Research Article

In-Depth Research and Analysis of Multilabel Learning Algorithm

Daowang Li and Canwei Wang
Department of Information and Engineering, Shandong Management University, Jinan 250357, China
Correspondence should be addressed to Daowang Li; lidw@sdmu.edu.cn
Received 1 June 2022; Accepted 14 June 2022; Published 4 July 2022
Academic Editor: Yaxiang Fan

Copyright © 2022 Daowang Li and Canwei Wang. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Multilabel learning (MLL), as a hot topic in the field of machine learning, has attracted wide attention from many scholars due to its ability to express output space polysemy. In recent years, a large number of achievements about MLL have emerged. Among these achievements, there are several typical issues worthy of attention. Firstly, the correlation among labels plays a key role in improving MLL model training process. Many MLL algorithms try to fully and effectively use the correlation among labels to improve the performance. Secondly, existing MLL evaluation metrics, which is different from those in binary classification, often reflects the generalization performance of MLL classifiers in some aspects. How to choose metrics in algorithms to improve their generalization performance and fairness is another issue that should be concerned. Thirdly, in many practical MLL applications, there are many unlabeled instances due to their labeling cost in training datasets. How to use the wealth information contained in the correlation among unlabeled instances may contribute to reducing of the labeling cost in MLL and improving performance. Fourthly, labels assigned to instances may not be equally descriptive in many applications. How to describe the importance of each label in output space to an instance has become one of research points that many scholars have paid attention to in recent years. This paper reviews the MLL-related research results of correlation among labels, evaluation metric, multilabel semisupervised learning, and label distribution learning (LDL) from a theoretical and algorithmic perspective. Finally, the related research work on MLL is summarized and discussed.

1. Introduction

Traditional supervised learning assumes that each sample corresponds to a single class label that is unique in the output space, which is binary classification. But many real-world learning tasks do not fit the hypothesis. For example, an image of Yao Ming smiling while holding a basketball in classification tasks can be labeled “movement,” “basketball,” “NBA superstar,” or “smile.” Another example is the article “Moonlight” in text categorization tasks, which might cover the topics “essay” or “zhu ziqing” labels. In these cases, an example is polysemous. In other words, an example is associated with multiple class labels in output space. In order to solve these problems, there appear multilabel learning frameworks [1].

A multilabel learning framework was originally applied to text categorization tasks. Because of its ability to express the world more accurately, MLL has attracted wide attention from researchers and has been gradually applied in applications such as automatic annotation of multimedia information, Web mining, information retrieval, and tag recommendation [2, 3]. A lot of influential MLL methods have been proposed in the recent years, and more and more detailed and deep-level problems have gradually become clear and solvable. A number of scholars have systematically studied and combed the research progresses towards MLL. Tsoumakas et al. gave a structured representation of sparse literatures on MLL methods with comments on their advantages and disadvantages. Label cardinality and label density were defined to quantify the multiple labels properties of a dataset, and some comparative experimental results were provided to demonstrate the performance of different MLL methods in their work. Zhang et al. providing the fundamentals on MLL including formal definition and evaluation metrics. And in their work, some representative MLL algorithms are highlighted and related analysis is carried out [4]. From the research in recent years, several research hotspots on MLL have attracted the interest of scholars. In
order to inspire the future research for scholars interested in MLL, we combed the relevant works. In this paper, we will focus on the following: (1) the correlation among labels, (b) evaluation metric, (c) multilevel semisupervised learning, and (d) distribution characteristics of labels.

2. The Unified Description of the Relevant Concepts

For the sake of further description, here, we give the formal description on the relevant concepts on MLL.

