\nabla_{w_t}J(w_t) \\ c_{t+1} &= c_t - \eta\cdot\nabla_{c_t}J(c_t),\end{aligned} \tag{11}$$

where η denotes the learning rate and $\nabla.J(\cdot)$ is the gradient of the objective function Eq. (10). Therefore, the key to gradient descent is obtaining the gradient of the objective function.

When w and c are fixed, to solve θ, Eq. (10) can be simplified to

$$\begin{aligned}\min_{\theta} \sum_{i=1}^{n}\sum_{l=1}^{L}\left(d_i^l \ln\left(\frac{d_i^l}{p(y_l|x_i)}\right)\right) + \lambda_1\|\theta\|_F^2 \\ + \lambda_3\sum_{i=1}^{n}\sum_{j=1}^{m} c_i^j \|p(y|x_i) - p_j\|_2^2.\end{aligned} \tag{12}$$

We obtain the gradient of the objective w.r.t. θ using gradient descent:

$$\begin{aligned}\nabla\theta_{l,k} = -\sum_{i=1}^{n} d_i^l x_i^k + \sum_{i=1}^{n} x_i^k p(y_l|x_i) + 2\lambda_1\theta_{l,k} \\ + 2\lambda_3\sum_{i=1}^{n}\sum_{j=1}^{m} c_i^j (p(y_l|x_i) - p_{j,l})\frac{\partial p(y_l|x_i)}{\partial \theta_{l,k}},\end{aligned} \tag{13}$$

where $\frac{\partial p(y_l|x_i)}{\partial \theta_{l,k}} = x_i^k p(y_l|x_i)(1 - p(y_l|x_i))$ and $p_{j,l}$ is the lth element of cluster center p_j.

When θ and c are fixed, to solve w, Eq. (10) can be simplified to

$$\begin{aligned}\min_{w} \sum_{i=1}^{n}\sum_{l=1}^{L}\left(d_i^l \ln\left(\frac{d_i^l}{p(y_l|x_i)}\right)\right) + \lambda_2\|w\|_F^2 \\ + \lambda_3\sum_{i=1}^{n}\sum_{j=1}^{m} c_i^j \|p(y|x_i) - p_j\|_2^2.\end{aligned} \tag{14}$$

Again, we obtain the gradient of the objective w.r.t. w using gradient descent:

$$\begin{aligned}\nabla w_{l,t} = -\sum_{i=1}^{n} d_i^l c_i^t + \sum_{i=1}^{n} c_i^t p(y_l|x_i) + 2\lambda_2 w_{l,t} \\ + 2\lambda_3\sum_{i=1}^{n}\sum_{j=1}^{m} c_i^j (p(y_l|x_i) - p_{j,l})\frac{\partial p(y_l|x_i)}{\partial w_{l,t}},\end{aligned} \tag{15}$$

where $\frac{\partial p(y_l|x_i)}{\partial w_{l,t}} = c_i^t p(y_l|x_i)(1 - p(y_l|x_i))$.

When θ and w are fixed, to solve c, Eq. (10) can be simplified to

$$\begin{aligned}\min_{c} \sum_{i=1}^{n}\sum_{l=1}^{L}\left(d_i^l \ln\left(\frac{d_i^l}{p(y_l|x_i)}\right)\right) + \mu\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{1}{c_i^j} \\ + \lambda_3\sum_{i=1}^{n}\sum_{j=1}^{m} c_i^j \|p(y|x_i) - p_j\|_2^2.\end{aligned} \tag{16}$$

Similarly, the gradient of the objective w.r.t. c is

$$\begin{aligned}\nabla c_i^j = -\sum_{l=1}^{L} d_i^l w_{l,j} + \frac{\sum_l (\exp(\sum_k \theta_{l,k}x_i^k + \sum_t w_{l,t}c_i^t)w_{l,j})}{\sum_l \exp(\sum_k \theta_{l,k}x_i^k + \sum_t w_{l,t}c_i^t)} \\ + 2\lambda_3 c_i^j \sum_l (p(y_l|x_i) - p_{j,l})\frac{\partial p(y_l|x_i)}{\partial c_i^j} \\ + \lambda_3\|p(y|x_i) - p_j\|_2^2 - \mu\frac{1}{(c_i^j)^2},\end{aligned} \tag{17}$$

where

$$\frac{\partial p(y_l|x_i)}{\partial c_i^j} = p(y_l|x_i)\left(w_{l,j} - \frac{\sum_l \left(\exp\left(\sum_k \theta_{l,k}x_i^k + \sum_t w_{l,t}c_i^t\right)w_{l,j}\right)}{\sum_l \exp(\sum_k \theta_{l,k}x_i^k + \sum_t w_{l,t}c_i^t)}\right).$$

Algorithm 1. The GD-LDL-SCL Algorithm

Input: training set $\{X, D\}$, parameters λ_1, λ_2, λ_3 and m.
Output: the label distribution D_t.

Train:
initialize θ, w, c, η and μ;
repeat
  compute $\nabla\theta$ by Eq. (13);
  $\theta = \theta - \eta \cdot \nabla\theta$;
  compute ∇w by Eq. (15);
  $w = w - \eta \cdot \nabla w$;
  compute ∇c by Eq. (17);
  $c = c - \eta \cdot \nabla c$;
  $\mu \quad \beta\mu, 0 < \beta < 1$;
until *convergence or maximum number of iterations*
for $j = 1$ *to* m **do**
  train a linear regression model R_j;
end
Test:
for $j = 1$ *to* m **do**
  predict c_t^j using R_j;
end
return the label distribution D_t according to Eq. (1).

Notice that the test instance x_t itself does not provide a local correlation vector c_t. Therefore, in the training phase, we first train m regression models on all training instances and their local correlation vectors, one for a dimension of the local correlation vector. In the testing phase, the outputs of m regression models on x_t are combined as its local correlation vector.

The optimization using vanilla gradient descent is presented in Algorithm 1, which is called GD-LDL-SCL. For the initial values of all parameters, we initialize the coefficient matrices θ and w to the all-ones matrices, and use the clustering result of k-means on the label space as the initial value of the local correlation matrix c. Specifically, if x_i belongs to the kth cluster, $c_i^k = 1$, otherwise, $c_i^k = 0$. Then, we utilize an alternating method to update the variables θ, w and c. Eq. (9) will converge to a global minimum since it is a convex function. After that, m regression models are trained to output the local correlation vector $c_t = [c_t^1, c_t^2, \ldots, c_t^m]$ for test instance x_t. Finally, we obtain the label distribution D_t by using Eq. (1).

However, vanilla gradient descent offers a few challenges that need to be addressed, i.e., 1) vanilla gradient descent needs to calculate the gradients for the whole training data in each update, thus it can be very time-consuming; 2) it can be difficult to choose a proper learning rate. A too small learning rate leads to slow convergence, while a too large learning rate does not guarantee convergence that causes the objective function to fluctuate around the minimum; 3) vanilla gradient descent applies the same learning rate to all parameter updates. If the data is sparse or the frequencies of features are very different, it is not proper to update all parameters to the same extent. Therefore, we attempt to optimize the sub-problems with adaptive moment estimation (Adam) [16].

Adam is the method that can calculate the learning rates for each parameter adaptively. It not only stores an exponentially decaying average of past squared gradients v_t, but also keeps an exponentially decaying average of past

gradients m_t, similar to Momentum method [28]. The formulas of m_t and v_t are as follows:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \end{aligned} \tag{18}$$

where β_1 and β_2 are the decay rates, which are always recommended closing to 1, and g_t is the gradient of the objective function at time step t. m_t and v_t are employed to estimate the first moment and the second moment of the gradients respectively. It is observed that m_t and v_t are biased towards zero when they are initialized as zero matrices, especially during the initial time steps. Thus, the biases of first and second moment are contracted as follows:

$$\begin{aligned} \widehat{m}_t &= \frac{m_t}{1-\beta_1^t}, \\ \widehat{v}_t &= \frac{v_t}{1-\beta_2^t}. \end{aligned} \tag{19}$$

Then, the update rules of parameters using Adam are as follows:

$$\Theta_{t+1} = \Theta_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \widehat{m}_t, \tag{20}$$

where Θ denotes the parameter that needs to update, η is the learning rate and ϵ is the smoothing term that avoids division by zero.

