Label Distribution Learning

Xin Geng*, *Member, IEEE*

Abstract—Although multi-label learning can deal with many problems with label ambiguity, it does not fit some real applications well where the overall distribution of the importance of the labels matters. This paper proposes a novel learning paradigm named *label distribution learning* (LDL) for such kind of applications. The label distribution covers a certain number of labels, representing the degree to which each label describes the instance. LDL is a more general learning framework which includes both single-label and multi-label learning as its special cases. This paper proposes six working LDL algorithms in three ways: problem transformation, algorithm adaptation, and specialized algorithm design. In order to compare the performance of the LDL algorithms, six representative and diverse evaluation measures are selected via a clustering analysis, and the first batch of label distribution datasets are collected and made publicly available. Experimental results on one artificial and fifteen real-world datasets show clear advantages of the specialized algorithms, which indicates the importance of special design for the characteristics of the LDL problem.

Index Terms—Multi-label learning, label distribution learning, learning with ambiguity

1 Introduction

Learning with ambiguity is a hot topic in recent machine learning and data mining research. A learning process is essentially building a mapping from the instances to the labels. This paper mainly focuses on the ambiguity at the label side of the mapping, i.e., one instance is not necessarily mapped to one label. In the existing learning paradigms, there are mainly two cases of label assignment: (1) a single label is assigned to this instance, and (2) multiple labels are assigned to this instance. Single-label learning (SLL) assumes that all the instances in the training set are labeled in the first way. Multi-label learning (MLL) [42] allows the training instances to be labeled in the second way. Thus, MLL can deal with the ambiguous case where one instance belongs to more than one classes (labels). Generally speaking, current MLL algorithms have been developed with two strategies [41]. The first strategy is *problem transformation*, where the basic idea is to transform the MLL task into one or more SLL tasks. For example, the MLL problem could be transformed into binary classification problems [38], a label ranking problem [22], or an ensemble learning problem [26]. The second strategy is *algorithm adaptation*, where the basic idea is to extend specific SLL algorithms to handle multi-label data. For example, it can be extended from *k*-NN [51], decision tree [37], or neural networks [50].

Both SLL and MLL actually aim to answer the essential question "*which label can describe the instance?*", while MLL deals with label ambiguity by allowing the answer to consist of more than one label. However, neither SLL nor MLL can directly handle the further question with more ambiguity "*how much does each label describe the instance?*", i.e., the relative importance of each label is also involved in the description of the instance. Surprisingly, the real-world data with the information about such relative importance of each label might be more common than many people think. To name just a few from the datasets used in Section 5, first, in many scientific experiments, the result is not a single output, but a series of numerical outputs. For example, the biological experiments on the yeast genes over a period of time yield different gene expression levels on a series of time points (the 2nd to 11th datasets in Table 1) [15]. The exact expression level on each time point alone is of little importance. What really matters is the overall expression distribution over the whole time period. If the learning task is to predict such distribution for a given gene, then it can be hardly fit into either the SLL or MLL framework because the role of each output in the distribution is crucial, and there is no partition of relevant and irrelevant labels at all. Another example is the emotion analysis from facial expressions (the 14th and 15th datasets in Table 1) [29], [48]. A facial expression often conveys a complex mixture of multiple basic emotions (e.g., happiness, sadness, surprise, fear, anger, and disgust). Each basic emotion plays a different role in the expression. The various intensities of all the basic emotions naturally form an emotion distribution for the facial expression. By regarding the emotion with the highest intensity or emotions with higher intensities than a threshold as the positive label(s), the problem can be fit into the SLL or MLL framework. Unfortunately, this will lose the important information of the different intensities of the related emotions. More examples of real-world data with label distributions can be found in Section 5.2.

For the applications mentioned above, 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, representing the degree to which y describes $\boldsymbol{x}$. For example, if $\boldsymbol{x}$ represents a facial expression image, y represents an emotion, then $d_{\boldsymbol{x}}^{y}$ should be the intensity of the emotion y expressed by the image $\boldsymbol{x}$. Without loss of generality, assume that $d_{\boldsymbol{x}}^{y} \in [0,1]$. Further suppose that the label set is complete, i.e., using all the labels in the set can always fully describe the instance. Then, $\sum_{y} d_{\boldsymbol{x}}^{y} = 1$.

- *Xin Geng is with the School of Computer Science and Engineering, and the Key Lab of Computer Network and Information Integration (Ministry of Education), Southeast University, Nanjing 211189, China.*
** Corresponding author. E-mail: xgeng@seu.edu.cn*

Such $d_{\boldsymbol{x}}^{y}$ is called the *description degree* of y to $\boldsymbol{x}$. For a particular instance, the description degrees of all the labels constitute a data form similar to probability distribution. So, it is called *label distribution*. The learning process on the instances labeled by label distributions is therefore called *label distribution learning* (LDL). The aforementioned applications reveal that the nature of label ambiguity might exceed the current framework of MLL. In such case, it is necessary to extend MLL to LDL. In fact, the real applications suitable for LDL might be more common than those suitable for MLL because when there are multiple labels associated with one instance, their importance or relevance to the instance can hardly be exactly the same. While MLL usually assumes indiscriminate importance within the relevant label set (e.g., all represented by '1's) as well as within the irrelevant label set (e.g., all represented by '0's), LDL allows direct modeling of different importance of each label to the instance, and thus can better match the nature of many real applications.

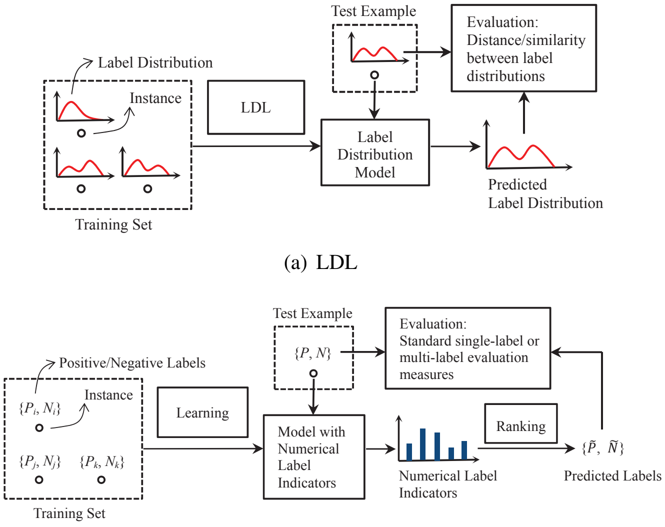
The main contribution of this paper includes: 1) A novel machine learning paradigm named label distribution learning is formulated; 2) Six working LDL algorithms are proposed and compared in the experiments; 3) Six measures for evaluation of LDL algorithms are suggested; 4) The first batch of 15 real-world label distribution datasets are prepared and made publicly available. This paper extends our preliminary work [17] in the following ways: 1) More technical details and insights of LDL are discussed; 2) The set of evaluation measures for LDL algorithms are empirically optimized to reveal different aspects of the algorithms; 3) Five additional real-world LDL datasets are provided; 4) Extensive additional experiments are performed and discussed in further detail.

The rest of the paper is organized as follows. Firstly, some related work are briefly reviewed and discussed in Section 2. Secondly, the problem of LDL is formulated in Section 3. Then, six LDL algorithms are proposed in Section 4. After that, the experiments on artificial as well as real-world datasets are reported in Section 5. Finally, the paper is summarized and some discussions on the future work are given in Section 6.

2 Related Work

It is not a rare case in existing single-label or multi-label machine learning literatures that an intermediate numerical indicator (e.g., probability, confidence, grade, score, vote, etc.) is calculated for each label [47], [51], [22], [11]. As shown in Fig. 1, LDL is different from these learning methods in mainly three aspects:

1) Each training instance of LDL is explicitly associated with a label distribution, rather than a single label or a relevant (positive) label set. The label distribution comes from the application itself as a natural part of the original data, while most numerical label indicators used in previous learning algorithms are artificially generated from the original data for later decision making.
2) The purpose of most numerical label indicators used in previous learning algorithms is to rank the labels, and then decide the positive label(s) through, say, thresholding over the ranking. The actual value of the indicator does not matter much so long as the ranking does not change. In most cases, it only cares about the partition between positive and negative labels. On the other hand, what LDL cares about is the overall label distribution. The value of each label's description degree is important.
3) The performance evaluation measures of previous learning algorithms with numerical label indicators are still those commonly used for SLL (e.g., classification accuracy, error rate, etc.) or MLL (e.g., Hamming loss, one-error, coverage, ranking loss, etc.). On the other hand, the performance of LDL should be evaluated by the similarity or distance between the predicted label distribution and the real label distribution, which will be further discussed in Section 5.1.



(a) LDL

(b) Typical existing learning methods with numerical label indicators

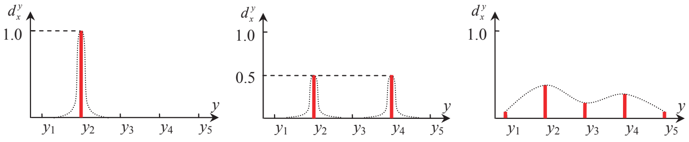
Fig. 1. Comparison of LDL and typical existing learning methods with numerical label indicators.

In the LDL framework, each instance is labeled by a real-valued vector (i.e., the label distribution). Merely from the data format point of view, there are also similar work in the machine learning literature where each instance is associated with a supervision vector (real-valued or binary). Two typical examples are known as *label embedding* [5] and *attribute learning* [2], [25]. Both of them are featured by the intermediate representations for the classes. In detail, *label embedding* [5] projects the class labels into a subspace to gain advantages in sharing representations of the labels. In the subspace, each class is represented by a real-valued vector that can be regarded as a code for that class. *Attribute learning* [2], [25] is mainly designed to leverage the prior knowledge of attribute-class association to deal with the missing classes (zero-shot learning) or scarce classes (few-shots learning). Each class has the same attribute representation for all its instances. For both label embedding and attribute learning, the intermediate vector representation is fully determined by the class label itself, i.e., all the instances belonging to the same class will be labeled by the same vector. Thus, although the supervision signal is transformed from a label into a vector, this does not change the nature of the learning paradigm: each instance is still associated with one class label (represented by a vector though), and the final aim is still standard classification. On the other hand, label distribution is part of the nature of the data rather than an intermediate representation. One label distribution represents the relative roles of all the labels with respect to the instance rather than a code for one class. Different instances therefore usually correspond to different label distributions. This clearly distinguishes LDL from either label embedding or attribute learning.

The goal of LDL is to predict multiple real-valued description degrees of the labels. When concerning prediction of multiple targets, there is a very general concept in the literature known as *multi-target learning* (MTL) [44]. In MTL, the multiple targets might refer to not only binary, nominal, ordinal, or real-valued variables, but also rankings or relational structures, such as a tree or a graph. Thus, MTL includes a variety of subfields of machine learning and statistics as its special cases, such as multi-label classification (multiple binary targets) [42], multivariate regression (multiple numerical targets) [23], sequence learning (ordered targets) [46], structured prediction (targets with inherent structure) [4], preference learning (preference relation between multiple targets) [16], multi-task learning (multiple targets in different but related domains) [9] and collective learning (dependent targets) [32]. If the targets are real-valued and they satisfy the two distribution constrains, i.e., $d_{\boldsymbol{x}}^{y} \in [0,1]$, and $\sum_y d_{\boldsymbol{x}}^{y} = 1$, then MTL becomes LDL. Therefore, LDL could be viewed as another special case of MTL.

From the conceptual point of view, it is worthwhile to distinguish description degree from the concept *membership* used in *fuzzy classification* [53]. Membership is a truth value that may range between completely true and completely false. It is designed to handle the status of *partial truth* which often appears in the non-numeric linguistic variables. On the other hand, description degree reflects the *ambiguity* of the label description of the instance, i.e., one label may only partially describe the instance, but it is completely true that the label describes the instance. Based on fuzzy set theory, a recent extension of multi-label classification, namely graded multilabel classification [11], allows for graded membership of an instance belonging to a class. Apart from the different meanings of continuous description degree and graded membership, LDL is also different from this work in methodology. The strategy in [11] is to reduce the graded multi-label problem to the conventional multi-label problem, while LDL aims to directly model the mapping from the instances to the label distributions.