Assume \( X = \mathbb{R}^d \) expresses a \( d \)-dimensional instance space, in which each instance could be expressed as a vector \( x_i = \{x_{i1}, x_{i2} \cdots x_{id}\} \in X; \ Y = \{y_{i1}, y_{i2} \cdots y_{iq}\} \) expresses \( q \)-dimensional class label space. A multilabel learning framework is to learn a mapping function \( h : X \rightarrow 2^Y \) from instance space \( X \) to the subsets of class label space \( Y \) in given training set \( D = \{(x_1, Y_1):(x_2, Y_2) \cdots (x_m, Y_m)\} \) \((Y_i \subseteq Y\) was a subset of label space \( Y, i \in [1,m]\)), noting the size of \( D \) as \(|D|\). For any unknown instance \( x \in X \) in a testing set, a multilabel classifier will predict a set \( h(x) \subseteq Y \) as a predicted label set of \( x \). In general, most of the multilabel learning model will attune \( h \) to a real-valued function \( f : X \times Y \rightarrow \mathbb{R} \), and \( f(x, y) \) can be expressed as a confidence value with which the MLL model predicts \( y \in Y \) was the label assigned to \( x \). Thus, \( f(x, y) \) will make the label \( y' \) corresponding to larger output as relevant labels of \( x \), while making the label \( y'' \) corresponding to smaller output as irrelevant labels of \( x \), that is, \( f(x, y') > f(x, y'') \). The above-mentioned classification prediction model usually sets a threshold value \( t \) in which cloud divides the label space \( Y \) into two parts: relevant and irrelevant segmentation. If the real value classifier model \( f(x, y) > t \), we can consider \( y \) relevant labels of \( x \); otherwise, \( y \) is considered irrelevant labels of \( x \).

There are usually two approaches to select \( t \): One is for the fixed constant (eg., 0.5); another one is for the variable function [5].

And it is worth mentioning that there are several concepts related to MLL. Multiclass classification deals with the cases in which each instance is associated with only one label, while the label belongs to a label set that consist to more than two labels. For example, an image of fruit could be associated with apple, banana, orange, or peach, but it could not be associated with more than one of them. If we describe binary classification as “True-False,” multiclass classification could be described as a “Single-Answer” from multiple items, and multilabel classification could be described as “Multiple-Answer.” Their relationship is shown in Figure 1

Multinstance learning refers to the cases in which some instances are packed into a bag as an example, and each example is associated with only one binary label. In contrast to MLL which handle the polysemy of labels in output space, multi-instance learning handles the polysemy of examples in an input space [6–8]. What is interesting and important is that if we combine multilabel with multi-instance, we would get a new learning framework—MIML (Multiple Instances Multiple labels) that could deal with complex and ambigu-ous cases. For example, we can put an image including sky, cloud, lake, and trees into a bag (without being splitted) as an example, and based on these examples, MIML could be trained to predict whether a new instance is correlated to the labels (sky, cloud, lake, and tree). There are a number of attempts towards exploiting MIML in real applications such as text categorization and image classification [9].

Ordinal classification in multilabel learning tries to generalize each original label \( \{y_{ij} \in (-1,1)\} \) to a member of graded label series \( \{y_{ij} \in \{m_1, m_2 \cdots m_k\}, m_1 < m_2 < \cdots < m_k\} \). It does not output the defined class labels but the fuzzy label series. This case may be reasonable in practical application, because it may be hard to answer “yes” or “no” exclusively but easier to answer “somewhat” or “almost.” Give an example of emotional classification of songs; the labels associated with a song may be not “angry-aggressive” at all, somewhat “quiet-still,” almost “relaxing-clam,” fully “happy-pleased” (“angry-aggressive”<“quiet-still”<“relaxing-clam”<“happy-pleased”), while the datasets on this case is few as far as we know [10].

Multitask classification attempts to train multiple tasks in parallel by sharing representations and induces bias information on related tasks to improve the generalization performance to others. Just like the correlation between labels for MLL, the bias information for related tasks is helpful if we want to extend tabula rasa learning to complex tasks. For example, each of the following: the number of lanes on the road, the location of road edge, the location of the road center, and the intensity of road surface, can be regarded as a task in the field of road-tracking, respectively. If we take a label as a task, MLL could be considered a special case of multitask. The differences between them lie in two aspects at least: one is that all the examples in MLL need to share the feature space while those in multitask need not; another one is that the goal of MLL is to obtain a label subset while those of multitask is to improve generalization performance using bias information. Some literatures indicate that techniques for multitask classification may be used to benefit MLL [11].
3. The Correlation among Labels