Besides, Adam updates parameters with a mini-batch gradient descent strategy. Mini-batch gradient descent performs an update for every mini-batch of δ training examples. Its advantages are that 1) it uses only a subset of training set in an update, thus it can speed up the update and save memory; 2) it reduces the variance of the parameter updates, which can lead to more stable convergence. Therefore, we obtain the gradients of Eq. (10) w.r.t. $\{\theta, w, c\}$, which are represented by $\nabla\theta_{l,k}^{(i':i'+\delta)}$, $\nabla w_{l,t}^{(i':i'+\delta)}$ and $\nabla c_i^{j\,(i':i'+\delta)}$ as follows:

$$\begin{aligned} \nabla\theta_{l,k}^{(i':i'+\delta)} &= -\sum_{i=i'}^{i'+\delta} d_i^l x_i^k + \sum_{i=i'}^{i'+\delta} x_i^k p(y_l|x_i) + 2\lambda_1\theta_{l,k} \\ &+ 2\lambda_3 \sum_{i=i'}^{i'+\delta} \sum_{j=1}^{m} c_i^j (p(y_l|x_i) - p_{j,l}) x_i^k p(y_l|x_i)(1-p(y_l|x_i)), \end{aligned} \tag{21}$$

$$\begin{aligned} \nabla w_{l,t}^{(i':i'+\delta)} &= -\sum_{i=i'}^{i'+\delta} d_i^l c_i^t + \sum_{i=i'}^{i'+\delta} c_i^t p(y_l|x_i) + 2\lambda_2 w_{l,t} \\ &+ 2\lambda_3 \sum_{i=i'}^{i'+\delta} \sum_{j=1}^{m} c_i^j (p(y_l|x_i) - p_{j,l}) c_i^t p(y_l|x_i)(1-p(y_l|x_i)), \end{aligned} \tag{22}$$

$$\begin{aligned} \nabla c_i^{j\,(i':i'+\delta)} &= \frac{\sum_l (\exp(\sum_k \theta_{l,k} x_i^k + \sum_t w_{l,t} c_i^t) w_{l,j})}{\sum_l \exp(\sum_k \theta_{l,k} x_i^k + \sum_t w_{l,t} c_i^t)} \\ &- \sum_{l=1}^{L} d_i^l w_{l,j} + 2\lambda_3 c_i^j \sum_l \Bigg[(p(y_l|x_i) - p_{j,l}) p(y_l|x_i) \\ &\left(w_{l,j} - \frac{\sum_l (\exp(\sum_k \theta_{l,k} x_i^k + \sum_t w_{l,t} c_i^t) w_{l,j})}{\sum_l \exp(\sum_k \theta_{l,k} x_i^k + \sum_t w_{l,t} c_i^t)} \right) \Bigg] \\ &+ \lambda_3 \|p(y|x_i) - p_j\|_2^2 - \mu \frac{1}{c_i^{j\,2}}. \end{aligned} \tag{23}$$

Thus, it performs much more efficiently and has potential to perform better than vanilla gradient descent. This improved algorithm is denoted by Adam-LDL-SCL, and its pseudo code is presented in Algorithm 2.

In summary, the Adam based optimization method addresses the challenges of vanilla gradient descent from the following two aspects: 1) the difficulty of choosing learning rate is addressed by using an adaptive strategy; 2) the time-consuming problem is addressed by using a mini-batch strategy. In detail, assuming that the time complexity of calculating the gradient of one sample is a constant C and the desired iteration step is T, the total time complexity of Adam based optimization method is $O(\delta CT)$, which is smaller than that of vanilla gradient descent method, $O(nCT)$.

Algorithm 2. The Adam-LDL-SCL Algorithm

Input: training set $\{X, D\}$, parameters λ_1, λ_2, λ_3 and m.
Output: the label distribution D_t.

Train:
initialize θ, w, c, η, β_1, β_2, ϵ, δ and μ;
$t = 1$, $m_0 = 0$, $v_0 = 0$;
repeat
  compute $\nabla\theta^{(i':i'+\delta)}$, $\nabla w^{(i':i'+\delta)}$ and $\nabla c^{(i':i'+\delta)}$ by Eq. (21), Eq. (22), and Eq. (23);
  update m_t and v_t w.r.t. $\{\theta, w, c\}$ by Eq. (18);
  compute $\widehat{m}_t$ and $\widehat{v}_t$ w.r.t. $\{\theta, w, c\}$ by Eq. (19);
  update θ, w and c by Eq. (20);
  $\mu \quad \beta\mu$, $0 < \beta < 1$;
  $t = t + 1$;
until *convergence or maximum number of iterations*
for $j = 1$ *to* m **do**
  train a linear regression model R_j;
end
Test:
for $j = 1$ to m **do**
  predict c_t^j using R_j;
end
return the label distribution D_t according to Eq. (1).

4 Experiments

4.1 Datasets

14 real-world data sets are used in the experiments including gene data, image data and text data. The characteristics of all 13 data sets are summarized in Table 1.

In detail, the first eight data sets used in the experiments are collected from biological experiments on the budding yeast *Saccharomyces cerevisiae* [29], where each data set denotes the result of an experiment. There are 2465 yeast genes in total, each of which is described by an associated phylogenetic profile vector with length 24. The normalized description degree of the corresponding label represents the gene expression level in different time points. There are 10 data sets in this series, but we only choose the data sets with 4 or more labels, because our goal is to improve the prediction performance by exploiting the label correlations, and the fewer the labels, the less the correlations.

The data set *Human Gene* is a large-scale data set with 30,542 examples, which is collected from the biological research that studies the relation between human genes and diseases. Each of example has 36 features that denote 36

TABLE 1
The Characteristics of the 14 Data Sets

No.	Dataset	#Examples	#Features	#Labels
1	Yeast-alpha	2,465	24	18
2	Yeast-cdc	2,465	24	15
3	Yeast-elu	2,465	24	14
4	Yeast-diau	2,465	24	7
5	Yeast-heat	2,465	24	6
6	Yeast-spo	2,465	24	6
7	Yeast-cold	2,465	24	4
8	Yeast-dtt	2,465	24	4
9	Human Gene	30,542	36	68
10	Nature Scene	2,000	294	9
11	SJAFFE	213	243	6
12	SBU_3DFE	2,500	243	6
13	Movie	7,755	1,869	5
14	Twitter_LDL	10,045	168	8

numerical descriptors for a sequence and 68 labels that denote 68 different diseases. Then, the gene expression levels after normalization constitute the label distribution of a particular human gene.

The data set *Nature Scene* is collected from 2,000 natural scene images with inconsistent multi-label ranking. Ten human rankers are demanded to label these images with nine possible labels, i.e., plant, sky, cloud, snow, building, desert, mountain, water and sun. For each image, each human ranker selects the relevant labels and ranks them in descending order independently. Thus, the result of multi-label rankings is expected to be highly inconsistent. Then, a nonlinear programming process is employed to transform the inconsistent rankings into a label distribution [30]. Finally, with the method proposed in [31], a 294-dimensional feature vector is extracted for each image.

The data sets *SJAFFE* and *SBU_3DFE* are two widely used facial expression image data sets, which are the extensions of JAFFE [32] and BU_3DFE [33]. There are 213 gray scale expression images posed by 10 Japanese female models, and each image is scored by 60 persons on the 6 basic emotion labels (i.e., happiness, sadness, surprise, fear, anger and disgust) with a five-level scale (1 represents the lowest emotion intensity, while 5 represents the highest emotion intensity). Different from most works on JAFFE, the average scores (after normalization) are used to represent the label distribution, rather than only considering the emotion with the highest score as a single-label problem, and the data set is extended to *SJAFFE* (Scored JAFFE). Similarly, the data set BU_3DFE with 2,500 facial expression images is extended to *SBU_3DFE* (Scored BU_3DFE). Besides, a 243-dimensional feature vector is extracted from each image in *SJAFFE* and *SBU_3DFE* by the Local Binary Patterns (LBP) method [34].

The data set *Movie* is about the user rating on movies coming from Netflix. There are 7,755 movies and 54,242,292 ratings from 478,656 users. For each movie, it is scored on a scale from 1 to 5 integral stars. The percentage of each rating level is calculated as the label distribution for each image. Besides, the features of a movie are extracted from the meta data (i.e., genre, director, actor, country, etc.), and the feature vector is of 1,869 dimensions by attribute transformation.

Finally, the last data set *Twitter_LDL* is an image sentiment data set that contains 10,045 images, whose labels fall in the typical eight-emotional space (i.e., anger, amusement, awe, contentment, disgust, excitement, fear and sadness). The features of the images in Twitter_LDL are extracted by three popular descriptors, i.e., LBP, HOG and Color Moment. Since the features we extracted are high-dimensional, we use PCA to reduce the dimensionality to 168.