Note also that $d_{\boldsymbol{x}}^{y}$ is *not* the *probability* that y correctly labels $\boldsymbol{x}$, but *the proportion that y accounts for in a full description of $\boldsymbol{x}$*. Thus, all the labels with a non-zero description degree are actually the 'correct' labels to describe the instance, but just with different importance measured by $d_{\boldsymbol{x}}^{y}$. Recognizing this, one can distinguish label distribution from the previous studies on probabilistic labels [39], [12], [35], where the basic assumption is that there is only one 'correct' label for each instance. Fortunately, although not a probability by definition, $d_{\boldsymbol{x}}^{y}$ still shares the same constraints with probability, i.e., $d_{\boldsymbol{x}}^{y} \in [0,1]$ and $\sum_y d_{\boldsymbol{x}}^{y} = 1$. Thus, many theories and methods in statistics can be applied to label distributions.



(a) Single-label annot. (b) Multi-label annot. (c) General case

Fig. 2. Example label distributions for single-label annotation, multi-label annotation, and the general case.

3 Formulation of LDL

First of all, the main notations used in this paper are listed as follows. The instance variable is denoted by $\boldsymbol{x}$, the particular i-th instant is denoted by $\boldsymbol{x}_i$, the label variable is denoted by y, the particular j-th label value is denoted by y_j, the description degree of y to $\boldsymbol{x}$ is denoted by $d_{\boldsymbol{x}}^{y}$, and the label distribution of $\boldsymbol{x}_i$ is denoted by $D_i = \{d_{\boldsymbol{x}_i}^{y_1}, d_{\boldsymbol{x}_i}^{y_2}, \cdots, d_{\boldsymbol{x}_i}^{y_c}\}$, where c is the number of possible label values.

By the definition of label distribution given in Section 1, both single-label annotation and multi-label annotation can be viewed as special cases of label distribution. Fig. 2 gives one label distribution example for single-label annotation, multi-label annotation, and the general case, respectively. For the single-label annotation (a), the label y_2 fully describes the instance, so $d_{\boldsymbol{x}}^{y_2} = 1$. For the multi-label annotation (b), each of the two positive labels y_2 and y_4 by default describes 50% of the instance, so $d_{\boldsymbol{x}}^{y_2} = d_{\boldsymbol{x}}^{y_4} = 0.5$. Finally, (c) represents a general case of label distribution, which satisfies the constraints $d_{\boldsymbol{x}}^{y} \in [0,1]$ and $\sum_y d_{\boldsymbol{x}}^{y} = 1$. Such examples illustrate that label distribution is more general than both single-label annotation and multi-label annotation, and thus can provide more flexibility in the learning process.

Nevertheless, more flexibility usually means larger output space. From single-label annotation to multi-label annotation, and then to label distribution, the size of the output space of the learning process becomes increasingly larger. In detail, for a problem with c different labels, there are c possible outputs for SLL, and $2^c - 1$ possible outputs for MLL. As for LDL, there are infinite possible outputs as long as they satisfy that $d_{\boldsymbol{x}}^{y} \in [0,1]$ and $\sum_y d_{\boldsymbol{x}}^{y} = 1$.

As mentioned before, since label distribution shares the same constraints with probability distribution, many statistical theories and methods can be applied to label distribution. First of all, $d_{\boldsymbol{x}}^{y}$ can be represented by the form of conditional probability, i.e., $d_{\boldsymbol{x}}^{y} = P(y|\boldsymbol{x})$. Then, the problem of LDL can be formulated as follows.

> Let $\mathcal{X} = \mathbb{R}^q$ denote the input space and $\mathcal{Y} = \{y_1, y_2, \cdots, y_c\}$ denote the complete set of labels. Given a training set $S = \{(\boldsymbol{x}_1, D_1), (\boldsymbol{x}_2, D_2), \cdots, (\boldsymbol{x}_n, D_n)\}$, the goal of ldl is to learn a conditional probability mass function $p(y|\boldsymbol{x})$ from S, where $\boldsymbol{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Suppose $p(y|\boldsymbol{x})$ is a parametric model $p(y|\boldsymbol{x};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the parameter vector. Given the training set S, the goal of LDL is to find the $\boldsymbol{\theta}$ that can generate a distribution similar to D_i

given the instance $\boldsymbol{x}_i$. As will be discussed in Section 5.1, there are different criteria that can be used to measure the distance or similarity between two distributions. For example, if the Kullback-Leibler divergence is used as the distance measure, then the best parameter vector $\boldsymbol{\theta}^*$ is determined by

$$\begin{aligned}\boldsymbol{\theta}^* &= \operatorname*{argmin}_{\boldsymbol{\theta}} \sum_i \sum_j \left(d_{\boldsymbol{x}_i}^{y_j} \ln \frac{d_{\boldsymbol{x}_i}^{y_j}}{p(y_j|\boldsymbol{x}_i;\boldsymbol{\theta})} \right) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_i \sum_j d_{\boldsymbol{x}_i}^{y_j} \ln p(y_j|\boldsymbol{x}_i;\boldsymbol{\theta}). \qquad (1)\end{aligned}$$

It is interesting to examine the traditional learning paradigms under the optimization criterion shown in Eq. (1). For SLL (see Fig. 2(a) for example), $d_{\boldsymbol{x}_i}^{y_j} = Kr(y_j, y(\boldsymbol{x}_i))$, where $Kr(\cdot,\cdot)$ is the Kronecker delta function and $y(\boldsymbol{x}_i)$ is the single label of $\boldsymbol{x}_i$. Consequently, Eq. (1) can be simplified to

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_i \ln p(y(\boldsymbol{x}_i)|\boldsymbol{x}_i;\boldsymbol{\theta}). \qquad (2)$$

This is actually the maximum likelihood (ML) estimation of $\boldsymbol{\theta}$. The later use of $p(y|\boldsymbol{x};\boldsymbol{\theta})$ for classification is equivalent to the maximum a posteriori (MAP) decision.

For MLL, each instance is associated with a label set (see Fig. 2(b) for example). Consequently, Eq. (1) can be changed into

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_i \frac{1}{|Y_i|} \sum_{y \in Y_i} \ln p(y|\boldsymbol{x}_i;\boldsymbol{\theta}), \qquad (3)$$

where Y_i is the label set associated with $\boldsymbol{x}_i$. Eq. (3) can be viewed as a ML criterion weighted by the reciprocal cardinality of the label set associated with each instance. In fact, this is equivalent to first applying Entropy-based Label Assignment (ELA) [42] to transform the multi-label instances into the weighted single-label instances, and then optimizing the ML criterion based on the weighted single-label instances.

It can be seen from the above analysis that with proper constraints, an LDL model can be transformed into the commonly used methods for SLL or MLL. Thus, LDL may be viewed as a more general learning framework which includes both SLL and MLL as its special cases.

4 LDL ALGORITHMS

We follow three strategies to design algorithms for LDL. The first strategy is problem transformation, i.e., transform the LDL problem into existing learning paradigms. The second strategy is algorithm adaptation, i.e., extend existing learning algorithms to deal with label distributions. The first two strategies are based on existing machine learning algorithms, which were also adopted in some other work [3], [8], [7]. The third strategy is to design specialized algorithms according to the characteristics of LDL. Following each of the three strategies, two typical algorithms are proposed in this section.

4.1 Problem Transformation

One straightforward way to transform an LDL problem into an SLL problem is to change the training examples into weighted single-label examples. In detail, each training example $(\boldsymbol{x}_i, D_i)$ is transformed into c single-label examples $(\boldsymbol{x}_i, y_j)$ with the weight $d_{\boldsymbol{x}_i}^{y_j}$, where $i = 1, \dots, n$ and $j = 1, \dots, c$. The training set is then resampled to the same size according to the weight of each example. The resampled training set becomes a standard single-label training set including $c \times n$ examples, and then any SLL algorithms can be applied to the training set. Note that although in the resampling step, one training instance with a label distribution is transformed into multiple instances, the resulting training set does not form a multi-instance learning (MIL) [13] or multi-instance multi-label learning (MIML) [52] problem. For MIL and MIML, the training set is composed by many bags each containing multiple instances. A bag is annotated by a label (MIL) or a label set (MIML), meaning at least one instance in the bag can be annotated by the label or the label set. But for each particular instance in the bag, its label is still unknown. On the other hand, the training set transformed from LDL training examples via resampling is a standard single-label training set. Each instance in this set is explicitly assigned with a label. Moreover, although the number of the training examples is increased from n to $c \times n$, it does not affect the complexity and scalability of the learning process much because the label side has been simplified from a label distribution (c elements) to a single label (only 1 element).

In order to predict the label distribution of a previously unseen instance $\boldsymbol{x}$, the learner must be able to output the confidence/probability for each label y_j, which can be regarded as the description degree of the corresponding label, i.e., $d_{\boldsymbol{x}}^{y_j} = P(y_j|\boldsymbol{x})$. Two representative algorithms are adopted here for this purpose. One is the Bayes classifier, the other is SVM. In detail, the Bayes classifier assumes Gaussian distribution for each class, and the posterior probability computed by the Bayes rule is regarded as the description degree of the corresponding label. As to SVM, the probability estimates are obtained by a pairwise coupling multi-class method [47], where the probability of each binary SVM is calculated by an improved implementation of Platt's posterior probabilities [28], and the class probability estimates are obtained by solving a linear system whose solution is guaranteed by the theory in finite Markov Chains. When Bayes and SVM are applied to the resampled training set, the resulted methods are denoted by PT-Bayes and PT-SVM, respectively, where 'PT' is the abbreviation of 'Problem Transformation'.

4.2 Algorithm Adaptation

Certain existing learning algorithms can be naturally extended to deal with label distributions, among which two adapted algorithms are proposed here. The first one is k-NN. Given a new instance $\boldsymbol{x}$, its k nearest neighbors are first found in the training set. Then, the mean of the label distributions of all the k nearest neighbors is calculated as the label distribution of $\boldsymbol{x}$, i.e.,

$$p(y_j|\boldsymbol{x}) = \frac{1}{k} \sum_{i \in N_k(\boldsymbol{x})} d_{\boldsymbol{x}_i}^{y_j}, (j = 1, 2, \dots, c), \qquad (4)$$

where $N_k(\boldsymbol{x})$ is the index set of the k nearest neighbors of $\boldsymbol{x}$ in the training set. This adapted algorithm is denoted by AA-kNN, where 'AA' is the abbreviation of 'Algorithm Adaptation'.

The second algorithm is the backpropagation (BP) neural network. The three-layer neural network has q (the dimensionality of $\boldsymbol{x}$) input units which receive $\boldsymbol{x}$, and c (the number of different labels) output units each of which outputs the description degree of a label y_j. For SLL or MLL, the desired output $\boldsymbol{t}$ is usually a vector with '1's at the positions corresponding to the positive labels of the input instance, and '0's otherwise. For LDL, $\boldsymbol{t}$ becomes the real label distribution of the input training instance. Thus, the target of the BP algorithm is to minimize the sum-squared error of the output of the neural network compared with the real label distributions. To make sure the output of the neural network $\boldsymbol{z} = \{z_1, z_2, \ldots, z_c\}$ satisfies that $z_j \in [0, 1]$ for $j = 1, 2, \ldots, c$ and $\sum_j z_j = 1$, the softmax activation function is used in each output unit. Let the net input to the j-th output unit be η_j, then the softmax output z_j is

$$z_j = \frac{\exp(\eta_j)}{\sum\limits_{k=1}^{c} \exp(\eta_k)}, (j = 1, 2, \ldots, c), \tag{5}$$

This adapted algorithm is denoted by AA-BP.

4.3 Specialized Algorithms

Different from the indirect strategy of problem transformation and algorithm adaptation, the specialized algorithms directly match the LDL problem, e.g., by directly solving the optimization problem in Eq. (1). One good start toward this end might be our previous work on facial age estimation [19], [20], where in order to solve the insufficient training data problem (the numbers of face images for some ages are small), each face image is labeled by not only its chronological age, but also the neighboring ages (close to the chronological age) so that one training face image can contribute to the learning of not only its chronological age, but also the neighboring ages. This actually forms a special label distribution with the highest description degree at the chronological age, and gradually decreasing description degrees on both neighboring sides of the chronological age. We proposed an algorithm IIS-LLD for such special data, where the key step was to solve an optimization problem similar to Eq. (1). Although IIS-LLD was designed for a particular form of label distribution, it can be generalized to deal with any LDL problems. Thus, we rename IIS-LLD in this paper as SA-IIS, where 'SA' is the abbreviation of 'Specialized Algorithm'. The optimization process of SA-IIS, however, has been evidenced to be not very efficient [30]. So, an improved version is further proposed in this section.