3.1. Category of Label Correlations. The key challenge of MLL might be increased dramatically with the increase of the label space scale \( |Y| \). The mainstream solution is to use correlation among labels in \( Y \) to lower learning space of model. For example, “blue sky” label usually appears at the same time with the “clouds” label; News pages marked as “politics” usually do not appear “entertainment” label. Formally speaking, \( P(y_i) \neq P(y_i | y_j) \). Strategies using the correlation among labels can be roughly divided into three categories based on its order: (1) the first-order strategy is designed to consider a single label to correspond to one binary classification, respectively, ignoring the relationship with other remaining labels. The strategy is simple and efficient, while the performance may not be optimal. (b) The second-order strategy is designed to consider the pairwise relation between labels, such as related and irrelevant labels through the label ranking. The performance is better than the first-order strategy, while the correlation between labels in practical application will surpass the premise of the second order relationship. (c) High-order strategy is designed to consider higher order relation among either all or random subsets of labels, such as considering the influence of the rest of all on each label. The ability in expressing the relationship is the strongest while its computational complexity limits applications to small scale.

On the other hand, the correlation among labels can be divided into unconditional and conditional correlation in terms of its probability distribution [12]. Unconditional correlation refers to the expected value of correlation globally, i.e., the expected value of correlation among labels associated with all instances in an input space. Normality speaking, the correlation among labels refers to the expected value of correlation globally, whereas conditional correlation captures the correlation among labels given a special instance \( x \in X \).

3.2. Why Label Correlation Work. The correlation among labels plays a key role in improving MLL model training process. Considering unconditional second order as an example, let \( b_i \) and \( b_j \) denote the decision boundaries corresponding to labels \( y_i = h_i(x) \) and \( y_j = h_j(x) \) separately, and \( \alpha \) denotes the angle between \( b_i \) and \( b_j \), as shown in Figure 2 (a rectangle represents an instance associated with \( y_i \); a round represents an instance associated with \( y_j \)).

And \( y_j \) can be expressed as
\[
y_j = f(y_i) + g(x),
\]
where \( f(\cdot) \) is a dependence function between \( y_i \) and \( y_j \), the strength of which is related to \( \alpha \); i.e., with the increase of \( \alpha \), the impact strength of \( f(\cdot) \) decreases gradually. And \( g(x) \) is a minimum about \( x \), i.e., \( g(x) = 0 \) with high probability or for most instances. Since \( g(x) \) plays a less important role, it is reasonable to believe that \( y_j \) is determined by \( f(y_i) \) in formula (1). That can be illustrated in Figure 3 in which the performance of MLL on the ordinate axis is expressed by subset 0/1 loss.

3.3. How to Obtain the Correlation among Labels. The correlation among labels is so important that many researchers are interested in this topic: how to obtain it. In fact, this is challenging. To the best of our knowledge, there are four methods to solve this problem. (1) The first one is statistics based on the number of the instances assigned to each label, such as chi-square, difference in proportions, and likelihood ratio test, on the number of these instances [13]. Taking the chi-square test as an example, assume the frequency counts of instances related to label \( y_i \) and \( y_j \) cooccurrences as presented in Table 1.

According to the chi-square test,
\[
x^2 = \frac{(ad - bc)^2}{(a + b)(c + d)(b + d)(a + c)}.
\]

The greater the value of \( x^2 \), the greater the likelihood that the hypothesis of “label \( y_i \) and \( y_j \) are correlated” holds. Thus, the correlated label pairs can be sorted according to \( x^2 \) value in descending order. (2) The second one is based on the similarity of features assigned to different labels. This method assumes that if there are sufficient instances assigned to label \( y_i \) and \( y_j \) that contain some similar features it is reasonable to believe \( y_i \) and \( y_j \) are correlated. Specifically, for any instance \( x_i \) in \( D \), if it is assigned to \( y_i \) and \( y_j \) simultaneously, put it into set \( A \). Note the common features for all instances in \( A \) as a vector \( f_c = \{x_m \cdots x_n\} \); the larger the \( |A| \) and \( |n - m + 1| \), the more reason to believe that \( y_i \) and \( y_j \) are correlated. Association rules and principal component analysis may be helpful in terms of similarity in feature space. It is worth mentioning that if the dimension of the features in the training set \( D \) is too large (i.e., \( q \ll d \)), the influence of a single label will be diluted. (3) The correlation among labels could be provided to MLL classifier manually using domain knowledge. For example, a hierarchy among labels or dividing labels into several mutually exclusive subsets in a practical application may be helpful in reducing computational complexity and improving accuracy. (4) Using the
which sorting function dependent on the successor to be established. First, set a y space in an ordered sequence

\[ \pi^{(j)} \]

fi chain algorithm is to converse MLL to a chain composed of label and does not indicate the size relation between them. For a 3.4.1. Classification among labels.

deal with multiple labels problems and how to use the corre-
algorithms based on label correlation to expound how to as a single label [14]. Here, we will review some classical algorithms are based and Label Powerset method in which binary relevance algorithm on which many state-of-the-art

algorithms are based and Label Powerset method in which

Although the correlation among labels plays Correlation.