TABLE 2
Evaluation Measures for LDL Algorithms

	Name	Formula
Distance	Euclidean↓	$Dis_1 = \sqrt{\sum_{j=1}^{L}(P_j - Q_j)^2}$
	Sϕrensen↓	$Dis_2 = \frac{\sum_{j=1}^{L}\lvert P_j - Q_j\rvert}{\sum_{j=1}^{c}\lvert P_j + Q_j\rvert}$
	Squared χ^2 ↓	$Dis_3 = \sum_{j=1}^{L}\frac{(P_j - Q_j)^2}{P_j + Q_j}$
	Kullback-Leibler(KL)↓	$Dis_4 = \sum_{j=1}^{L} P_j \ln \frac{P_j}{Q_j}$
Similarity	Intersection↑	$Sim_1 = \sum_{j=1}^{L}\min(P_j, Q_j)$
	Fidelity↑	$Sim_2 = \sum_{j=1}^{L}\sqrt{P_j Q_j}$

4.2 Evaluation Measures

In this paper, six measures, including four distance-based measures and two similarity-based measures, are chosen to evaluate the LDL algorithms. The distance-based measures are euclidean↓, Sϕrensen↓, Squared χ^2 ↓, and Kullback-Leibler(KL)↓. The similarity-based measures are Intersection↑ and Fidelity↑. The formulas are presented in Table 2, where P_j and Q_j are the jth element of the real label distribution and the predicted label distribution, respectively. For the former four distance-based measures, "↓" indicates "the smaller the better". For the latter two similarity-based measures, "↑" indicates "the larger the better".

4.3 Experimental Setting

To verify the performance of our proposed algorithms including GD-LDL-SCL and Adam-LDL-SCL, six state-of-the-art LDL algorithms, i.e., PT-SVM [19], AA-kNN [6], IIS-LLD [6], BFGS-LLD [7], EDL [13] and LDLLC [14], are compared on all data sets. The parameter settings for each algorithm are listed as follows. For PT-SVM, "C-SVC" type in LIBSVM is implemented with RBF kernel ($C = 1.0$ and $Gamma = 0.01$). The number of neighbors k in AA-kNN is set to 5. For BFGS-LLD, c_1 and c_2 are set to 10^{-4} and 0.9, respectively. For LDLLC, λ_1 and λ_2 are set to 0.1 and 0.01, respectively. For IIS-LLD and EDL, default parameter values in the corresponding literatures are applied. In GD-LDL-SCL and Adam-LDL-SCL, the parameters λ_1, λ_2 and λ_3 in Eq. (9) are tuned from the candidate set $\{10^{-3}, 10^{-2}, 10^{-1}, \ldots, 10^{3}\}$. The number of clusters obtained by k-means is tuned from 0 to 14, i.e., $m \in [0, 14]$. Notice that, $m = 0$ means that we do not consider the local correlation vector as the additional features. That is, there is no c and w in the output model, which becomes the classical LDL model.

4.4 Experimental Results

For each data set, the hold-out method is employed in this paper. In detail, the instances in each data set are randomly

TABLE 3
Comparison Results (mean ± std. (rank)) of Different LDL Algorithms on Real-World Data Sets Measured by Euclidean↓

Dataset	PT-SVM	AA-kNN	IIS-LLD	BFGS-LLD	EDL	LDLLC	GD-LDL-SCL	Adam-LDL-SCL
Yeast-alpha	.0276 ± .0006(7)	.0279 ± .0006(8)	.0269 ± .0004(6)	.0251 ± .0004(4)	.0260 ± .0011(5)	.0227 ± .0001(2)	.0229 ± .0003(3)	**.0225 ± .0003(1)**
Yeast-cdc	.0298 ± .0007(7)	.0301 ± .0009(8)	.0290 ± .0010(6)	.0284 ± .0011(5)	.0283 ± .0006(4)	.0276 ± .0009(3)	.0274 ± .0008(2)	**.0272 ± .0005(1)**
Yeast-elu	.0293 ± .0008(5)	.0297 ± .0010(6)	.0307 ± .0009(7)	.0308 ± .0009(8)	.0289 ± .0005(4)	.0277 ± .0006(2)	.0278 ± .0005(3)	**.0275 ± .0003(1)**
Yeast-diau	.0628 ± .0037(8)	.0567 ± .0019(6)	.0539 ± .0031(4)	**.0444 ± .0022(1)**	.0597 ± .0010(7)	.0541 ± .0022(5)	.0531 ± .0008(3)	.0525 ± .0008(2)
Yeast-heat	.0625 ± .0023(5)	.0624 ± .0020(4)	.0703 ± .0036(7)	.0728 ± .0031(8)	.0629 ± .0016(6)	.0605 ± .0015(3)	.0592 ± .0016(2)	**.0587 ± .0008(1)**
Yeast-spo	.0878 ± .0019(7)	.0879 ± .0030(8)	.0863 ± .0041(6)	.0819 ± .0045(4)	.0843 ± .0029(5)	**.0806 ± .0019(1)**	.0809 ± .0019(3)	.0808 ± .0007(2)
Yeast-cold	.0753 ± .0080(6)	.0724 ± .0027(4)	.0767 ± .0004(7)	.0745 ± .0004(5)	.0771 ± .0018(8)	.0679 ± .0003(3)	.0675 ± .0003(2)	**.0669 ± .0014(1)**
Yeast-dtt	.0516 ± .0029(7)	.0512 ± .0019(6)	.0535 ± .0023(8)	.0495 ± .0019(4)	.0508 ± .0022(5)	.0477 ± .0016(3)	.0474 ± .0017(2)	**.0472 ± .0008(1)**
Human Gene	.0886 ± .0192(5)	.1036 ± .0044(8)	.0877 ± .0078(4)	.0864 ± .0044(3)	.0887 ± .0021(6)	.1035 ± .0013(7)	.0858 ± .0016(2)	**.0855 ± .0011(1)**
Nature Scene	.5727 ± .0389(8)	.4269 ± .0256(2)	.4808 ± .0130(5)	.4384 ± .0090(3)	.5223 ± .0098(7)	.4852 ± .0141(6)	.4398 ± .0097(4)	**.4252 ± .0058(1)**
SJAFFE	.1615 ± .0301(8)	.1233 ± .0189(4)	.1509 ± .0149(6)	.1338 ± .0169(5)	.1216 ± .0061(3)	.1572 ± .0097(7)	.1141 ± .0054(2)	**.1099 ± .0057(1)**
SBU_3DFE	.1712 ± .0067(8)	.1631 ± .0079(5)	.1658 ± .0032(6)	.1625 ± .0040(4)	.1510 ± .0018(3)	.1685 ± .0041(7)	**.1407 ± .0008(1)**	.1408 ± .0013(2)
Movie	.3027 ± .0189(8)	.1814 ± .0063(4)	.2121 ± .0058(7)	.1939 ± .0059(6)	.1765 ± .0115(3)	.1845 ± .0041(5)	.1721 ± .0026(2)	**.1706 ± .0030(1)**
Twitter_LDL	.6750 ± .2366(8)	.4235 ± .0045(6)	.3947 ± .0039(5)	**.3761 ± .0039(1)**	.6428 ± .0022(7)	.3792 ± .0044(3)	.3775 ± .0048(2)	.3815 ± .0037(4)
Avg. Rank	6.92	5.64	6.00	4.35	5.21	4.07	2.21	1.42

The best performance of each data set is marked in bold.

TABLE 4
Comparison Results (mean ± std. (rank)) of Different LDL Algorithms on Real-World Data Sets Measured by Sϕrensen↓