SA-IIS assumes the parametric model $p(y|\boldsymbol{x}; \boldsymbol{\theta})$ to be the maximum entropy model [6], i.e.,

$$p(y|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{1}{Z} \exp\left(\sum_k \theta_{y,k} g_k(\boldsymbol{x})\right), \tag{6}$$

Algorithm 1: SA-IIS

```
Input: The training set $S = \{(\boldsymbol{x}_i, D_i)\}_{i=1}^{n}$ and the
       convergence criterion $\varepsilon$
Output: $p(y|\boldsymbol{x}; \boldsymbol{\theta})$
Initialize the model parameter vector $\boldsymbol{\theta}^{(0)}$;
$l \leftarrow 0$;
repeat
    $l \leftarrow l + 1$;
    Solve Eq. (8) for $\delta_{y,k}$ by Gauss-Newton method;
    $\boldsymbol{\theta}^{(l)} \leftarrow \boldsymbol{\theta}^{(l-1)} + \boldsymbol{\Delta}$;
until $T(\boldsymbol{\theta}^{(l)}) - T(\boldsymbol{\theta}^{(l-1)}) < \varepsilon$;
$p(y|\boldsymbol{x}; \boldsymbol{\theta}) \leftarrow \frac{1}{Z} \exp\left(\sum_k \theta_{y,k}^{(l)} g_k(\boldsymbol{x})\right)$;
```

where $Z = \sum_y \exp\left(\sum_k \theta_{y,k} g_k(\boldsymbol{x})\right)$ is the normalization factor, $\theta_{y,k}$ is an element in $\boldsymbol{\theta}$, and $g_k(\boldsymbol{x})$ is the k-th feature of $\boldsymbol{x}$. Substituting Eq. (6) into Eq. (1) yields the target function of $\boldsymbol{\theta}$

$$\begin{aligned} T(\boldsymbol{\theta}) \;=\; & \sum_{i,j} d_{\boldsymbol{x}_i}^{y_j} \sum_k \theta_{y_j,k} g_k(\boldsymbol{x}_i) \\ & - \sum_i \ln \sum_j \exp\left(\sum_k \theta_{y_j,k} g_k(\boldsymbol{x}_i)\right). \end{aligned} \tag{7}$$

The optimization of Eq. (7) uses a strategy similar to Improved Iterative Scaling (IIS) [34], a well-known algorithm for maximizing the likelihood of the maximum entropy model. IIS starts with an arbitrary set of parameters. Then for each step, it updates the current estimate of the parameters $\boldsymbol{\theta}$ to $\boldsymbol{\theta} + \boldsymbol{\Delta}$, where $\boldsymbol{\Delta}$ maximizes a lower bound to the change in likelihood $\Omega = T(\boldsymbol{\theta} + \boldsymbol{\Delta}) - T(\boldsymbol{\theta})$. The element of $\boldsymbol{\Delta}$, $\delta_{y_j,k}$, can be obtained by solving the equation

$$\begin{aligned} & \sum_i p(y_j|\boldsymbol{x}_i; \boldsymbol{\theta}) g_k(\boldsymbol{x}_i) \exp(\delta_{y_j,k} s(g_k(\boldsymbol{x}_i)) g^{\#}(\boldsymbol{x}_i)) \\ & \qquad - \sum_i d_{\boldsymbol{x}_i}^{y_j} g_k(\boldsymbol{x}_i) = 0, \end{aligned} \tag{8}$$

where $g^{\#}(\boldsymbol{x}_i) = \sum_k |g_k(\boldsymbol{x}_i)|$ and $s(g_k(\boldsymbol{x}_i))$ is the sign of $g_k(\boldsymbol{x}_i)$. The detailed derivation of Eq. (8) can be found in the Appendix. What is nice about Eq. (8) is that $\delta_{y_j,k}$ appears alone, and therefore can be solved one by one through nonlinear equation solvers, such as the Gauss-Newton method. The pseudocode of SA-IIS is given in Algorithm 1.

It has been reported in the literature [30] that IIS often performs worse than several other optimization algorithms such as conjugate gradient and quasi-Newton methods. Here we follow the idea of an effective quasi-Newton method BFGS [33] to further improve SA-IIS.

Consider the second-order Taylor series of $T'(\boldsymbol{\theta}) = -T(\boldsymbol{\theta})$ at the current estimate of the parameter vector $\boldsymbol{\theta}^{(l)}$:

$$T'(\boldsymbol{\theta}^{(l+1)}) \approx T'(\boldsymbol{\theta}^{(l)}) + \nabla T'(\boldsymbol{\theta}^{(l)})^{\mathrm{T}} \boldsymbol{\Delta} + \frac{1}{2} \boldsymbol{\Delta}^T \boldsymbol{H}(\boldsymbol{\theta}^{(l)}) \boldsymbol{\Delta}, \tag{9}$$

where $\boldsymbol{\Delta} = \boldsymbol{\theta}^{(l+1)} - \boldsymbol{\theta}^{(l)}$ is the update step, $\nabla T'(\boldsymbol{\theta}^{(l)})$ and $\boldsymbol{H}(\boldsymbol{\theta}^{(l)})$ are the gradient and Hessian matrix of $T'(\boldsymbol{\theta})$ at $\boldsymbol{\theta}^{(l)}$, respectively. The minimizer of Eq. (9) is

$$\boldsymbol{\Delta}^{(l)} = -\boldsymbol{H}^{-1}(\boldsymbol{\theta}^{(l)}) \nabla T'(\boldsymbol{\theta}^{(l)}). \tag{10}$$

Algorithm 2: SA-BFGS

```
Input: The training set $S = \{(\boldsymbol{x}_i, D_i)\}_{i=1}^n$ and the
       convergence criterion $\varepsilon$
Output: $p(y|\boldsymbol{x};\boldsymbol{\theta})$
Initialize the model parameter vector $\boldsymbol{\theta}^{(0)}$;
Initialize the inverse Hessian approximation $\boldsymbol{B}^{(0)}$;
Compute $\nabla T'(\boldsymbol{\theta}^{(0)})$ by Eq. (15);
$l \leftarrow 0$;
repeat
    Compute search direction $\boldsymbol{p}^{(l)} \leftarrow -\boldsymbol{B}^{(l)}\nabla T'(\boldsymbol{\theta}^{(l)})$;
    Compute the step length $\alpha^{(l)}$ by a line search procedure to
    satisfy Eq. (12) and (13);
    $\boldsymbol{\theta}^{(l+1)} \leftarrow \boldsymbol{\theta}^{(l)} + \alpha^{(l)}\boldsymbol{p}^{(l)}$;
    Compute $\nabla T'(\boldsymbol{\theta}^{(l+1)})$ by Eq. (15);
    Compute $\boldsymbol{B}^{(l+1)}$ by Eq. (14);
    $l \leftarrow l+1$;
until $\|\nabla T'(\boldsymbol{\theta}^{(l)})\| < \varepsilon$;
$p(y|\boldsymbol{x};\boldsymbol{\theta}) \leftarrow \frac{1}{Z}\exp\left(\sum_k \theta_{y,k}^{(l)} g_k(\boldsymbol{x})\right)$;
```

The line search Newton method uses $\boldsymbol{\Delta}^{(l)}$ as the search direction $\boldsymbol{p}^{(l)} = \boldsymbol{\Delta}^{(l)}$ and updates the parameter vector by

$$\boldsymbol{\theta}^{(l+1)} = \boldsymbol{\theta}^{(l)} + \alpha^{(l)}\boldsymbol{p}^{(l)}, \tag{11}$$

where the step length $\alpha^{(l)}$ is obtained from a line search procedure to satisfy the *strong Wolfe conditions* [33]:

$$T'(\boldsymbol{\theta}^{(l)} + \alpha^{(l)}\boldsymbol{p}^{(l)}) \leq T'(\boldsymbol{\theta}^{(l)}) + c_1\alpha^{(l)}\nabla T'(\boldsymbol{\theta}^{(l)})^{\mathrm{T}}\boldsymbol{p}^{(l)}, \tag{12}$$

$$|\nabla T'(\boldsymbol{\theta}^{(l)} + \alpha^{(l)}\boldsymbol{p}^{(l)})^{\mathrm{T}}\boldsymbol{p}^{(l)}| \leq c_2|\nabla T'(\boldsymbol{\theta}^{(l)})^{\mathrm{T}}\boldsymbol{p}^{(l)}|, \tag{13}$$

where $0 < c_1 < c_2 < 1$.

One problem of the above method is the calculation of the inverse Hessian matrix in each iteration, which is computationally expensive. The idea of BFGS is to avoid explicit calculation of $\boldsymbol{H}^{-1}(\boldsymbol{\theta}^{(l)})$ by approximating it with an iteratively updated matrix $\boldsymbol{B}$ (detailed derivation can be found in [33]):

$$\begin{aligned}\boldsymbol{B}^{(l+1)} = (\boldsymbol{I} - \boldsymbol{\rho}^{(l)}\boldsymbol{s}^{(l)}(\boldsymbol{u}^{(l)})^{\mathrm{T}})\boldsymbol{B}^{(l)}(\boldsymbol{I} - \boldsymbol{\rho}^{(l)}\boldsymbol{u}^{(l)}(\boldsymbol{s}^{(l)})^{\mathrm{T}}) \\ +\boldsymbol{\rho}^{(l)}\boldsymbol{s}^{(l)}(\boldsymbol{s}^{(l)})^{\mathrm{T}},\end{aligned} \tag{14}$$

where $\boldsymbol{s}^{(l)} = \boldsymbol{\theta}^{(l+1)} - \boldsymbol{\theta}^{(l)}$, $\boldsymbol{u}^{(l)} = \nabla T'(\boldsymbol{\theta}^{(l+1)}) - \nabla T'(\boldsymbol{\theta}^{(l)})$, and $\boldsymbol{\rho}^{(l)} = \frac{1}{\boldsymbol{s}^{(l)}\boldsymbol{u}^{(l)}}$. As to the optimization of the target function $T'(\boldsymbol{\theta})$, the computation of BFGS is mainly related to the first-order gradient, which can be obtained through

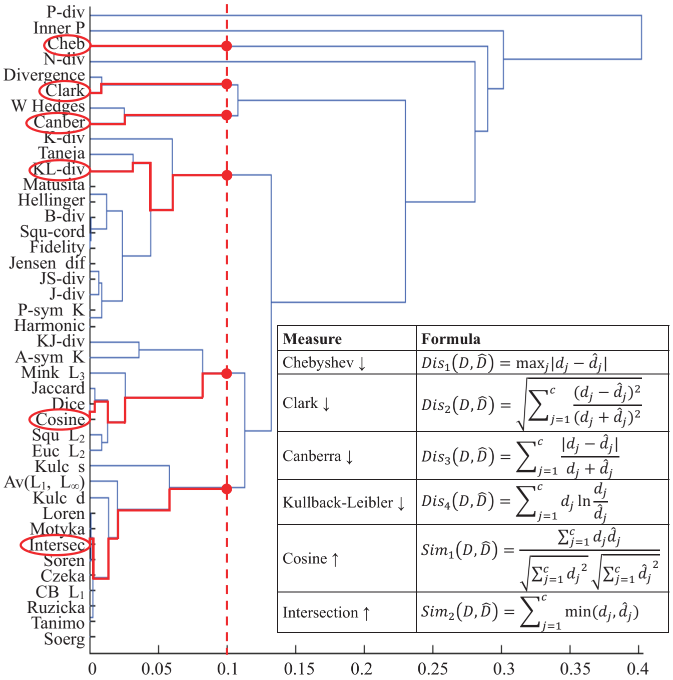
$$\frac{\partial T'(\boldsymbol{\theta})}{\partial \theta_{y_j,k}} = \sum_i \frac{\exp\left(\sum\limits_k \theta_{y_j,k} g_k(\boldsymbol{x}_i)\right) g_k(\boldsymbol{x}_i)}{\sum\limits_j \exp\left(\sum\limits_k \theta_{y_j,k} g_k(\boldsymbol{x}_i)\right)} - \sum_i d_{\boldsymbol{x}_i}^{y_j} g_k(\boldsymbol{x}_i). \tag{15}$$

Thus, it performs much more efficiently than the standard line search Newton method, and based on previous studies [30], it stands a good chance of outperforming the IIS-based algorithm SA-IIS. This improved algorithm is denoted by SA-BFGS, and its pseudocode is shown in Algorithm 2.