3.4. MLL Classification Algorithms Based on Label Correlation. Although the correlation among labels plays an important role to MLL, some straightforward algorithms do not take it into account when dealing with multiple labels problems, i.e., the first order strategy. For example, the binary relevance algorithm on which many state-of-the-art algorithms are based and Label Powerset method in which each label set exists in the training dataset will be considered as a single label [14]. Here, we will review some classical algorithms based on label correlation to expound how to deal with multiple labels problems and how to use the corre-
alternatives among labels.

3.4.1. Classifier Chain Algorithm. The basic idea of classifier chain algorithm is to converse MLL to a chain composed of some binary classifiers in which the precursor must be dependent on the successor to be established. First, set a sorting function \( \pi(Y) \) to arrange all the labels in the label space in an ordered sequence \( y_{\pi(1)} < y_{\pi(2)} < \cdots < y_{\pi(q)} \); in which \( y_{\pi(1)} < y_{\pi(2)} < \cdots < y_{\pi(q)} \) represents \( y_{\pi(i)} \) is prior to \( y_{\pi(j)} \) and does not indicate the size relation between them. For a label \( y_{\pi(j)} \) in an ordered sequence, a dichotomous classification training set corresponding to it is constructed as follows:

\[
\mathcal{D}_{\pi(j)} = \left\{ \left( x_i, \text{Pre}_{\pi(j)}^{(i)}, \varphi \left( y_{\pi(j)}, y_{\pi(i)} \right) \right) \mid 1 \leq i \leq m \right\}.
\]  

Table 1: Frequency counts of instances related to label.

<table>
<thead>
<tr>
<th>( y_j )</th>
<th>( \neg y_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

\[
\text{Pre}_{\pi(j)}^i \] represents the relevance between all the precursor labels of \( y_{\pi(j)} \) in an ordered sequence and \( x_i \), that is,

\[
\text{Pre}_{\pi(j)}^i = \left( \varphi \left( y_{\pi(1)}, y_{\pi(i)} \right), \cdots, \varphi \left( y_{\pi(j-1)}, y_{\pi(i)} \right) \right)^T. \tag{4}
\]

After \( \mathcal{D}_{\pi(j)} \) is constructed, you can choose a traditional supervised learning algorithm \( \mathcal{B}(\cdot) \) to train binary classifiers \( h_{\pi(j)} \leftarrow \mathcal{B}(\mathcal{D}_{\pi(j)}) \). For a testing instance \( x \), it traversal recursively the binary classifiers chain to determine whether a label \( y_{\pi(j)} \) is assigned to \( x \) or not, and the predicted label set of \( x \) can be written as

\[
Y = \left\{ y_{\pi(j)} \mid h_{\pi(j)}(x) = 1, 1 \leq j \leq q \right\}. \tag{5}
\]

The flowchart of the algorithm classifier chains is shown in Figure 4. If the output of binary classifiers \( h_{\pi(j)}(x) \) is a real value, we need to symbol for it to judge whether the class labels corresponding to the value are the related labels of \( x \) or not.

The successor binary classifier of classifier chain algorithm relies on the precursor classifier, so the sort function \( \pi(Y) \) is extremely critical. The ECC (ensemble of classifier chains) algorithm gives an efficient sorting method. Firstly, ECC constructs \( n \) sort functions \( \pi^1(Y), \pi^2(Y), \cdots, \pi^n(Y) \) on class label space, and then for any sort function \( \pi^i(Y) \), it generates a new dataset \( D^i \) by sampling from the original dataset \( D \). Finally, for any \( \pi^i(Y) \), a binary classifier chain is produced in corresponding \( D^i \) to integrate. For sampling without replacement, the size is generally 67% of the original dataset, while for sampling with replacement, it retains the same size as the original dataset. The algorithm is a high-

order strategy, which has the advantage of taking into account about all the relations among labels; the disadvan-
tage is that it is not suitable for parallel training on large-

Figure 3: The relationship between the performance of MLL and angle \( \alpha \).
scale datasets due to the successive relation in the chain. Additionally, the dimension of the data is also increasing with constantly iterating.