Dataset	PT-SVM	AA-kNN	IIS-LLD	BFGS-LLD	EDL	LDLLC	GD-LDL-SCL	Adam-LDL-SCL
Yeast-alpha	.0445 ± .0009(7)	.0449 ± .0012(8)	.0429 ± .0012(5)	.0408 ± .0011(4)	.0429 ± .0022(6)	**.0272 ± .0001(1)**	.0372 ± .0005(3)	.0368 ± .0004(2)
Yeast-cdc	.0458 ± .0012(7)	.0462 ± .0013(8)	.0445 ± .0015(5)	.0449 ± .0016(6)	.0429 ± .0008(4)	.0422 ± .0013(3)	.0417 ± .0012(2)	**.0417 ± .0007(1)**
Yeast-elu	.0438 ± .0012(5)	.0443 ± .0014(6)	.0472 ± .0014(7)	.0475 ± .0012(8)	.0431 ± .0008(4)	.0412 ± .0006(3)	.0412 ± .0005(2)	**.0410 ± .0005(1)**
Yeast-diau	.0686 ± .0041(8)	.0622 ± .0022(6)	.0593 ± .0032(4)	**.0476 ± .0023(1)**	.0653 ± .0010(7)	.0596 ± .0026(5)	.0582 ± .0011(3)	.0574 ± .0010(2)
Yeast-heat	.0627 ± .0022(4)	.0632 ± .0018(5)	.0692 ± .0033(7)	.0791 ± .0029(8)	.0633 ± .0017(6)	.0610 ± .0016(3)	.0597 ± .0014(2)	**.0593 ± .0008(1)**
Yeast-spo	.0893 ± .0022(7)	.0899 ± .0024(8)	.0861 ± .0036(5)	.0833 ± .0038(2)	.0872 ± .0029(6)	**.0830 ± .0019(1)**	.0836 ± .0018(3)	.0837 ± .0007(4)
Yeast-cold	.0654 ± .0069(7)	.0630 ± .0024(4)	.0653 ± .0034(6)	.0641 ± .0035(5)	.0668 ± .0016(8)	.0589 ± .0019(3)	.0585 ± .0013(2)	**.0580 ± .0012(1)**
Yeast-dtt	.0447 ± .0024(7)	.0443 ± .0017(6)	.0480 ± .0023(8)	**.0409 ± .0017(1)**	.0440 ± .0018(5)	.0415 ± .0013(4)	.0411 ± .0015(3)	.0411 ± .0007(2)
Human Gene	.2191 ± .0124(6)	.2548 ± .0036(7)	.2187 ± .0054(4)	.2162 ± .0060(3)	.2191 ± .0018(5)	.2724 ± .0026(8)	.2146 ± .0014(2)	**.2139 ± .0015(1)**
Nature Scene	.6337 ± .0218(7)	**.4369 ± .0188(1)**	.5400 ± .0082(6)	.4769 ± .0070(4)	.6344 ± .0074(8)	.5321 ± .0119(5)	.4561 ± .0077(2)	.4710 ± .0031(3)
SJAFFE	.1545 ± .0253(7)	.1201 ± .0148(4)	.1482 ± .0130(6)	.1312 ± .0129(5)	.1184 ± .0063(3)	.1551 ± .0082(8)	.1110 ± .0053(2)	**.1077 ± .0059(1)**
SBU_3DFE	.1649 ± .0056(8)	.1523 ± .0066(4)	.1608 ± .0037(6)	.1569 ± .0035(5)	.1449 ± .0018(3)	.1640 ± .0042(7)	**.1352 ± .0009(1)**	.1354 ± .0010(2)
Movie	.2877 ± .0209(8)	.1781 ± .0055(4)	.2026 ± .0048(7)	.1816 ± .0053(5)	.1766 ± .0099(3)	.1825 ± .0035(6)	.1763 ± .0019(2)	**.1758 ± .0028(1)**
Twitter_LDL	.6272 ± .1036(7)	.4031 ± .0036(6)	.3985 ± .0029(5)	.3746 ± .0040(3)	.6623 ± .0025(8)	.3784 ± .0040(4)	**.3645 ± .0039(1)**	.3674 ± .0035(2)
Avg. Rank	6.79	5.50	5.79	4.29	5.42	4.36	2.14	1.71

The best performance of each data set is marked in bold.

divide into 5 parts, one part for testing and the remainder for training. We apply each method 10 times on each data set, and the experimental results are presented in the form of "mean±std. (rank)". The ranks refer to the prediction effect of all LDL algorithms on each measure, and a smaller value represents a better performance. Besides, the best result on each measure is marked in bold in each table. The experimental results are reported in Tables 3, 4, 5, 6, 7, and 8. Each table reports the comparison result on one evaluation measure.

There are eight LDL algorithms in the experiment, in which PT-SVM is based on the problem transformation strategy, AA-kNN is based on the adaptation strategy, and the rest six algorithms are all specialized algorithms for LDL. By analyzing the experimental results reported from Tables 3, 4, 5, 6, 7, and 8, we can observe that the specialized LDL algorithms generally have the better performance than PT-based algorithms and AA-based algorithms in most cases. The reason is that the specialized LDL algorithms are designed to directly maximize the similarity between the predicted label distribution and the real label distribution. Meanwhile, for PT-based algorithms, the transformation from a label distribution training set to a single-label training set may lose the original label distribution and potential correlation information. For AA-based algorithms, k-NN treats the description degree in the distribution as independent ones, ignoring the distribution information among them. As for the specialized LDL algorithms, our proposed GD-LDL-SCL and Adam-LDL-SCL obtain the top 2 performances in most cases. GD-LDL-SCL and Adam-LDL-SCL are comparable with each other. It is worth mentioning that, in most cases, our proposed methods perform better than the EDL algorithm and LDLLC algorithm which exploit label correlations globally. The result suggests that it is more reasonable to exploit label correlations on local samples.

To investigate the relative performance among the comparing algorithms, *Friedman test* is further performed, which is a favorable statistical test for comparisons of more than two algorithms over multiple data sets [35]. Table 9 reports the Friedman statistics F_F and the corresponding critical value on each evaluation measures. As can we seen, the null hypothesis of indistinguishable performance among the comparing methods is rejected for each evaluation measure

TABLE 5
Comparison Results (mean ± std. (rank)) of Different LDL Algorithms on Real-World Data Sets Measured by Squared χ^2 ↓

Dataset	PT-SVM	AA-kNN	IIS-LLD	BFGS-LLD	EDL	LDLLC	GD-LDL-SCL	Adam-LDL-SCL
Yeast-alpha	.0071 ± .0003(7)	.0073 ± .0003(8)	.0069 ± .0004(6)	.0063 ± .0008(4)	.0067 ± .0006(5)	**.0026 ± .0001(1)**	.0054 ± .0001(3)	.0052 ± .0001(2)
Yeast-cdc	.0077 ± .0004(7)	.0080 ± .0004(8)	.0073 ± .0005(6)	.0070 ± .0004(4)	.0072 ± .0004(5)	**.0035 ± .0004(1)**	.0068 ± .0005(3)	.0067 ± .0003(2)
Yeast-elu	.0068 ± .0005(5)	.0071 ± .0006(7)	.0071 ± .0004(6)	.0075 ± .0004(8)	.0067 ± .0003(3)	.0068 ± .0004(4)	.0063 ± .0003(2)	**.0062 ± .0002(1)**
Yeast-diau	.0169 ± .0018(8)	.0145 ± .0011(6)	.0144 ± .0014(5)	**.0089 ± .0008(1)**	.0158 ± .0005(7)	.0132 ± .0005(4)	.0125 ± .0005(3)	.0123 ± .0004(2)
Yeast-heat	.0141 ± .0010(4.5)	.0141 ± .0010(4.5)	.0182 ± .0016(7)	.0188 ± .0016(8)	.0143 ± .0008(6)	.0128 ± .0008(3)	.0127 ± .0009(2)	**.0126 ± .0006(1)**
Yeast-spo	.0280 ± .0015(7)	.0286 ± .0020(8)	.0251 ± .0036(5)	.0229 ± .0019(2)	.0268 ± .0015(6)	**.0222 ± .0007(1)**	.0243 ± .0010(3)	.0245 ± .0003(4)
Yeast-cold	.0147 ± .0033(6)	.0136 ± .0011(4)	.0157 ± .0015(8)	.0139 ± .0013(5)	.0154 ± .0009(7)	.0121 ± .0007(3)	.0121 ± .0006(2)	**.0120 ± .0006(1)**
Yeast-dtt	.0071 ± .0009(8)	.0071 ± .0007(7)	.0068 ± .0005(5)	.0058 ± .0005(2)	.0069 ± .0007(6)	.0061 ± .0003(4)	.0059 ± .0004(3)	**.0058 ± .0003(1)**
Human Gene	.1902 ± .1091(7)	.2314 ± .0084(8)	.1873 ± .0129(5)	.1847 ± .0138(3)	.1902 ± .0027(6)	.1866 ± .0029(4)	.1826 ± .0026(2)	**.1809 ± .0028(1)**
Nature Scene	1.0465 ± .0521(8)	.6494 ± .0423(2)	.8166 ± .0193(6)	.6841 ± .0175(5)	.9824 ± .0161(7)	**.6282 ± .0209(1)**	.6556 ± .0150(3)	.6747 ± .0058(4)
SJAFFE	.0720 ± .0192(7)	.0968 ± .0262(8)	.0673 ± .0112(6)	.0549 ± .0173(5)	.0458 ± .0041(4)	**.0365 ± .0036(1)**	.0403 ± .0032(2)	.0404 ± .0035(3)
SBU_3DFE	.0810 ± .0057(8)	.0773 ± .0079(7)	.0759 ± .0028(6)	.0725 ± .0034(5)	.0636 ± .0014(4)	**.0398 ± .0017(1)**	.0563 ± .0006(3)	.0551 ± .0008(2)
Movie	.2492 ± .0287(8)	.1146 ± .0121(4)	.1332 ± .0090(7)	.1233 ± .0107(6)	.1176 ± .0138(5)	**.0616 ± .0019(1)**	.1082 ± .0022(3)	.1076 ± .0030(2)
Twitter_LDL	1.0256 ± .3908(7)	.5723 ± .0063(6)	.5453 ± .0057(5)	.5035 ± .0072(3)	1.0875 ± .0046(8)	.5115 ± .0072(4)	**.4945 ± .0059(1)**	.5022 ± .0055(2)
Avg. Rank	6.96	6.25	5.93	4.36	5.64	2.36	2.50	2.00

The best performance of each data set is marked in bold.