5 Experiments

5.1 Evaluation Measures

The output of an LDL algorithm is a label distribution, which is different from both the single label output of SLL and the label set output of MLL. Accordingly, the evaluation measures for LDL algorithms should be different from those used for SLL and MLL algorithms. A natural choice of such measure is the average distance or similarity between the predicted and real label distributions. There are many measures for the distance/similarity between probability distributions which can be well borrowed to measure the distance/similarity between label distributions. For example, Cha [10] performed a semantic similarity analysis on 41 measures for the distance/similarity between distributions from 8 syntactic families. The agglomerative single linkage clustering algorithm [14] was run on all the 41 measures calculated from 30 independent experiments. This resulted in a dendrogram shown in Fig. 3, where the horizontal axis represents the distance between two clusters of distance/similarity measures, and the vertical axis represents different distance/similarity measures.



Measure	Formula
Chebyshev ↓	$Dis_1(D,\widehat{D}) = \max_j\|d_j - \hat{d}_j\|$
Clark ↓	$Dis_2(D,\widehat{D}) = \sqrt{\sum_{j=1}^{c}\frac{(d_j - \hat{d}_j)^2}{(d_j + \hat{d}_j)^2}}$
Canberra ↓	$Dis_3(D,\widehat{D}) = \sum_{j=1}^{c}\frac{\|d_j - \hat{d}_j\|}{d_j + \hat{d}_j}$
Kullback-Leibler ↓	$Dis_4(D,\widehat{D}) = \sum_{j=1}^{c} d_j \ln\frac{d_j}{\hat{d}_j}$
Cosine ↑	$Sim_1(D,\widehat{D}) = \frac{\sum_{j=1}^{c} d_j\hat{d}_j}{\sqrt{\sum_{j=1}^{c} d_j^{\ 2}}\sqrt{\sum_{j=1}^{c} \hat{d}_j^{\ 2}}}$
Intersection ↑	$Sim_2(D,\widehat{D}) = \sum_{j=1}^{c} \min(d_j, \hat{d}_j)$

Fig. 3. Evaluation measure selection from the dendrogram (modified from [10]) for the distribution distance/similarity measures.

On a particular dataset, each of the measures may reflect a certain aspect of an algorithm. It is hard to say which measure is the best. Thus, we propose to use a set of measures when comparing different LDL algorithms, which is analogous to the common practice of using multiple evaluation measures for MLL algorithms. In order to obtain a set of representative and diverse measures, we select the measures from the dendrogram shown in Fig. 3 following four principles: 1. The distance between the clusters of any two measures in the set is greater than 0.1 (indicated by the red dash line in Fig. 3); 2. Each measure in the set comes from a different syntactic family summarized in [10]; 3. The calculation of the selected measures is not prone, except for some extreme cases, to unstable situations such as division by zero or logarithm of zero; 4. The selected measures are relatively widely used in the related areas. As shown in Fig. 3, six measures are finally selected in this

TABLE 1
Statistics of the 16 Datasets Used in the Experiments

No.	Dataset	# Examples (n)	# Features (q)	# Labels (c)
1	Artificial	500 (train) 40,401 (test)	3	3
2	Yeast-alpha	2,465	24	18
3	Yeast-cdc	2,465	24	15
4	Yeast-elu	2,465	24	14
5	Yeast-diau	2,465	24	7
6	Yeast-heat	2,465	24	6
7	Yeast-spo	2,465	24	6
8	Yeast-cold	2,465	24	4
9	Yeast-dtt	2,465	24	4
10	Yeast-spo5	2,465	24	3
11	Yeast-spoem	2,465	24	2
12	Human Gene	30,542	36	68
13	Natural Scene	2,000	294	9
14	SJAFFE	213	243	6
15	SBU_3DFE	2,500	243	6
16	Movie	7,755	1,869	5

way, i.e., Chebyshev distance (Cheb), Clark distance (Clark), Canberra metric (Canber), Kullback-Leibler divergence (KL-div), cosine coefficient (Cosine), and intersection similarity (Intersec), which belong to the Minkowski family, the χ^2 family, the L_1 family, the Shannon's entropy family, the inner product family, and the intersection family, respectively [10]. The first four are distance measures and the last two are similarity measures. Suppose the real label distribution is $D = \{d_1, d_2, \ldots, d_c\}$, the predicted label distribution is $\hat{D} = \{\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_c\}$, then the formulae of the six measures are summarized in the lower-right corner of Fig. 3, where the "↓" after the distance measures indicates "the smaller the better", and the "↑" after the similarity measures indicates "the larger the better". Considering that the cluster distance between any two selected measures is greater than 0.1, and they all come from different families, the selected measures are significantly different in both syntax and semantics. Thus, they have a good chance, as will be verified in Section 5.4, to reflect different aspects of an LDL algorithm.

5.2 Datasets

There are in total 16 datasets used in the experiments including an artificial toy dataset and 15 real-world datasets[1]. Some basic statistics about these 16 datasets are given in Table 1.

The first dataset is an artificial toy dataset which is generated to show in a direct and visual way whether the LDL algorithms can learn the mapping from the instance to the label distribution. In this dataset, the instance $\boldsymbol{x}$ is of three-dimensional and there are three labels, i.e., $q = 3$ and $c = 3$. The label distribution $D = \{d_{\boldsymbol{x}}^{y_1}, d_{\boldsymbol{x}}^{y_2}, d_{\boldsymbol{x}}^{y_3}\}$ of an instance $\boldsymbol{x} = [x_1, x_2, x_3]^{\mathrm{T}}$ is created in the following way.

$$\begin{aligned} t_i &= ax_i + bx_i^2 + cx_i^3 + d, i = 1, \ldots, 3, && (16) \\ \psi_1 &= (\boldsymbol{w}_1^{\mathrm{T}}\boldsymbol{t})^2, && (17) \\ \psi_2 &= (\boldsymbol{w}_2^{\mathrm{T}}\boldsymbol{t} + \lambda_1\psi_1)^2, && (18) \\ \psi_3 &= (\boldsymbol{w}_3^{\mathrm{T}}\boldsymbol{t} + \lambda_2\psi_2)^2, && (19) \\ d_{\boldsymbol{x}}^{y_i} &= \frac{\psi_i}{\psi_1 + \psi_2 + \psi_3}, i = 1, \ldots, 3, && (20) \end{aligned}$$

where $\boldsymbol{t} = [t_1, t_2, t_3]^{\mathrm{T}}$. Note that Eq. (18) and (19) deliberately make the description degree of one label depend on those of other labels. The parameters in Eq. (16)-(19) are set as $a = 1$, $b = 0.5$, $c = 0.2$, $d = 1$, $\boldsymbol{w}_1 = [4, 2, 1]^{\mathrm{T}}$, $\boldsymbol{w}_2 = [1, 2, 4]^{\mathrm{T}}$, $\boldsymbol{w}_3 = [1, 4, 2]^{\mathrm{T}}$, and $\lambda_1 = \lambda_2 = 0.01$. To generate the training set, each component of $\boldsymbol{x}$ is uniformly sampled within the range $[-1, 1]$. In total, there are 500 instances sampled in this way. Then, the label distribution corresponding to each instance is calculated via Eq. (16)-(20). Such 500 examples are used as the training set for the LDL algorithms proposed in Section 4.

In order to show the result of the LDL algorithms in a direct and visual way, the test examples of the toy dataset are selected from a certain manifold in the instance space. The first two components of the test instance $\boldsymbol{x}$, x_1 and x_2, are located at a grid of the interval 0.01 within the range $[-1, 1]$ on both dimensions, i.e., there are in total $201 \times 201 = 40,401$ test instances. The third component x_3 is calculate by

$$x_3 = \sin((x_1 + x_2) \times \pi). \qquad (21)$$

Then, the label distribution of each test instance, either the ground-truth calculated via Eq. (16)-(20) or the prediction given by the LDL algorithms, is transformed into a color (details in Section 5.4.1). Thus the ground-truth and predicted label distributions of the test instances can be compared visually through the color pattern on the manifold.

The second to the eleventh datasets (from *Yeast-alpha* to *Yeast-spoem*) are real-world datasets collected from ten biological experiments on the budding yeast *Saccharomyces cerevisiae* [15]. Each dataset records the result of one experiment. There are in total $2,465$ yeast genes included, each of which is represented by an associated phylogenetic profile vector of the length 24. For each dataset, the labels correspond to the discrete time points during one biological experiment. The gene expression level (after normalization) at each time point provides a natural measure of the description degree of the corresponding label. The number of labels in the ten Yeast Gene datasets is summarized in Table 1. The description degrees (normalized gene expression levels) of all the labels (time points) constitute a label distribution for a particular yeast gene.

The twelfth dataset *Human Gene* is a large-scale real-world dataset collected from the biological research on the relationship between human genes and diseases. There are in total $30,542$ human genes included in this dataset, each of which is represented by the 36 numerical descriptors for a gene sequence proposed in [49]. The labels correspond to 68 different diseases. The gene expression level (after normalization) for each disease is regarded as the description degree of the corresponding label. The description degrees (normalized gene expression level) of all the 68 labels (diseases) constitute a label distribution for a particular human gene.

The thirteenth dataset *Natural Scene* results from the inconsistent multilabel rankings of $2,000$ natural scene images. There are nine possible labels associated with these images, i.e., plant, sky, cloud, snow, building, desert, mountain, water, and sun. Ten human rankers are requested to label the images. For each image, they first select from the nine candidate labels

[1] The datasets and the Matlab code of the LDL algorithms are available at our web site: http://cse.seu.edu.cn/PersonalPage/xgeng/LDL/index.htm

what they think are relevant to the image, and then rank the relevant labels in descending order of relevance to the image. Each human ranker makes his/her decisions independently, and the resulting multilabel rankings are expectably highly inconsistent. Then, the inconsistent rankings for each image are transformed into a label distribution by a nonlinear programming process [18], which finds the common label distribution that is most compatible with all personal rankings. Finally, for each image, a 294-dimensional feature vector is extracted by the method proposed in [8].

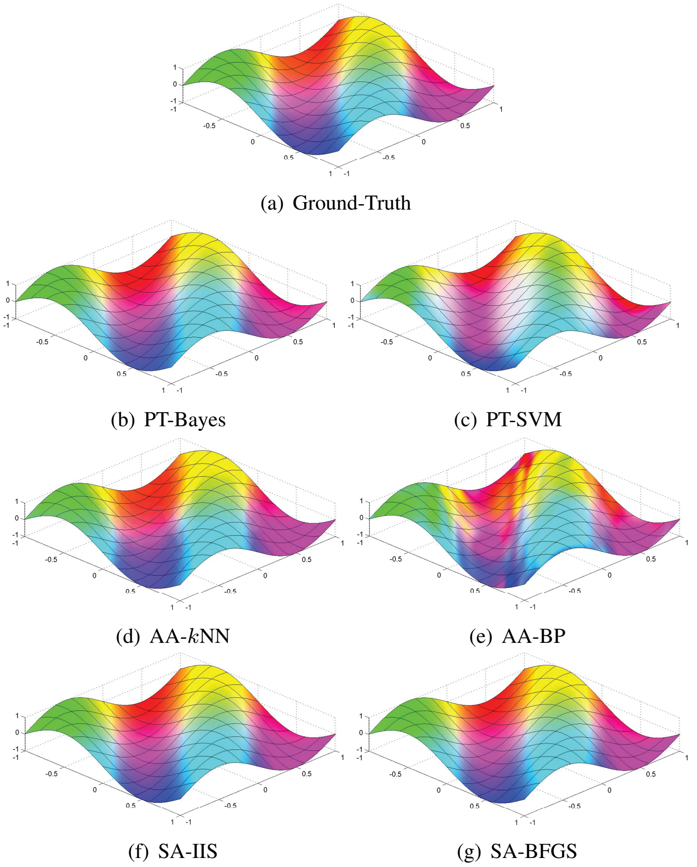
The fourteenth and fifteenth datasets are extensions of two widely used facial expression image databases, i.e., JAFFE [29] and BU_3DFE [48]. The JAFFE database contains 213 grayscale expression images posed by 10 Japanese female models. A 243-dimensional feature vector is extracted from each image by the method of Local Binary Patterns (LBP) [1]. Each image is scored by 60 persons on the 6 basic emotion labels (i.e., happiness, sadness, surprise, fear, anger, and disgust) with a 5-level scale. The average score of each emotion is used to represent the emotion intensity. Instead of only considering the emotion with the highest score as most work on JAFFE does, the dataset *SJAFFE* (Scored JAFFE) used in this paper keeps all the scores and normalizes them into a label distribution over all the 6 emotion labels. Similarly, for the bigger database BU_3DFE containing $2,500$ facial expression images, each image is scored by 23 persons in the same way JAFFE is scored, resulting in the label distribution version of the dataset *SBU_3DFE* (Scored BU_3DFE).