TABLE 6
Comparison Results (mean ± std. (rank)) of Different LDL Algorithms on Real-World Data Sets Measured by KL↓

Dataset	PT-SVM	AA-kNN	IIS-LLD	BFGS-LLD	EDL	LDLLC	GD-LDL-SCL	Adam-LDL-SCL
Yeast-alpha	.0071 ± .0003(7)	.0074 ± .0004(8)	.0069 ± .0004(6)	.0063 ± .0004(4)	.0068 ± .0006(5)	.0054 ± .0001(2.5)	.0054 ± .0001(2.5)	**.0052 ± .0001(1)**
Yeast-cdc	.0076 ± .0004(7)	.0079 ± .0004(8)	.0072 ± .0005(6)	.0070 ± .0005(4)	.0072 ± .0004(5)	.0068 ± .0001(3)	.0067 ± .0004(2)	**.0067 ± .0003(1)**
Yeast-elu	.0068 ± .0005(5)	.0071 ± .0006(7)	.0071 ± .0004(6)	.0073 ± .0003(8)	.0067 ± .0003(3)	.0068 ± .0003(4)	.0062 ± .0003(2)	**.0062 ± .0002(1)**
Yeast-diau	.0167 ± .0017(8)	.0145 ± .0010(6)	.0141 ± .0013(5)	**.0083 ± .0008(1)**	.0155 ± .0005(7)	.0130 ± .0010(4)	.0124 ± .0005(3)	.0122 ± .0005(2)
Yeast-heat	.0141 ± .0010(4.5)	.0141 ± .0010(4.5)	.0182 ± .0016(7)	.0186 ± .0015(8)	.0143 ± .0008(6)	.0131 ± .0006(3)	.0126 ± .0009(2)	**.0125 ± .0006(1)**
Yeast-spo	.0284 ± .0015(7)	.0286 ± .0002(8)	.0252 ± .0022(5)	**.0226 ± .0021(1)**	.0269 ± .0016(6)	.0236 ± .0013(2)	.0242 ± .0010(3)	.0243 ± .0004(4)
Yeast-cold	.0146 ± .0033(6)	.0136 ± .0011(4)	.0155 ± .0015(8)	.0143 ± .0015(5)	.0153 ± .0009(7)	.0120 ± .0006(2)	.0121 ± .0006(3)	**.0120 ± .0005(1)**
Yeast-dtt	.0071 ± .0009(8)	.0070 ± .0007(7)	.0068 ± .0005(5)	**.0054 ± .0004(1)**	.0068 ± .0008(6)	.0061 ± .0005(4)	.0058 ± .0005(3)	.0058 ± .0003(2)
Human Gene	.2466 ± .0324(6)	.2894 ± .0176(7)	.2414 ± .0310(4)	.2372 ± .0243(3)	.2466 ± .0052(5)	.3262 ± .0077(8)	.2347 ± .0042(2)	**.2316 ± .0049(1)**
Nature Scene	1.4950 ± .0193(8)	1.2227 ± .0161(7)	.9495 ± .0431(4)	.8047 ± .0377(2)	1.1649 ± .0242(6)	.9772 ± .0433(5)	.8694 ± .0342(3)	**.7788 ± .0090(1)**
SJAFFE	.0764 ± .0220(8)	.0524 ± .0263(4)	.0712 ± .0135(6)	.0559 ± .0168(5)	.0487 ± .0046(3)	.0754 ± .0083(7)	**.0409 ± 0.0038(1)**	.0415 ± .0040(2)
SBU_3DFE	.0889 ± .0074(8)	.0832 ± .0107(7)	.0818 ± .0033(5)	.0776 ± .0042(4)	.0672 ± .0015(3)	.0827 ± .0038(6)	.0591 ± .0006(2)	**.0579 ± .0008(1)**
Movie	.2553 ± .0359(8)	.1259 ± .0173(4)	.1399 ± .0102(7)	.1363 ± .0138(6)	.1287 ± .0014(5)	.1243 ± .0018(3)	**.1034 ± .0091(1)**	.1088 ± .0140(2)
Twitter_LDL	1.5432 ± .6752(7)	1.6956 ± .0512(8)	.6671 ± .0107(3)	**.6287 ± .0106(1)**	1.3268 ± .0065(6)	.6348 ± .0114(2)	.6683 ± .0104(4)	.6949 ± .0098(5)
Avg. Rank	6.96	6.39	5.50	3.86	5.21	3.96	2.39	1.79

The best performance of each data set is marked in bold.

TABLE 7
Comparison Results (mean ± std. (rank)) of Different LDL Algorithms on Real-World Data Sets Measured by Intersection↑

Dataset	PT-SVM	AA-kNN	IIS-LLD	BFGS-LLD	EDL	LDLLC	GD-LDL-SCL	Adam-LDL-SCL
Yeast-alpha	.9565 ± .0009(7)	.9561 ± .0012(8)	.9571 ± .0012(5)	.9574 ± .0009(4)	.9570 ± .0022(6)	.9628 ± .0004(2)	.9627 ± .0005(3)	**.9632 ± .0004(1)**
Yeast-cdc	.9554 ± .0012(7)	.9538 ± .0013(8)	.9556 ± .0015(6)	.9558 ± .0016(5)	.9571 ± .0008(4)	.9577 ± .0013(3)	.9582 ± .0013(2)	**.9583 ± .0007(1)**
Yeast-elu	.9562 ± .0012(5)	.9557 ± .0014(6)	.9528 ± .0015(8)	.9552 ± .0017(7)	.9569 ± .0007(4)	.9580 ± .0013(3)	.9587 ± .0005(2)	**.9589 ± .0005(1)**
Yeast-diau	.9314 ± .0041(8)	.9378 ± .0022(6)	.9407 ± .0033(4)	**.9513 ± .0027(1)**	.9347 ± .0010(7)	.9404 ± .0026(5)	.9417 ± .0011(3)	.9426 ± .0011(2)
Yeast-heat	.9373 ± .0022(4)	.9368 ± .0018(5)	.9309 ± .0033(7)	.9304 ± .0034(8)	.9366 ± .0017(6)	9389 ± .0014(3)	.9402 ± .0014(2)	**.9407 ± .0007(1)**
Yeast-spo	.9107 ± .0022(7)	.9096 ± .0034(8)	.9139 ± .0036(5)	.9168 ± .0039(2)	.9128 ± .0028(6)	**9169 ± .0019(1)**	.9164 ± .0019(3)	.9163 ± .0007(4)
Yeast-cold	.9346 ± .0069(7)	.9370 ± .0024(4)	.9347 ± .0034(6)	.9348 ± .0035(5)	.9332 ± .0016(8)	.9414 ± .0019(3)	.9414 ± .0013(2)	**.9420 ± .0012(1)**
Yeast-dtt	.9553 ± .0024(7)	.9557 ± .0017(6)	.9520 ± .0023(8)	.9584 ± .0023(4)	.9560 ± .0018(5)	.9585 ± .0013(3)	.9587 ± .0015(2)	**.9589 ± .0007(1)**
Human Gene	.7810 ± .0126(6)	.7451 ± .0036(7)	.7813 ± .0054(4)	.7838 ± .0060(3)	.7810 ± .0018(5)	.7276 ± .0026(8)	.7854 ± .0013(2)	**.7860 ± .0015(1)**
Nature Scene	.3656 ± .0218(8)	**.5630 ± .0188(1)**	.4600 ± .0082(6)	.5231 ± .0070(4)	.3662 ± .0074(7)	.4679 ± .0119(5)	.5439 ± .0077(2)	.5289 ± .0031(3)
SJAFFE	.8455 ± .0253(7)	.8798 ± .0148(4)	.8518 ± .0130(6)	.8688 ± .0129(5)	.8816 ± .0063(3)	.8449 ± .0082(8)	.8890 ± .0054(2)	**.8922 ± .0058(1)**
SBU_3DFE	.8351 ± .0056(8)	.8477 ± .0066(4)	.8392 ± .0037(6)	.8431 ± .0035(5)	.8551 ± .0018(3)	.8360 ± .0042(7)	**.8647 ± .0007(1)**	.8646 ± .0010(2)
Movie	.7123 ± .0209(8)	.8219 ± .0055(4)	.7974 ± .0048(7)	.8184 ± .0053(5)	.8228 ± .0094(3)	.8175 ± .0035(6)	**.8247 ± .0017(1)**	.8241 ± .0028(2)
Twitter_LDL	.3728 ± .1936(7)	.5969 ± .0036(6)	.6015 ± .0029(5)	.6234 ± .0040(3)	.3377 ± .0025(8)	.6216 ± .0040(4)	**.6354 ± .0039(1)**	.6325 ± .0035(2)
Avg. Rank	6.86	5.50	5.93	4.36	5.36	4.36	2.00	1.64

The best performance of each data set is marked in bold.

at 0.05 significance level. Then, we apply the post-hoc *Bonferroni-Dunn test* [35] to test whether our proposed algorithms (GD-LDL-SCL and Adam-LDL-SCL) achieve competitive performance against the comparing algorithms. In detail, we consider the GD-LDL-SCL and Adam-LDL-SCL as the control algorithms, respectively. Besides, we calibrate the average rank difference against the comparing algorithms with the *critical difference* (CD). Therefore, the

TABLE 8
Comparison Results (mean ± std. (rank)) of Different LDL Algorithms on Real-World Data Sets Measured by Fidelity↑

Dataset	PT-SVM	AA-kNN	IIS-LLD	BFGS-LLD	EDL	LDLLC	GD-LDL-SCL	Adam-LDL-SCL
Yeast-alpha	.9981 ± .0001(7)	.9980 ± .0001(8)	.9983 ± .0011(6)	.9985 ± .0011(4)	.9983 ± .0002(5)	.9987 ± .0001(2)	.9986 ± .0001(3)	**.9988 ± .0001(1)**
Yeast-cdc	.9980 ± .0001(7.5)	.9980 ± .0001(7.5)	.9982 ± .0012(6)	.9983 ± .0011(4)	.9982 ± .0001(5)	.9983 ± .0002(3)	.9983 ± .0001(2)	**.9984 ± .0001(1)**
Yeast-elu	.9983 ± .0002(5)	.9982 ± .0002(6)	.9982 ± .0035(7)	.9979 ± .0009(8)	.9983 ± .0001(4)	.9984 ± .0001(2.5)	.9984 ± .0001(2.5)	**.9985 ± .0001(1)**
Yeast-diau	.9957 ± .0004(8)	.9963 ± .0003(5)	.9964 ± .0036(4)	**.9978 ± .0031(1)**	.9960 ± .0002(7)	.9962 ± .0002(6)	.9968 ± .0001(3)	.9969 ± .0001(2)
Yeast-heat	.9964 ± .0003(4.5)	.9964 ± .0003(4.5)	.9954 ± .0042(8)	.9961 ± .0048(7)	.9963 ± .0003(6)	.9966 ± .0003(3)	.9967 ± .0003(2)	**.9968 ± .0002(1)**
Yeast-spo	.9929 ± .0004(7)	.9927 ± .0005(8)	.9937 ± .0005(5)	**.9951 ± .0007(1)**	.9932 ± .0004(6)	.9939 ± .0003(2)	.9938 ± .0002(4)	.9938 ± .0001(3)
Yeast-cold	.9963 ± .0008(6)	.9966 ± .0003(5)	.9960 ± .0039(8)	.9968 ± .0036(4)	.9961 ± .0003(7)	.9967 ± .0023(3)	**.9969 ± .0001(1)**	.9968 ± .0001(2)
Yeast-dtt	.9982 ± .0003(8)	.9982 ± .0002(6)	.9983 ± .0013(5)	**.9989 ± .0010(1)**	.9982 ± .0002(7)	9985 ± .0002(3.5)	.9985 ± .0002(3.5)	.9985 ± .0001(2)
Human Gene	.9443 ± .0281(6)	.9324 ± .0034(7)	.9454 ± .0049(4)	.9462 ± .0052(3)	.9444 ± .0010(5)	.9267 ± .0014(8)	.9469 ± .0007(2)	**.9474 ± .0009(1)**
Nature Scene	.5796 ± .0254(8)	.7301 ± .0194(4)	.6766 ± .0080(6)	.7307 ± .0076(3)	.6087 ± .0062(7)	6859 ± .0104(5)	**.7429 ± .0060(1)**	.7339 ± .0026(2)
SJAFFE	.9815 ± .0050(8)	.9868 ± .0050(4)	.9827 ± .0030(6)	.9859 ± .0049(5)	.9883 ± .0011(3)	.9818 ± .0018(7)	**.9897 ± .0008(1)**	.9896 ± .0009(2)
SBU_3DFE	.9791 ± .0015(8)	.9800 ± .0022(7)	.9806 ± .0007(5)	.9815 ± .0009(4)	.9837 ± .0004(3)	.9801 ± .0009(6)	.9856 ± .0001(2)	**.9859 ± .0002(1)**
Movie	.9306 ± .0082(8)	.9691 ± .0040(4)	.9642 ± .0033(6)	.9624 ± .0034(7)	.9686 ± .0059(5)	.9692 ± .0010(3)	**.9739 ± .0006(1)**	.9733 ± .0008(2)
Twitter_LDL	.5809 ± .1626(7)	.7549 ± .0028(6)	.7777 ± .0019(5)	.7959 ± .0028(3)	.5478 ± .0019(8)	.7924 ± .0029(4)	**.8074 ± .0019(1)**	.8042 ± .0021(2)
Avg. Rank	7.00	5.86	5.79	3.93	5.57	4.14	2.07	1.64

The best performance of each data set is marked in bold.

control algorithm is deemed to perform significantly compared to a comparing algorithm if their average ranks differ by at least one CD (CD = 2.490 in this paper: # comparing algorithms $k = 8$, # data sets $N = 14$).

The CD diagrams on each evaluation measure is reported in Fig. 4, and the average rank of each algorithm is marked along the axis (lower ranks to the right). In addition, the algorithm whose average rank is within one CD to that of GD-LDL-SCL or Adam-LDL-SCL is connected with a thick line. Otherwise, any algorithm that is not connected with GD-LDL-SCL or Adam-LDL-SCL is deemed to have significant difference between them. As can we seen from Fig. 4, the following observations can be obtained: 1) Adam-LDL-SCL always achieves the optimal average rank on all evaluation measures, and GD-LDL-SCL can achieve the sub-optimal performance except for *Squared* χ^2; 2) Adam-LDL-SCL outperms PT-Bayes, AA-kNN, IIS-LLD, BFGS-LLD, EDL and LDLLC in terms of *euclidean*, *Sørensen*, *Intersection*, and outperms PT-Bayes, AA-kNN, IIS-LLD and EDL in terms of *Squared* χ^2, *KL* and *Fidelity*; 3) GD-LDL-SCL outper PT-Bayes, AA-kNN, IIS-LLD and EDL on all evaluation measures; 4) Adam-LDL-SCL and GD-LDL-SCL are comparable in terms of all evaluation measures.

4.5 Influence of Parameters

To examine the robustness of our proposed methods, the influence of parameters is analyzed in the experiment, which includes λ_1 (regularization parameter for the first Frobenius norm regularizer), λ_2 (regularization parameter for the second Frobenius norm regularizer), λ_3 (regularization parameter for local correlations) in Eq. (9) and m (the number of clusters). Fig. 5 reports the related experimental results on data set *Yeast-cold*. Since GD-LDL-SCL and Adam-LDL-SCL solve the same optimization problem with same parameters, and it can be seen from Tables 3, 4, 5, 6, 7, and 8 that the results of GD-LDL-SCL and Adam-LDL-SCL are similar, we only show the result of GD-LDL-SCL in the following. We run GD-LDL-SCL with λ_1 which is set to $\{10^{-4}, 10^{-3}, \ldots, 10^{2}, 10^{3}\}$ and so does with λ_2 and λ_3. Moreover, to investigate the influence of the number of clusters, we run GD-LDL-SCL with m varying from 0 to 14 with step size of 2. The related results are shown in Figs. 5 and 6, respectively.

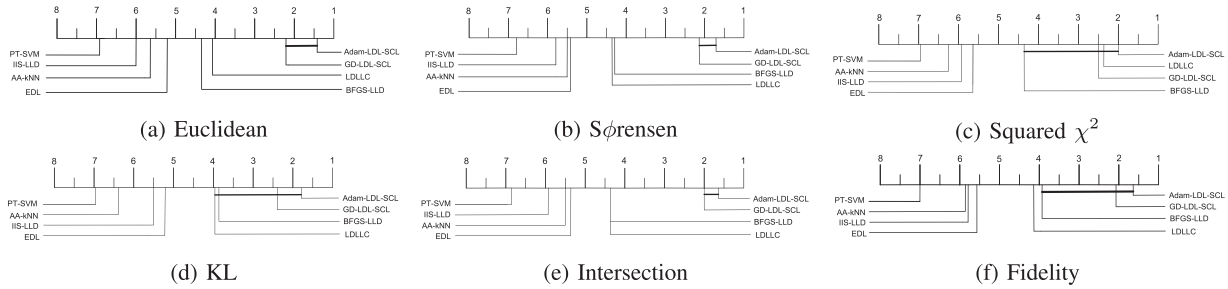
As for the influence of parameters λ_1, λ_2 and λ_3 shown in Fig. 5, it can be observed that the performance is getting worse when three parameters are large enough because the objective function is no more dominated by the first term, and the performance achieve better results when the value of λ_1, λ_2 and λ_3 within a certain range ($\lambda_1 \in [10^{-3}, 10^{-5}]$, $\lambda_2 \in [10^{-2}, 10^{-3}]$, $\lambda_3 \in [10^{-1}, 10^{-4}]$). Moreover, we can also observe that the influence of λ_2 is smaller than the influence of λ_1 and λ_3. In this paper, the setting of λ_1, λ_2 and λ_3 are as follows:

- As for the data sets of *Yeast* and *Twitter_LDL*, we set $\lambda_1 = 10^{-3}$, $\lambda_2 = 10^{-3}$, $\lambda_3 = 10^{-3}$.
- As for *Nature Scene*, *SJAFFE* and *SBU_3DFE*, we set $\lambda_1 = 10^{-4}$, $\lambda_2 = 10^{-3}$, $\lambda_3 = 10^{-3}$.
- As for *Human Gene* and *Movie*, we set $\lambda_1 = 10^{-5}$, $\lambda_2 = 10^{-3}$, $\lambda_3 = 10^{-4}$.