Finally, the sixteenth dataset is about the user ratings on movies. The dataset includes $7,755$ movies and $54,242,292$ ratings from $478,656$ different users. The ratings come from Netflix, which are on a scale from 1 to 5 integral stars (5 labels). The rating label distribution is calculated for each movie as the percentage of each rating level. The features of the movie are extracted from the metadata such as genre, director, actor, country, budget, etc. Categorical attributes are transformed into binary vectors. The final feature vector extracted from each movie is of $1,869$-dimensional.

5.3 Methodology

All the six algorithms described in Section 4, i.e., PT-Bayes, PT-SVM, AA-kNN, AA-BP, SA-IIS, and SA-BFGS, are applied to the 16 datasets shown in Table 1 and compared by the six measures listed in Fig. 3. On the 15 real-world datasets, ten-fold cross validation is conducted for each algorithm and the mean value and standard deviation of each evaluation measure are recorded.

For each algorithm, several parameter configurations are tried. On the artificial dataset, one fifth of the training set is randomly selected as the validation set. The model is trained on those examples left in the training set and tested on the validation set to select the best parameters. Then, the model is trained on the whole training set with the best parameters and tested on the test set. On the real-world datasets, the parameter selection process is nested into the ten-fold cross validation. In detail, the whole data is first randomly split into 10 chunks. Each time, one chunk is used as test set, another is used as validation set, and the rest 8 chunks are used as training set. Then, the model is trained with different parameter settings on the training set and tested on the validation set. This procedure is repeated 10 times (each time with different training and validation sets), and the parameter setting with the best average performance is selected. After that, the original validation set is merged into the training set and the test set remains unchanged. The model is trained with the selected parameter setting on the updated training set and tested on the test set. This procedure is repeated 10 times and the average performance is recorded.



(a) Ground-Truth

(b) PT-Bayes

(c) PT-SVM

(d) AA-kNN

(e) AA-BP

(f) SA-IIS

(g) SA-BFGS

Fig. 4. Comparison between the ground-truth and predicted label distributions (regarded as RGB colors) on the artificial test manifold.

5.4 Results

5.4.1 Artificial Dataset

In order to visually show the results of the LDL algorithms on the artificial dataset, the description degrees of the three labels are regarded as the three color channels of the RGB color space, respectively. In this way, the color of a point in the instance space will visually represent its label distribution. Thus, the predictions made by the LDL algorithms can be compared with the ground-truth label distributions through observing the color patterns on the manifold where the test examples lie on. For easier comparison, the images are visually enhanced by applying a decorrelation stretch process. The results are shown in Fig. 4. It can be seen that the two specialized LDL algorithms, SA-IIS and SA-BFGS, predict

TABLE 2
Experimental Results on the Artificial Dataset

Criterion	PT-Bayes	PT-SVM	AA-kNN	AA-BP	SA-IIS	SA-BFGS
Chebyshev ↓	0.080(3)	0.653(6)	0.086(4)	0.101(5)	0.0767(2)	**0.0766(1)**
Clark ↓	**0.341(1)**	1.135(6)	0.382(4)	0.520(5)	0.349(2)	0.352(3)
Canberra ↓	**0.488(1)**	1.823(6)	0.564(4)	0.699(5)	0.489(2)	0.495(3)
Kullback-Leibler ↓	0.030(3)	1.482(6)	0.035(4)	0.066(5)	**0.029(1)**	0.030(2)
Cosine ↑	0.990(3)	0.377(6)	0.989(4)	0.983(5)	0.99116(2)	**0.99120(1)**
Intersection ↑	0.920(3)	0.347(6)	0.914(4)	0.899(5)	0.9233(2)	**0.9234(1)**
Avg. Rank	2.33	6.00	4.00	5.00	1.83	1.83
Running Time (ms)	22 / 45	391 / 1,153	0 / 79,961	101,568 / 149	1,168 / 33	187 / 33

almost identical color patterns with the ground-truth. PT-Bayes and AA-kNN can also discover similar color patterns with the ground-truth. However, PT-SVM and AA-BP fail to obtain a reasonable result.

For quantitative analysis, the six measures listed in Fig. 3 calculated from the LDL predictions on the artificial dataset are given in Table 2. On each measure, the algorithms are ranked in decreasing order of their performance, and the best performance is highlighted by boldface. The ranks are given in the parentheses right after the measure values, and the average ranks are given in the second last row of the table. The running time (train / test) on an Intel Core i7 3.4GHz workstation is given in the last row of the table. Note that since the training set and the test set of the artificial dataset are fixed (see Table 1) in order to visualize in Fig. 4 the LDL predictions on exactly the same test set, each LDL algorithm only runs once on the test set. So there is no record of standard deviation as in the ten-fold cross validation on the real-world datasets shown in Table 4 to Table 9.

The first observation from Table 2 is that the performances of the best three algorithms (SA-IIS, SA-BFGS and PT-Bayes) evaluated by different measures are quite different. This proves the effectiveness of the selection process described in Section 5.1 to obtain a set of representative and diverse measures. Each of the six measures reflects a certain aspect of the compared LDL algorithms. For example, PT-Bayes appears very good on the measures 'Clark' and 'Canberra' because it has been evidenced that such measures are sensitive to small changes near zero [21], and the Gaussian assumption in PT-Bayes makes it less apt to produce near-zero outputs than SA-IIS and SA-BFGS.

The second observation from Table 2 is that the quantitative comparison results are consistent with the visual comparison results in Fig. 4. Based on the average ranks, the performances of the six LDL algorithms on the artificial dataset can be ranked as SA-BFGS $=$ SA-IIS $\succ$ PT-Bayes $\succ$ AA-kNN $\succ$ AA-BP $\succ$ PT-SVM. The specialized algorithms (SA-IIS and SA-BFGS) achieve the best performance since they directly aim to minimize the distance between the predicted and real label distributions (in fact, their optimization target is one of the measures in Fig. 3, i.e., the Kullback-Leibler divergence). PT-Bayes also achieves relatively good performance since the Gaussian assumption on each class could match the simple case of the toy data well. AA-kNN can also get reasonable result since it is extended from the stable training-free algorithm kNN. Finally, the poor performance of PT-SVM and AA-BP is possibly due to overfitting since they have more parameters to learn and the training set (500 examples) is relatively small compared to the test set (40,401 examples) and they are sampled from very different distributions.

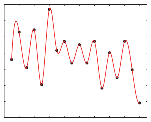
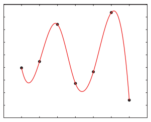
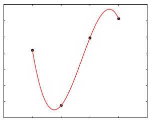
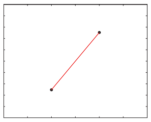
As for the running time, the training time of AA-BP is the longest due to slow convergence of the neural network. AA-kNN does not need to train but costs the most test time. SA-BFGS and SA-IIS cost the same minimum test time because they share the same parametric model. But consistent to the remarks in [30], the training of SA-BFGS is much faster than that of SA-IIS. In general, SA-BFGS and PT-Bayes are the most efficient LDL algorithms in Table 2.

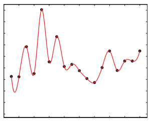
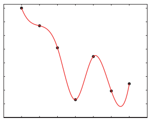
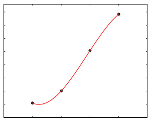
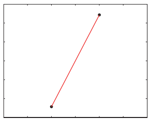
5.4.2 Real-world Datasets

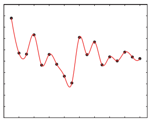
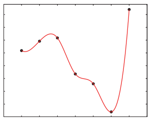
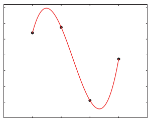
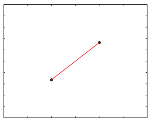
To give a direct idea of the LDL predictions on the real-world datasets, some typical examples of the predicted label distributions by the six LDL algorithms are shown in Table 3. The first row shows the real label distributions of four typical test instances with 18, 7, 4 and 2 labels, which come from the 2nd, 5th, 9th, and 11th datasets in Table 1, respectively. Each of the following rows shows the corresponding predictions of one LDL algorithm. The evaluation measures shown in Fig. 3 are given under each predicted label distribution in two sets. The upper set includes the four distances {Chebyshev, Clark, Canberra, Kullback-Leibler}, and the lower set includes the two similarities {cosine, intersection}. The best performance for each case is highlighted by boldface. As can be seen that the specialized algorithms (SA-IIS and SA-BFGS) can generally give better label distribution predictions compared with other algorithms, in the sense of either visual similarity or quantitative evaluation measures. It is worth mentioning that the order of labels will affect the shape of the label distribution, but not the correlations among the labels. As long as the label orders are the same in the training and test sets, it will not affect, at least for the algorithms proposed in this paper, the effectiveness of LDL.

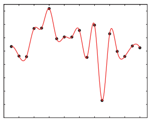
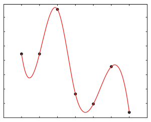
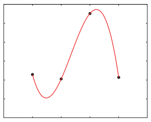
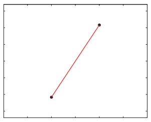
Table 4 to Table 9 tabulate the results of the six LDL algorithms on the 15 real-world datasets (the 2nd to 16th datasets in Table 1) evaluated by the six measures Chebyshev distance, Clark distance, Canberra metric, Kullback-Leibler divergence, cosine coefficient, and intersection similarity, respectively. Since the LDL algorithms are tested via ten-fold cross validation, the performance is represented by "mean ± standard deviation" of the corresponding measure calculated during the ten-fold cross validation. In each table, the best performance on each dataset is highlighted by boldface. The LDL algorithms are ranked in decreasing order of their performance on each dataset. The ranks are given in the parentheses

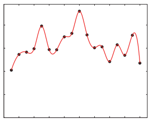
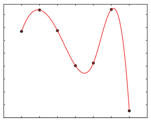
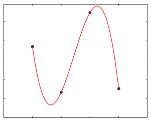
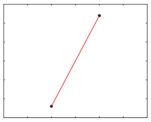
TABLE 3
Typical Examples of the Real and Predicted Label Distributions

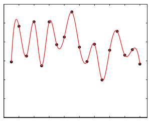
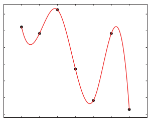
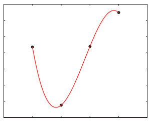
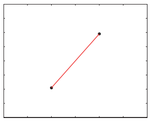
Real	 18 Labels	 7 Labels	 4 Labels	 2 Labels

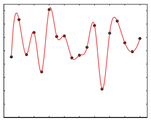
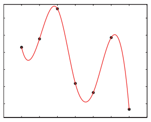
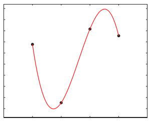
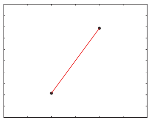
PT-Bayes	
	{0.048, 0.552, 1.885, 0.036}
	{0.960 0.891}
		
		{0.042, 0.230, 0.492, 0.016}
		{0.984, 0.929}
			
			{0.170, 0.516, 0.911, 0.128}
			{0.907, 0.786}
				
				{0.192, 0.310, 0.421, 0.086}
				{0.939, 0.808}

PT-SVM	
	{0.006, 0.105, 0.368, 0.0012}
	{0.9988, 0.980}
		
		{0.040, 0.190, 0.359, 0.011}
		{0.990, 0.949}
			
			{0.041, 0.108, 0.188, 0.006}
			{0.995 0.954}
				
				{0.018, 0.025, 0.036, 0.001}
				{0.999, 0.982}

AA-kNN	
	{0.005, 0.091, 0.282, 0.0009}
	{0.9991, 0.9844}
		
		{0.028, 0.151, 0.381, 0.007}
		{0.993, 0.945}
			
			{0.030, 0.080, 0.124, 0.003}
			{0.997, 0.970}
				
				{0.058, 0.085, 0.118, 0.007}
				{0.994, 0.942}

AA-BP	
	{0.008, 0.156, 0.583, 0.003}
	{0.997, 0.968}
		
		{0.014, 0.076, 0.172, 0.002}
		{0.998, 0.975}
			
			{0.036, 0.085, 0.145 0.004}
			{0.996, 0.964}
				
				{0.069, 0.102, 0.143, 0.010}
				{0.991, 0.931}

SA-IIS	
	{**0.004**, **0.083**, 0.283, **0.00077**}
	{**0.99923**, 0.9843}
		
		{0.013, 0.072, 0.160, 0.002}
		{0.998, 0.977}
			
			{0.016, 0.042 0.067 0.0008}
			{0.9992, 0.984}
				
				{0.012, 0.018, 0.025, 0.0003}
				{0.9997, 0.988}

SA-BFGS	
	{0.006, 0.086, **0.260**, 0.00081}
	{0.99919, **0.986**}
		
		{**0.012**, **0.056**, **0.120**, **0.001**}
		{**0.999**, **0.983**}
			
			{**0.014**, **0.034**, **0.055**, **0.0006**}
			{**0.9994**, **0.987**}
				
				{**0.007**, **0.010**, **0.014**, **0.0001**}
				{**0.9999**, **0.993**}

right after the performance values. The average rank of each algorithm over all the datasets is also calculated and given in the last row of each table.