For the influence of number of clusters m, as shown in Fig. 6, the worst result appears when $m = 0$, which means the label correlations on local samples are no longer considered in the output model. Not surprisingly, this worst result is comparable to the results of IIS-LLD and BFGS-LLD. Furthermore, the performance of GD-LDL-SCL rises steadily when m is less than 7, and tends to be stable after m becomes sufficiently large. In this paper, we set $m = 12$ in data sets *Human Gene* and *Movie*, $m = 5$ in *SJAFFE* and *Twitter_LDL*, and $m = 8$ in other data sets.

TABLE 9
Friedman Statistics F_F in Terms of Each Evaluation Metric and the Critical Value at 0.05 Significance Level (# Comparing Algorithms $k = 8$ and # Datasets $N = 14$)

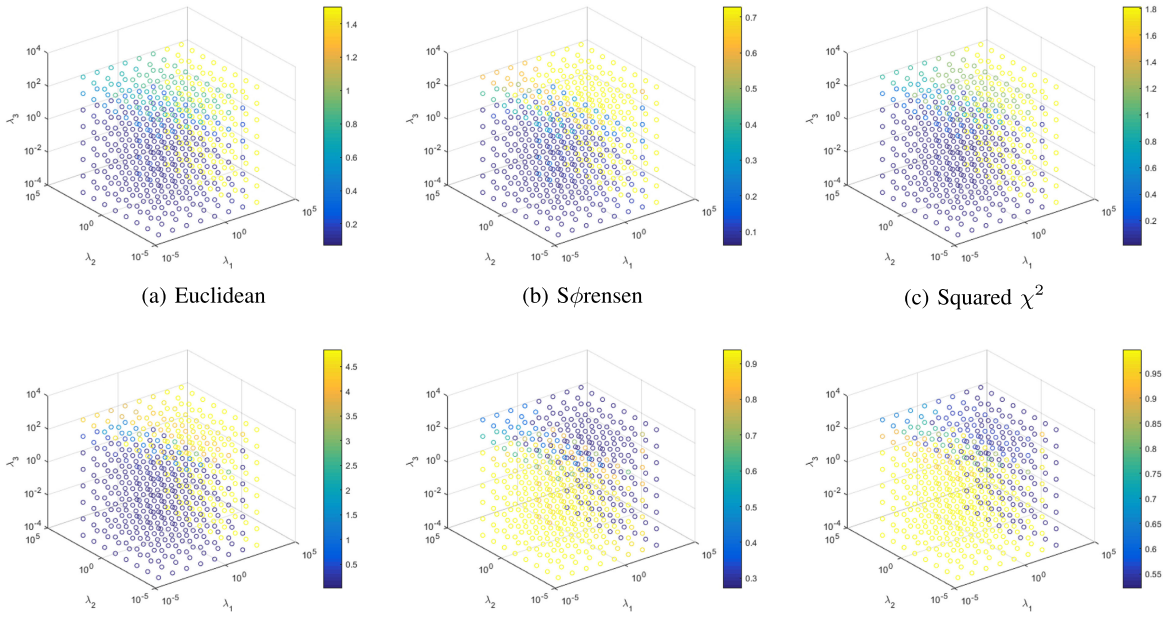
Evaluation metrics	F_F	Critical value
euclidean	16.0859	2.1119
Sørensen	14.5368	
Squared χ^2	24.1626	
KL	17.6143	
Intersection	17.1873	
Fidelity	19.9862	



(a) Euclidean (b) Sϕrensen (c) Squared χ^2

(d) KL (e) Intersection (f) Fidelity

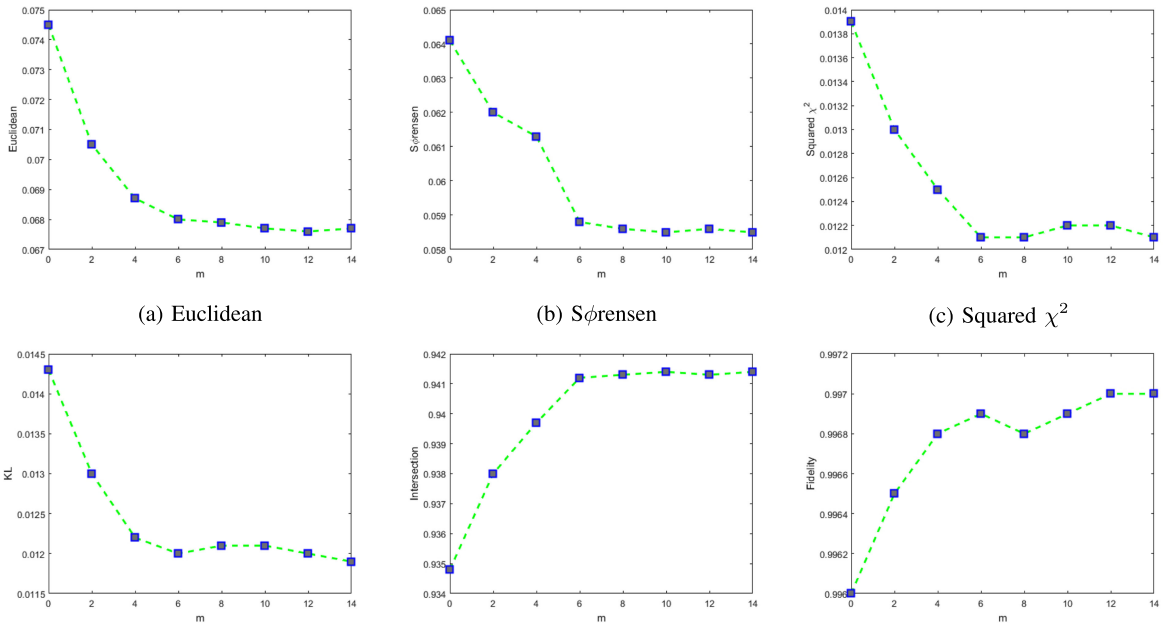
Fig. 4. Comparison of our algorithms against six comparison algorithms with the *Bonferroni-Dunn* test. Algorithms not connected with Adam-LDL-SCL and GD-LDL-SCL in the CD diagram are considered to have significantly performance from the control algorithms (CD = 2.490 at 0.05 significance level).



(a) Euclidean (b) Sϕrensen (c) Squared χ^2

(d) KL (e) Intersection (f) Fidelity

Fig. 5. Influence of λ_1, λ_2, and λ_3 with six measures on data set *Yeast-cold*.



(a) Euclidean (b) Sϕrensen (c) Squared χ^2

(d) KL (e) Intersection (f) Fidelity

Fig. 6. Influence of m with six measures on dataset *Yeast-cold*.

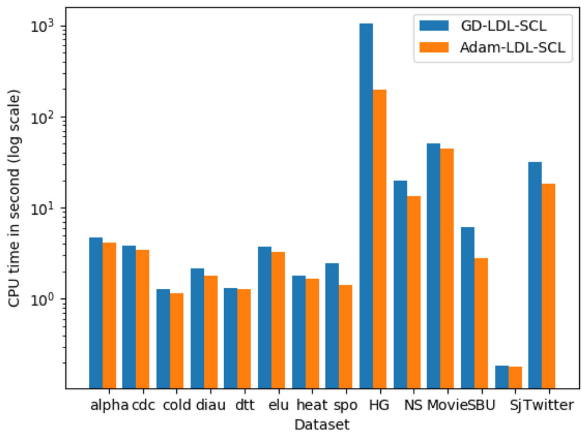


Fig. 7. Training time (in second) of GD-LDL-SCL and Adam-LDL-SCL on all datasets. *Human Gene*, *Nature Scene*, *SJAFFE*, and *SBU_3DFE* are abbreviated to HG, NS, SJ, and SBU.

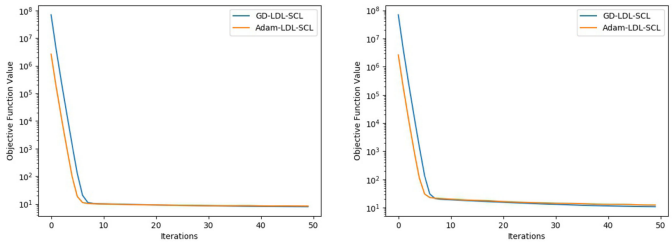
4.6 Training Time and Convergency

The training time of GD-LDL-SCL and Adam-LDL-SCL is shown in Fig. 7. It can be observed that Adam-LDL-SCL is much faster than GD-LDL-SCL especially on large-scale data sets. In addition, to investigate the convergence of the optimization methods that used in our proposed model, we plot the corresponding values of objective function Eq. (10) regarding GD-LDL-SCL and Adam-LDL-SCL on two data sets (*Yeast-alpha* and *Yeast-cold*) in Fig. 8. It can be seen that the value of objective function decreases as the number of iterations increases, and gradually tends to a stable value after a few iterations (about 10 iterations for *Yeast-alpha* and *Yeast-cold*). Besides, Adam-LDL-SCL converges faster than GD-LDL-SCL, and both of them can achieve the similar minimum value.

5 Conclusion

Label distribution learning can be seen as a generalization form of multi-label learning, which can address the more complex label ambiguity problems. Contrasting to some existing algorithms that exploit label correlations in a global manner, in this paper, we exploit the label correlations at a local level by assuming that the instances in different local samples usually share different label correlations. A local correlation vector is derived for each instance, which encodes the influence of different local samples, and two optimization algorithms named GD-LDL-SCL and Adam-LDL-SCL are proposed. Extensive experiments on some real-world data sets demonstrate that our proposed methods can effectively address the label distribution problems and have superior performance to some state-of-the-art label distribution learning methods.

In view of the fact that LDL has a stronger expression ability in label ambiguity problems, it has been widely applied to many fields, such as yeast gene expression, human gene expression, natural scene classification, emotion recognition, film score prediction, and so on. However, the data annotation of the label distribution requires more manpower and resources in the real world, it will be a better way to do the label enhancement first based on the multi-label data. Then applying LDL algorithms may obtain a better result than traditional multi-label learning methods. In future work, label enhancement based multi-label learning is worthwhile to be further explored.



(a) *Yeast-alpha* (b) *Yeast-cold*

Fig. 8. Convergence of GD-LDL-SCL and Adam-LDL-SCL on *Yeast-alpha* and *Yeast-cold*.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China under Grant 2017YFC0820601, the National Natural Science Foundation of China (61773208, 61772275, 61906090, 61672285), and the Natural Science Foundation of Jiangsu Province (BK20191287, BK20170809, BK20170033). The work of Xiuyi Jia is supported by the State Key Laboratory for Novel Software Technology, Nanjing University, P.R. China.

References

[1] G. Tsoumakas, I. Katakis, and D. Taniar, "Multi-label classification: An overview," *Int. J. Data Warehousing Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[2] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Proc. IEEE Int. Conf. Data Mining*, 2008, pp. 995–1000.

[3] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach. Learn.*, vol. 85, no. 3, pp. 333–348, 2011.

[4] Z. Li and J. Tang, "Weakly supervised deep matrix factorization for social image understanding," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 276–288, Jan. 2017.

[5] X. Geng, "Label distribution learning," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1734–1748, Jul. 2016.

[6] X. Geng, K. Smith-Miles, and Z. H. Zhou, "Facial age estimation by learning from label distributions," in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 451–456.

[7] X. Geng, C. Yin, and Z. H. Zhou, "Facial age estimation by learning from label distributions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 10, pp. 2401–2412, Oct. 2013.

[8] X. Geng and R. Ji, "Label distribution learning," in *Proc. IEEE Int. Conf. Data Mining Workshops*, 2014, pp. 377–383.

[9] X. Yang, B. B. Gao, C. Xing, and Z. W. Huo, "Deep label distribution learning for apparent age estimation," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, 2015, pp. 344–350.

[10] J. Yang, D. She, and M. Sun, "Joint image emotion classification and distribution learning via deep convolutional neural network," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3266–3272.

[11] X. Jia, T. Ren, L. Chen, J. Wang, J. Zhu, and X. Long, "Weakly supervised label distribution learning based on transductive matrix completion with sample correlations," *Pattern Recognit. Lett.*, vol. 125, pp. 453–462, 2019.

[12] D. Zhou, X. Zhang, Y. Zhou, Q. Zhao, and X. Geng, "Emotion distribution learning from texts," in *Proc. Conf. Empirical Methods Natural Language Process.*, 2016, pp. 638–647.

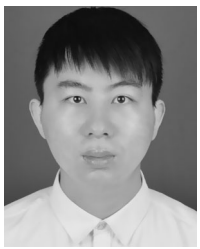
[13] Y. Zhou, H. Xue, and X. Geng, "Emotion distribution recognition from facial expressions," in *Proc. ACM Int. Conf. Multimedia*, 2015, pp. 1247–1250.

[14] X. Jia, W. Li, J. Liu, and Y. Zhang, "Label distribution learning by exploiting label correlations," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3310–3317.

[15] S. J. Huang and Z. H. Zhou, "Multi-label learning by exploiting label correlations locally," in *Proc. AAAI Conf. Artif. Intell.*, 2012, pp. 949–955.
[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–13.
[17] X. Zheng, X. Jia, and W. Li, "Label distribution learning by exploiting sample correlations locally," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4556–4563.
[18] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra, "A maximum entropy approach to natural language processing," *Comput. Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.
[19] X. Geng, Q. Wang, and Y. Xia, "Facial age estimation by adaptive label distribution learning," in *Proc. IEEE Int. Conf. Pattern Recognit.*, 2014, pp. 4465–4470.
[20] X. Geng and Y. Xia, "Head pose estimation based on multivariate label distribution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1837–1842.
[21] M. Xu and Z. Zhou, "Incomplete label distribution learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 3175–3181.
[22] Z. Peng and Z. Zhou, "Label distribution learning by optimal transport," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4506–4513.
[23] S. H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *Int. J. Math. Models Methods Appl. Sci.*, vol. 1, no. 4, pp. 300–307, 2007.
[24] Q. Zhao and X. Geng, "Selection of target function in label distribution learning," *J. Frontiers Comput. Sci. Technol.*, vol. 11, no. 5, pp. 708–719, 2017.
[25] A. Subramanian, P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. G. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, et al., "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles," *Proc. Nat. Academy Sci. United States America*, vol. 102, no. 43, pp. 15 545–15 550, 2005.
[26] H. Chuang, E. Lee, Y. Liu, D. Lee, and T. Ideker, "Network-based classification of breast cancer metastasis," *Molecular Syst. Biol.*, vol. 3, no. 1, pp. 140–149, 2007.
[27] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.
[28] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999.
[29] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns." *Proc. Nat. Academy Sci. United States America*, vol. 95, no. 25, pp. 14 863–14 868, 1998.
[30] X. Geng and L. Luo, "Multilabel ranking with inconsistent rankers," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3742–3747.
[31] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognit.*, vol. 37, no. 9, pp. 1757–1771, 2004.
[32] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2002, pp. 200–205.
[33] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato, "A 3D facial expression database for facial behavior research," in *Proc. IEEE Int. Conf. Autom. Face Gesture Recognit.*, 2006, pp. 211–216.
[34] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.
[35] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, no. 1, pp. 1–30, 2006.

Xiuyi Jia received the PhD degree in computer science from Nanjing University, Nanjing, China, in 2011. He is currently an associate professor in the School of Computer Science and Engineering, Nanjing University of Science and Technology. His recent research focuses on machine learning, granular computing, and data mining. He is a member of the IEEE.

Zechao Li received the BE degree from the University of Science and Technology of China, in 2008, and the PhD degree from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, in 2013. He is currently a professor with the Nanjing University of Science and Technology. His research interests include big media analysis, computer vision, etc. He is a member of the IEEE.



Xiang Zheng received the master's degree from the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include data mining and machine learning.

Weiwei Li received the PhD degree in computer science from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2016. She is currently an assistant professor at the Nanjing University of Aeronautics and Astronautics. Her research interests include machine learning, software data mining, and knowledge engineering.

Sheng-Jun Huang received the BSc and PhD degrees in computer science from Nanjing University, China, in 2008 and 2014, respectively. He is now a professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. His main research interests include machine learning and data mining. He has been selected to the Young Elite Scientists Sponsorship Program by CAST in 2016, and won the China Computer Federation Outstanding Doctoral Dissertation Award in 2015, the KDD Best Poster Award at the in 2012, and the Microsoft Fellowship Award in 2011. He is a junior associate editor of *Frontiers of Computer Science*. He is a member of the IEEE.