As can be seen from Table 4 to Table 9, for each particular dataset, the rankings of the six LDL algorithms are often different on different measures. This corresponds to the design of a diverse set of evaluation measures described in Section 5.1. Thus, when comparing two LDL algorithms on a particular dataset, all the six measures should be simultaneously considered. When looking at the average ranks over all the 15 real-world datasets, the rankings of the algorithms on five measures (Clark, Canberra, Kullback-Leibler, cosine, and intersection) are very consistent, i.e., SA-BFGS $\succ$ SA-IIS $\succ$ AA-kNN $\succ$ PT-SVM $\succ$ AA-BP$\succ$ PT-Bayes. The specialized algorithms (SA-BFGS and SA-IIS) generally perform better than those transformed from traditional learning algorithms (PT-Bayes, PT-SVM, AA-kNN, and AA-BP) because they directly aim to minimize the distance between the predicted and real label distributions. Moreover, SA-BFGS improves the performance of SA-IIS and achieves the best performance in most cases by using more effective optimization process. AA-BP and PT-Bayes perform worse than their siblings (AA-kNN and PT-SVM, respectively) because the many parameters in the BP neural network makes AA-BP vulnerable to overfitting, and the Gaussian distributions assumption for PT-Bayes might be inappropriate for the complex real-world datasets. Finally, AA-kNN performs better than PT-SVM because AA-kNN keeps the label distribution and thus keeps the overall labelling structure for each instance, while PT-SVM breaks down the original label distributions by weighted resampling. The average ranking on the Chebyshev distance, however, is slightly different in that the average ranks of PT-SVM and AA-BP are the same. Note that the Chebyshev distance only cares about the worst match over the whole label distribution (see Fig. 3). The transformation from an LDL training set into an SLL training set for the 'PT' algorithms (PT-SVM

TABLE 4
Experimental Results (mean±std(rank)) on the Real-world Datasets Measured by Chebyshev Distance ↓

Dataset	PT-Bayes	PT-SVM	AA-kNN	AA-BP	SA-IIS	SA-BFGS
Yeast-alpha	0.174±0.011(6)	0.017±0.002(4)	0.0147±0.001(2)	0.036±0.003(5)	0.0148±0.001(3)	**0.013±0.001(1)**
Yeast-cdc	0.172±0.016(6)	0.020±0.003(4)	0.0177±0.001(2)	0.037±0.002(5)	0.0178±0.001(3)	**0.016±0.001(1)**
Yeast-elu	0.166±0.014(6)	0.019±0.001(4)	0.0177±0.0005(2)	0.036±0.002(5)	0.0178±0.001(3)	**0.016±0.001(1)**
Yeast-diau	0.167±0.007(6)	0.046±0.004(4)	0.0392±0.001(3)	0.048±0.003(5)	0.0386±0.001(2)	**0.037±0.002(1)**
Yeast-heat	0.176±0.018(6)	0.046±0.001(4)	0.045±0.001(3)	0.052±0.003(5)	0.043±0.001(2)	**0.042±0.001(1)**
Yeast-spo	0.178±0.009(6)	0.065±0.006(4)	0.064±0.002(3)	0.067±0.005(5)	0.060±0.004(2)	**0.058±0.004(1)**
Yeast-cold	0.177±0.011(6)	0.0574±0.003(5)	0.055±0.002(3)	0.0572±0.003(4)	0.053±0.002(2)	**0.051±0.002(1)**
Yeast-dtt	0.177±0.010(6)	0.040±0.001(4)	0.0392±0.001(3)	0.043±0.002(5)	0.0388±0.001(2)	**0.036±0.001(1)**
Yeast-spo5	0.211±0.011(6)	0.0929±0.006(3)	0.096±0.005(5)	0.094±0.006(4)	0.0928±0.006(2)	**0.091±0.005(1)**
Yeast-spoem	0.190±0.017(6)	0.091±0.005(4)	0.093±0.004(5)	0.0892±0.005(3)	0.0891±0.005(2)	**0.087±0.005(1)**
Human Gene	0.195±0.085(6)	0.054±0.004(3)	0.065±0.005(5)	0.059±0.004(4)	0.0534±0.004(2)	**0.0533±0.004(1)**
Natural Scene	0.407±0.027(5)	0.414±0.036(6)	0.374±0.013(4)	0.335±0.016(2)	0.341±0.017(3)	**0.322±0.017(1)**
s-JAFFE	0.121±0.016(4)	0.127±0.017(5)	0.114±0.017(2)	0.130±0.017(6)	0.117±0.015(3)	**0.107±0.015(1)**
s-BU_3DFE	0.116±0.004(5)	0.119±0.006(6)	0.103±0.003(2)	0.113±0.005(4)	0.111±0.004(3)	**0.088±0.003(1)**
Movie	0.199±0.009(5)	0.213±0.039(6)	0.154±0.005(3)	0.157±0.013(4)	0.150±0.008(2)	**0.136±0.010(1)**
Avg. Rank	5.67	4.40	3.13	4.40	2.40	1.00

TABLE 5
Experimental Results (mean±std(rank)) on the Real-world Datasets Measured by Clark Distance ↓

Dataset	PT-Bayes	PT-SVM	AA-kNN	AA-BP	SA-IIS	SA-BFGS
Yeast-alpha	1.729±0.076(6)	0.277±0.031(4)	0.232±0.012(2)	0.711±0.054(5)	0.233±0.012(3)	**0.210±0.014(1)**
Yeast-cdc	1.491±0.062(6)	0.260±0.029(4)	0.237±0.014(3)	0.568±0.037(5)	0.235±0.012(2)	**0.216±0.013(1)**
Yeast-elu	1.379±0.082(6)	0.234±0.015(4)	0.218±0.005(3)	0.505±0.034(5)	0.216±0.007(2)	**0.199±0.005(1)**
Yeast-diau	0.771±0.032(6)	0.246±0.014(4)	0.212±0.004(3)	0.263±0.017(5)	0.209±0.007(2)	**0.200±0.009(1)**
Yeast-heat	0.670±0.047(6)	0.198±0.007(4)	0.195±0.005(3)	0.228±0.015(5)	0.188±0.003(2)	**0.182±0.003(1)**
Yeast-spo	0.695±0.029(6)	0.273±0.024(4)	0.271±0.011(3)	0.292±0.022(5)	0.255±0.017(2)	**0.250±0.017(1)**
Yeast-cold	0.484±0.032(6)	0.155±0.008(4)	0.150±0.007(3)	0.155±0.009(5)	0.144±0.005(2)	**0.139±0.005(1)**
Yeast-dtt	0.480±0.029(6)	0.108±0.005(4)	0.106±0.004(3)	0.118±0.007(5)	0.105±0.004(2)	**0.098±0.004(1)**
Yeast-spo5	0.438±0.027(6)	0.187±0.013(3)	0.193±0.011(5)	0.189±0.012(4)	0.187±0.013(2)	**0.184±0.012(1)**
Yeast-spoem	0.315±0.029(6)	0.134±0.008(4)	0.137±0.006(5)	0.1323±0.008(3)	0.1321±0.007(2)	**0.129±0.008(1)**
Human Gene	4.674±0.415(6)	2.139±0.087(3)	2.388±0.109(4)	3.344±0.250(5)	2.123±0.088(2)	**2.111±0.086(1)**
Natural Scene	2.523±0.027(5)	2.557±0.045(6)	**1.418±0.057(1)**	2.458±0.023(3)	2.461±0.025(4)	2.411±0.023(2)
s-JAFFE	0.430±0.035(4)	0.457±0.039(5)	0.410±0.050(2)	0.510±0.054(6)	0.419±0.034(3)	**0.399±0.044(1)**
s-BU_3DFE	0.467±0.009(4)	0.494±0.022(6)	0.396±0.006(2)	0.477±0.030(5)	0.416±0.009(3)	**0.367±0.009(1)**
Movie	0.799±0.035(6)	0.797±0.108(5)	0.652±0.023(3)	0.675±0.048(4)	0.591±0.028(2)	**0.589±0.038(1)**
Avg. Rank	5.67	4.27	3.00	4.67	2.33	1.07

TABLE 6
Experimental Results (mean±std(rank)) on the Real-world Datasets Measured by Canberra Metric ↓

Dataset	PT-Bayes	PT-SVM	AA-kNN	AA-BP	SA-IIS	SA-BFGS
Yeast-alpha	6.382±0.305(6)	0.921±0.107(4)	0.758±0.040(2)	2.352±0.173(5)	0.763±0.042(3)	**0.684±0.046(1)**
Yeast-cdc	4.987±0.222(6)	0.785±0.084(4)	0.717±0.041(3)	1.718±0.110(5)	0.709±0.036(2)	**0.649±0.041(1)**
Yeast-elu	4.461±0.282(6)	0.691±0.047(4)	0.644±0.016(3)	1.488±0.098(5)	0.639±0.019(2)	**0.583±0.016(1)**
Yeast-diau	1.744±0.071(6)	0.528±0.031(4)	0.455±0.011(3)	0.568±0.033(5)	0.449±0.017(2)	**0.431±0.020(1)**
Yeast-heat	1.415±0.102(6)	0.396±0.016(4)	0.392±0.010(3)	0.459±0.031(5)	0.377±0.005(2)	**0.364±0.006(1)**
Yeast-spo	1.473±0.069(6)	0.565±0.049(4)	0.559±0.024(3)	0.599±0.043(5)	0.523±0.034(2)	**0.513±0.035(1)**
Yeast-cold	0.845±0.059(6)	0.267±0.014(4)	0.260±0.013(3)	0.268±0.015(5)	0.249±0.009(2)	**0.240±0.010(1)**
Yeast-dtt	0.846±0.051(6)	0.186±0.008(4)	0.182±0.007(3)	0.204±0.012(5)	0.181±0.005(2)	**0.169±0.005(1)**
Yeast-spo5	0.681±0.038(6)	0.287±0.019(3)	0.297±0.016(5)	0.291±0.018(4)	0.287±0.019(2)	**0.283±0.018(1)**
Yeast-spoem	0.424±0.038(6)	0.187±0.011(4)	0.191±0.008(5)	0.1842±0.011(3)	0.1840±0.010(2)	**0.179±0.011(1)**
Human Gene	34.238±3.634(6)	14.631±0.647(3)	16.283±0.818(4)	22.788±1.841(5)	14.541±0.653(2)	**14.453±0.645(1)**
Natural Scene	7.149±0.109(5)	7.208±0.205(6)	**3.044±0.137(1)**	6.767±0.095(4)	6.765±0.104(3)	6.620±0.097(2)
s-JAFFE	0.904±0.086(4)	0.935±0.074(5)	0.843±0.113(2)	1.046±0.124(6)	0.875±0.086(3)	**0.820±0.103(1)**
s-BU_3DFE	1.116±0.020(5)	1.147±0.064(6)	0.841±0.014(2)	1.051±0.064(4)	0.934±0.022(3)	**0.794±0.019(1)**
Movie	1.547±0.075(6)	1.537±0.216(5)	1.276±0.046(4)	1.269±0.089(3)	**1.137±0.057(1)**	1.138±0.079(2)
Avg. Rank	5.73	4.27	3.07	4.60	2.20	1.13

and PT-Bayes) breaks down the original label distributions by weighted resampling. The unbalanced and unstable resampling process could possibly make the worst case worse (e.g., the label with small description degree is not sampled at all). This explains why the 'PT' algorithms perform slightly worse on the Chebyshev distance.

One apparent difference between the results on the real-world datasets and those on the artificial dataset is that PT-Bayes performs much worse on the real-world datasets than on the artificial dataset. This might be because that the Gaussian assumption for each label may match the relatively simple artificial toy dataset better than those complex real-world datasets. Another noticeable point is that the values of the Clark distance (Table 5) and the Canberra metric (Table 6) on the Human Gene and Natural Scene datasets are significantly larger than those on other datasets, no matter which LDL algorithm is applied. The reason is of two-fold. On the one hand, the two datasets might be inherently more difficult to learn than other datasets, which can be evidenced by the relatively poor performances on them evaluated by other measures. On the other hand, such gap on the Clark distance and Canberra metric is particularly significant due to the definition of these

TABLE 7
Experimental Results (mean±std(rank)) on the Real-world Datasets Measured by Kullback-Leibler Divergence ↓

Dataset	PT-Bayes	PT-SVM	AA-kNN	AA-BP	SA-IIS	SA-BFGS
Yeast-alpha	0.719±0.080(6)	0.009±0.002(4)	0.0066±0.001(2)	0.081±0.011(5)	0.0067±0.001(3)	**0.006±0.001(1)**
Yeast-cdc	0.603±0.073(6)	0.010±0.002(4)	0.0083±0.001(3)	0.060±0.007(5)	0.0082±0.001(2)	**0.007±0.001(1)**
Yeast-elu	0.556±0.071(6)	0.008±0.001(4)	0.0074±0.0004(3)	0.051±0.009(5)	0.0073±0.0005(2)	**0.006±0.0004(1)**
Yeast-diau	0.306±0.036(6)	0.019±0.002(4)	0.015±0.001(3)	0.024±0.004(5)	0.014±0.001(2)	**0.013±0.001(1)**
Yeast-heat	0.255±0.040(6)	0.0148±0.001(4)	0.0145±0.001(3)	0.021±0.004(5)	0.0133±0.0004(2)	**0.0126±0.0005(1)**
Yeast-spo	0.281±0.031(6)	0.0304±0.005(4)	0.0302±0.002(3)	0.034±0.006(5)	0.0254±0.003(2)	**0.0246±0.003(1)**
Yeast-cold	0.208±0.031(6)	0.0147±0.001(4)	0.014±0.001(3)	0.0149±0.002(5)	0.013±0.001(2)	**0.012±0.001(1)**
Yeast-dtt	0.206±0.029(6)	0.0073±0.001(4)	0.0072±0.001(3)	0.009±0.001(5)	0.0070±0.001(2)	**0.006±0.001(1)**
Yeast-spo5	0.214±0.025(6)	0.03010±0.003(3)	0.033±0.003(5)	0.031±0.003(4)	0.03007±0.003(2)	**0.029±0.003(1)**
Yeast-spoem	0.190±0.038(6)	0.0280±0.004(4)	0.0285±0.003(5)	0.026±0.003(3)	0.025±0.003(2)	**0.024±0.003(1)**
Human Gene	1.887±0.766(6)	0.240±0.019(3)	0.301±0.026(4)	0.500±0.068(5)	0.238±0.019(2)	**0.236±0.019(1)**
Natural Scene	3.065±0.487(6)	1.447±0.243(4)	2.767±0.137(5)	0.875±0.029(3)	0.870±0.026(2)	**0.854±0.062(1)**
s-JAFFE	0.074±0.014(4)	0.086±0.016(5)	0.071±0.023(3)	0.113±0.030(6)	0.070±0.012(2)	**0.064±0.016(1)**
s-BU_3DFE	0.079±0.004(4)	0.089±0.007(6)	0.065±0.002(2)	0.085±0.009(5)	0.068±0.004(3)	**0.049±0.002(1)**
Movie	0.953±0.352(6)	0.268±0.079(5)	0.201±0.011(4)	0.179±0.03(3)	**0.137±0.013(1)**	0.140±0.020(2)
Avg. Rank	5.73	4.13	3.40	4.60	2.07	1.07

TABLE 8
Experimental Results (mean±std(rank)) on the Real-world Datasets Measured by Cosine Coefficient ↑

Dataset	PT-Bayes	PT-SVM	AA-kNN	AA-BP	SA-IIS	SA-BFGS
Yeast-alpha	0.743±0.015(6)	0.991±0.002(4)	0.9935±0.001(2)	0.949±0.006(5)	0.9934±0.001(3)	**0.995±0.001(1)**
Yeast-cdc	0.766±0.017(6)	0.991±0.002(4)	0.9920±0.001(3)	0.960±0.004(5)	0.9921±0.001(2)	**0.993±0.001(1)**
Yeast-elu	0.780±0.018(6)	0.992±0.001(4)	0.99286±0.0003(3)	0.965±0.004(5)	0.99290±0.0005(2)	**0.994±0.0004(1)**
Yeast-diau	0.856±0.007(6)	0.982±0.002(4)	0.986±0.001(3)	0.979±0.002(5)	0.987±0.001(2)	**0.988±0.001(1)**
Yeast-heat	0.866±0.015(6)	0.98607±0.001(4)	0.98612±0.001(3)	0.981±0.003(5)	0.987±0.0004(2)	**0.988±0.0005(1)**
Yeast-spo	0.859±0.008(6)	0.971±0.005(4)	0.972±0.002(3)	0.969±0.004(5)	0.976±0.003(2)	**0.977±0.003(1)**
Yeast-cold	0.898±0.008(6)	0.9860±0.001(4)	0.987±0.001(3)	0.9859±0.002(5)	0.988±0.001(2)	**0.989±0.001(1)**
Yeast-dtt	0.897±0.008(6)	0.9930±0.0005(4)	0.9931±0.0005(3)	0.991±0.001(5)	0.9933±0.0004(2)	**0.994±0.0004(1)**
Yeast-spo5	0.893±0.008(6)	0.9732±0.003(3)	0.970±0.003(5)	0.9728±0.003(4)	0.9733±0.003(2)	**0.974±0.003(1)**
Yeast-spoem	0.914±0.011(6)	0.976±0.003(4)	0.975±0.002(5)	0.9777±0.002(3)	0.9780±0.002(2)	**0.979±0.002(1)**
Human Gene	0.456±0.089(6)	0.832±0.011(3)	0.766±0.020(4)	0.726±0.026(5)	0.833±0.011(2)	**0.834±0.011(1)**
Natural Scene	0.559±0.014(5)	0.490±0.082(6)	0.624±0.016(4)	0.697±0.011(3)	0.698±0.008(2)	**0.710±0.017(1)**
s-JAFFE	0.930±0.013(4)	0.920±0.014(5)	0.9337±0.018(3)	0.908±0.019(6)	0.9340±0.012(2)	**0.940±0.015(1)**
s-BU_3DFE	0.924±0.004(5)	0.914±0.006(6)	0.938±0.002(2)	0.926±0.006(4)	0.935±0.004(3)	**0.954±0.002(1)**
Movie	0.850±0.008(5)	0.806±0.061(6)	0.880±0.006(4)	0.895±0.014(3)	0.905±0.008(2)	**0.912±0.010(1)**
Avg. Rank	5.67	4.33	3.33	4.53	2.13	1.00

TABLE 9
Experimental Results (mean±std(rank)) on the Real-world Datasets Measured by Intersection Similarity ↑

Dataset	PT-Bayes	PT-SVM	AA-kNN	AA-BP	SA-IIS	SA-BFGS
Yeast-alpha	0.660±0.016(6)	0.949±0.006(4)	0.9581±0.002(2)	0.877±0.008(5)	0.9577±0.002(3)	**0.962±0.003(1)**
Yeast-cdc	0.681±0.015(6)	0.948±0.006(4)	0.9528±0.003(3)	0.891±0.007(5)	0.9531±0.002(2)	**0.957±0.003(1)**
Yeast-elu	0.695±0.019(6)	0.951±0.003(4)	0.9546±0.001(3)	0.899±0.006(5)	0.9547±0.001(2)	**0.959±0.001(1)**
Yeast-diau	0.764±0.008(6)	0.926±0.004(4)	0.937±0.002(3)	0.922±0.004(5)	0.938±0.002(2)	**0.940±0.003(1)**
Yeast-heat	0.773±0.018(6)	0.935±0.003(4)	0.936±0.002(3)	0.925±0.005(5)	0.938±0.001(2)	**0.940±0.001(1)**
Yeast-spo	0.765±0.010(6)	0.906±0.008(4)	0.908±0.004(3)	0.902±0.007(5)	0.914±0.005(2)	**0.915±0.006(1)**
Yeast-cold	0.802±0.012(6)	0.9339±0.004(5)	0.936±0.003(3)	0.9340±0.004(4)	0.938±0.002(2)	**0.941±0.002(1)**
Yeast-dtt	0.801±0.011(6)	0.954±0.002(4)	0.9549±0.002(3)	0.950±0.003(5)	0.9552±0.001(2)	**0.958±0.001(1)**
Yeast-spo5	0.789±0.011(6)	0.9071±0.006(3)	0.904±0.005(5)	0.906±0.006(4)	0.9072±0.006(2)	**0.909±0.005(1)**
Yeast-spoem	0.810±0.017(6)	0.909±0.005(4)	0.907±0.004(5)	0.9108±0.005(3)	0.9109±0.005(2)	**0.913±0.005(1)**
Human Gene	0.470±0.062(6)	0.781±0.010(3)	0.742±0.014(4)	0.671±0.025(5)	0.783±0.010(2)	**0.784±0.010(1)**
Natural Scene	0.350±0.014(6)	0.364±0.055(5)	0.544±0.018(2)	0.499±0.012(3)	0.487±0.012(4)	**0.548±0.017(1)**
s-JAFFE	0.846±0.016(4)	0.839±0.015(5)	0.855±0.021(2)	0.824±0.022(6)	0.851±0.016(3)	**0.860±0.019(1)**
s-BU_3DFE	0.834±0.003(5)	0.827±0.009(6)	0.872±0.002(2)	0.847±0.008(4)	0.862±0.004(3)	**0.884±0.003(1)**
Movie	0.725±0.011(5)	0.711±0.052(6)	0.780±0.007(4)	0.788±0.015(3)	0.800±0.010(2)	**0.809±0.013(1)**
Avg. Rank	5.73	4.33	3.13	4.47	2.33	1.00

two measures. As mentioned before, the Clark distance and Canberra metric are sensitive to small changes near zero [21]. It can be found from Fig. 3 that when the description degree of the j-th label in the real label distribution, d_j, is close to zero, the j-th additive term in the formula of 'Clark' or 'Canberra' will be close to its maximum value 1. If there are many near-zero description degrees in the real label distribution, then the value of 'Clark' or 'Canberra' tends to be large. For the Human Gene dataset, one gene is usually related to only a few diseases. Similarly, for the Natural Scene dataset, one image is usually relevant to only a few scene classes. Thus, the real label distributions in both of these two datasets contain many near-zero description degrees, which makes the values of 'Clark' and 'Canberra' relatively large.

6 Summary and Discussions

This paper proposes *label distribution learning*, which is a more general learning framework than single-label learning and multi-label learning. It can deal with not only multiple labels of one instance, but also the different importance of these labels. This paper proposes six working LDL algorithms in three ways: problem transformation, algorithm adaptation, and specialized algorithm design. In order to compare these algorithms, six evaluation measures are used and the first batch of label distribution datasets are prepared and made publicly available. Experimental results on one artificial and fifteen real-world datasets show clear advantages of the specialized algorithms. This illustrates that the characteristics of LDL require special design to achieve good performance.

LDL is motivated by the real-world data with natural measures of description degrees (e.g., gene expression level). However, as a general learning framework, LDL might also be used to solve other kinds of problems. Generally speaking, there are at least three scenarios where LDL could be helpful:

1) There is a natural measure of description degree that associates the labels with the instances. This is the most direct application of LDL, as described in this paper.
2) When there are multiple labeling sources (e.g., multiple experts) for one instance, known as *learning with auxiliary information* [31] or *learning from multiple experts* [43], the annotations from different sources might be significantly inconsistent. In such case, it is usually better for the learning algorithm to integrate the labels from all the sources rather than to decide one or more 'winning label(s)' via majority voting [36]. One good way to incorporate all the labeling sources is to generate a label distribution for the instance: the label favored by more sources is given a higher description degree, while that chosen by fewer sources is assigned with a lower description degree. In this way, the multi-labeling-source problem is transformed into a LDL problem.
3) Some labels are highly correlated with other labels. Utilizing such correlation is one of the most important approaches to improve the learning process [24], [45], [40]. LDL provides a new way toward this purpose. The key step is to transform an SLL or MLL problem into an LDL problem. This can be achieved by generating a label distribution for each instance according to the correlation among the labels.

Each of the three scenarios actually covers a vast area of applications. A lot of interesting work, both at the theoretical level and at the application level, may be conducted in the future. In addition, the relationship between LDL and traditional machine learning paradigms, e.g., SLL and MLL, is another promising research direction. For example, we have applied LDL to solve the MLL problems by generating the label distributions via an iterative label propagation process among the training samples [27]. Comprehensive experiments show that the LDL-augmented MLL algorithm could significantly improve the state-of-the-art MLL performance.

Appendix
Derivation of Eq. (8)

The change of $T(\boldsymbol{\theta})$ in Eq. (7) between adjacent steps is

$$T(\boldsymbol{\theta}+\boldsymbol{\Delta})-T(\boldsymbol{\theta})=\sum_{i,j} d_{\boldsymbol{x}_i}^{y_j}\sum_k \delta_{y_j,k}g_k(\boldsymbol{x}_i)- \sum_i \ln\sum_j p(y_j|\boldsymbol{x}_i;\boldsymbol{\theta})\exp\left(\sum_k \delta_{y_j,k}g_k(\boldsymbol{x}_i)\right), \tag{22}$$

where $\delta_{y_j,k}$ is the increment for $\theta_{y_j,k}$. Applying the inequality $-\ln x \geq 1-x$ yields

$$T(\boldsymbol{\theta}+\boldsymbol{\Delta})-T(\boldsymbol{\theta})\geq\sum_{i,j} d_{\boldsymbol{x}_i}^{y_j}\sum_k \delta_{y_j,k}g_k(\boldsymbol{x}_i)+n- \sum_{i,j} p(y_j|\boldsymbol{x}_i;\boldsymbol{\theta})\exp\left(\sum_k \delta_{y_j,k}g_k(\boldsymbol{x}_i)\right). \tag{23}$$

Differentiating the right side of Eq. (23) w.r.t. $\delta_{y_j,k}$ yields the coupled equations of $\delta_{y,k}$ which are hard to be solved. To decouple the interaction among $\delta_{y,k}$, Jensen's inequality is applied here, i.e., for a probability mass function $p(x)$ and another arbitrary function $f(x)$,

$$\exp\left(\sum_x p(x)f(x)\right)\leq\sum_x p(x)\exp\left(f(x)\right). \tag{24}$$

The last term of Eq. (23) can be rewritten as

$$\sum_{i,j} p(y_j|\boldsymbol{x}_i;\boldsymbol{\theta})\exp\left(\sum_k \delta_{y_j,k}s(g_k(\boldsymbol{x}_i))g^{\#}(\boldsymbol{x}_i)\frac{|g_k(\boldsymbol{x}_i)|}{g^{\#}(\boldsymbol{x}_i)}\right), \tag{25}$$

where $g^{\#}(\boldsymbol{x}_i)=\sum_k|g_k(\boldsymbol{x}_i)|$ and $s(g_k(\boldsymbol{x}_i))$ is the sign of $g_k(\boldsymbol{x}_i)$. Since $|g_k(\boldsymbol{x}_i)|/g^{\#}(\boldsymbol{x}_i)$ can be viewed as a probability mass function, Jensen's inequality can be applied to Eq. (23) to yield

$$T(\boldsymbol{\theta}+\boldsymbol{\Delta})-T(\boldsymbol{\theta})\geq\sum_{i,j} d_{\boldsymbol{x}_i}^{y_j}\sum_k \delta_{y_j,k}g_k(\boldsymbol{x}_i)+n- \sum_{i,j} p(y_j|\boldsymbol{x}_i;\boldsymbol{\theta})\sum_k\frac{|g_k(\boldsymbol{x}_i)|}{g^{\#}(\boldsymbol{x}_i)}\exp(\delta_{y_j,k}s(g_k(\boldsymbol{x}_i))g^{\#}(\boldsymbol{x}_i)). \tag{26}$$

Denote the right side of Eq. (26) as $\mathcal{A}(\boldsymbol{\Delta}|\boldsymbol{\theta})$, which is a lower bound to $T(\boldsymbol{\theta}+\boldsymbol{\Delta})-T(\boldsymbol{\theta})$. Setting the derivative of $\mathcal{A}(\boldsymbol{\Delta}|\boldsymbol{\theta})$ w.r.t. $\delta_{y_j,k}$ to zero gives Eq. (8).

Acknowledgment

This research was supported by the National Science Foundation of China (61273300, 61232007), the Jiangsu Natural Science Funds for Distinguished Young Scholar (BK20140022), and the Collaborative Innovation Center of Wireless Communications Technology.

References

[1] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, 2006.

[2] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Portland, OR, 2013, pp. 819–826.

[3] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[4] G. BakIr, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S. Vishwanathan, Eds., *Predicting Structured Data*. Cambridge, MA: The MIT Press, 2007.

[5] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *Advances in Neural Information Processing Systems 23*, Vancouver, Canada, 2010, pp. 163–171.

[6] A. L. Berger, S. D. Pietra, and V. J. D. Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[7] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[8] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[9] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[10] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 1, pp. 300–307, 2007.

[11] W. Cheng, K. Dembczynski, and E. Hüllermeier, "Graded multilabel classification: The ordinal case," in *Proc. 27th Int'l Conf. Machine Learning*, Haifa, Israel, 2010, pp. 223–230.

[12] T. Denoeux and L. M. Zouhal, "Handling possibilistic labels in pattern classification using evidential reasoning," *Fuzzy Sets and Systems*, vol. 122, no. 3, pp. 409–424, 2001.

[13] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artif. Intell.*, vol. 89, no. 1-2, pp. 31–71, 1997.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Malden, MA: Wiley, 2001.

[15] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Science*, vol. 95, no. 25, pp. 14 863–14 868, 1998.

[16] J. Fürnkranz and E. Hüllermeier, Eds., *Preference Learning*. New York, NY: Springer, 2010.

[17] X. Geng and R. Ji, "Label distribution learning," in *Proc. 2013 Int'l Conf. Data Mining Workshops*, Dallas, TA, 2013, pp. 377–383.

[18] X. Geng and L. Luo, "Multilabel ranking with inconsistent rankers," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Columbus, OH, 2014, pp. 3742–3747.

[19] X. Geng, K. Smith-Miles, and Z.-H. Zhou, "Facial age estimation by learning from label distributions," in *Proc. 24th AAAI Conf. Artificial Intelligence*, Atlanta, GA, 2010, pp. 451–456.

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

[21] A. Gordon, *Classification*, 2nd ed. New York, NY: Chapman and Hall/CRC, 1999.

[22] E. Hüllermeier, J. Fürnkranz, W. Cheng, and K. Brinker, "Label ranking by learning pairwise preferences," *Artif. Intell.*, vol. 172, no. 16-17, pp. 1897–1916, 2008.

[23] A. J. Izenman, *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*, 2nd ed. New York, NY: Springer, 2013.

[24] F. Kang, R. Jin, and R. Sukthankar, "Correlated label propagation with application to multi-label learning," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, New York, NY, 2006, pp. 1719–1726.

[25] C. H. Lampert, H. Nickisch, and S. Harmeling, "Attribute-based classification for zero-shot visual object categorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 453–465, 2014.

[26] P. Li, H. Li, and M. Wu, "Multi-label ensemble based on variable pairwise constraint projection," *Information Sciences*, vol. 222, pp. 269–281, 2013.

[27] Y. Li, M. Zhang, and X. Geng, "Leveraging implicit relative labeling-importance information for effective multi-label learning," in *Proc. IEEE Int'l Conf. Data Mining*, Atlantic City, NJ, 2015, pp. 251–260.

[28] H. Lin, C. Lin, and R. C. Weng, "A note on platt's probabilistic outputs for support vector machines," *Machine Learning*, vol. 68, no. 3, pp. 267–276, 2007.

[29] M. J. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, "Coding facial expressions with gabor wavelets," in *Proc. 3rd Int'l Conf. Face & Gesture Recognition*, Nara, Japan, 1998, pp. 200–205.

[30] R. Malouf, "A comparison of algorithms for maximum entropy parameter estimation," in *Proc. 6th Conf. Computational Natural Language Learning*, Taipei, Taiwan, 2002, pp. 49–55.

[31] Q. Nguyen, H. Valizadegan, and M. Hauskrecht, "Learning classification with auxiliary probabilistic information," in *Proc. 11th IEEE Int'l Conf. Data Mining*, Vancouver, Canad, 2011, pp. 477–486.

[32] M. Nickel, V. Tresp, and H. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proc. 28th Int'l Conf. Machine Learning*, Bellevue, WA, 2011, pp. 809–816.

[33] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. New York, NY: Springer, 2006.

[34] S. D. Pietra, V. J. D. Pietra, and J. D. Lafferty, "Inducing features of random fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 380–393, 1997.

[35] B. Quost and T. Denoeux, "Learning from data with uncertain labels by boosting credal classifiers," in *Proc. 1st ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data*, Paris, France, 2009, pp. 38–47.

[36] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *Journal of Machine Learning Research*, vol. 11, no. Apr, pp. 1297–1322, 2010.

[37] J. Read, A. Bifet, G. Holmes, and B. Pfahringer, "Scalable and efficient multi-label classification for evolving data streams," *Machine Learning*, vol. 88, no. 1-2, pp. 243–272, 2012.

[38] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, pp. 333–359, 2011.

[39] P. Smyth, "Learning with probabilistic supervision," in *Computational Learning Theory and Natural Learning System*, T. Petsche, Ed. MA: MIT Press, 1995, vol. III, pp. 163–182.

[40] Y.-Y. Sun, Y. Zhang, and Z.-H. Zhou, "Multi-label learning with weak label," in *Proc. 24th AAAI Conf. Artificial Intelligence*, Atlanta, GA, 2010, pp. 593–598.

[41] G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou, "Tutorial on learning from multi-label data," in *European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Bled, Slovenia, 2009.

[42] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int'l Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[43] H. Valizadegan, Q. Nguyen, and M. Hauskrecht, "Learning classification models from multiple experts," *Journal of Biomedical Informatics*, vol. 46, no. 6, pp. 1125–1135, 2013.

[44] W. Waegeman, K. Dembczynski, and E. Hüllermeier, "Tutorial: Multi-target prediction," in *Proc. 30th Int'l Conf. Machine Learning*, Atlanta, GA, 2013.

[45] H. Wang, M. Huang, and X. Zhu, "A generative probabilistic model for multi-label classification," in *Proc. 8th IEEE Int'l Conf. Data Mining*, Pisa, Italy, 2008, pp. 628–637.

[46] F. Wörgötter and B. Porr, "Temporal sequence learning, prediction, and control: A review of different models and their relation to biological mechanisms," *Neural Computation*, vol. 17, no. 2, pp. 245–319, 2005.

[47] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, vol. 5, pp. 975–1005, 2004.

[48] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato, "A 3d facial expression database for facial behavior research," in *Proc. 7th IEEE Int'l Conf. Automatic Face and Gesture Recognition*, Southampton, UK, 2006, pp. 211–216.

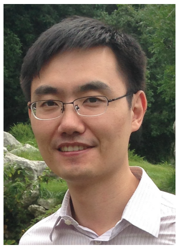
[49] J.-F. Yu, D.-K. Jiang, K. Xiao, Y. Jin, J.-H. Wang, and X. Sun, "Discriminate the falsely predicted proteinccoding genes in aeropyrum pernix k1 genome based on graphical representation," *MATCH Commun. Math. Comput. Chem.*, vol. 67, no. 3, pp. 845–866, 2012.

[50] M.-L. Zhang and Z.-H. Zhou, "Multilabel neural networks with applications to functional genomics and text categorization," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1338–1351, 2006.

[51] ——, "Multi-label learning by instance differentiation," in *Proc. 22nd AAAI Conf. Artificial Intelligence*, Vancouver, Canada, 2007, pp. 669–674.

[52] Z.-H. Zhou and M.-L. Zhang, "Multi-instance multi-label learning with application to scene classification," in *Advances in Neural Information Processing Systems 19*, Vancouver, Canada, 2007, pp. 1609–1616.

[53] H.-J. Zimmermann, Ed., *Practical Applications of Fuzzy Technologies*. Netherlands: Kluwer Academic Publishers, 1999.



Xin Geng (M'13) received the B.Sc. (2001) and M.Sc. (2004) degrees in computer science from Nanjing University, China, and the Ph.D (2008) degree from Deakin University, Australia. He joined the School of Computer Science and Engineering at Southeast University, China, in 2008, and is currently a professor and vice dean of the school. His research interests include pattern recognition, machine learning, and computer vision. He has published over 40 refereed papers and holds 4 patents in these areas.