3.4.2. Calibrated Label Ranking. The idea of this algorithm is to convert MLL problems into label ranking problems by comparing each pair wise among labels. For the label space which scale is \( q \), we need \( (q-1)/2 \) label pairs in total to describe the one-one relationship among labels. For each label pair \((y_j, y_k)\), construct a training set corresponding to it:

\[
\mathcal{D}_{jk} = \left\{ \left( x_i, \theta \left( Y_i, y_j, y_k \right) \right) \mid \varphi \left( Y_i, y_j \right) \neq \varphi \left( Y_i, y_k \right), 1 \leq i \leq m \right\},
\]

in which

\[
\theta(Y_i, y_k) = \begin{cases} 
1, & y_j \in Y_i, y_k \notin Y_i, \\
1 - 1, & y_k \in Y_i, y_j \notin Y_i.
\end{cases}
\]

Thus, for each instance \( x_i \), it would be resampled \(|Y_i|*|\bar{Y}_i|\) times for training binary classifiers. And then, the algorithm selects a traditional binary classification algorithm \( \mathfrak{B}(\cdot) \) to train a binary classifier \( h_{jk} \rightarrow \mathfrak{B}(\mathcal{D}_{jk}) \). For instance, \( x \) the learning system will vote to \( y_j \) if \( h_{jk}(x) > 0 \); otherwise, vote to \( y_k \). For a testing instance \( x \), the algorithm calls \( q(q - 1)/2 \) classifiers to vote for each label as follows:

\[
V \left( x, y_j \right) = \sum_{k=1}^{q-1} \left[ h_{jk}(x) \leq 0 \right] + \sum_{k=j+1}^{q} \left[ h_{jk}(x) \geq 0 \right].
\]

In order to divide labels into two types of related and unrelated labels by voting, the algorithm set a virtual label \( y_v \) (do not care about its value) as a threshold and constructed \( q \) training sets corresponding to the label pair of \( y_v \) and \( y_j (j \in [1,q]) \):

\[
\mathfrak{D}_{y_j} = \left\{ \left( x_i, \varphi \left( Y_i, y_j \right) \right) \mid 1 \leq i \leq m \right\},
\]

Ibid., \( q \) binary classifiers were trained on \( \mathfrak{D}_{y_v} \). Let \( y_v \) and \( y_j \) join the vote process, if the \( h_{y_v}(x) \geq 0 \) that indicated \( y_j \) was at the front of the threshold, that is, relevant label. New voting function is updated to

\[
V^* \left( x, y_j \right) = V \left( x, y_j \right) + \left[ h_{y_v}(x) > 0 \right].
\]

The flowchart of the algorithm Calibrated Label Ranking is shown in Figure 5. The algorithm takes into account the relationship between each label pair and therefore belongs to the second-order strategy. Compared with the one-vs.-rest relationship, the one-vs.-one relationship that the algorithm considered to construct training set could alleviate the imbalance problem among labels. In addition, the

![Figure 4: The flow chart of algorithm classifier chains.](image-url)
number of constructed binary classifiers is upgraded to the secondary from the linear scale, so the algorithm could be improved by using the Pruning branch method in the testing phase [15].

3.4.3. Random k Label Sets. The idea of the method is to convert MLL into an ensemble learning problem based on multiclass. The basic classifiers in the ensemble learning randomly selected n label space subsets contained k labels by using LP (Label Powerset) technology (each subset as a class). Thus, it converted MLL to multiclass problems [16].

Assume the multiclass training set is \( D_M = \{ (x_i, \sigma(Y_j)) \} \) \( 1 \leq i \leq m \). A multiclassifier \( h(.) \leftarrow M(D_M) \) could be obtained by using a conventional multiclass learning algorithm \( M \). When the label space \( Y \) is very large, there would be too many label subsets in \( D \), which would lead to high complexity. Random \( k \) label sets specified the scale of label subsets to be \( k \), and randomly selected \( n \) subsets from \( (q/k) \) for learning. So the scale of label space in multiclass tasks is reduced from \( q \) to \( k \) \( (k < q) \). For any label subset \( Y_j^k \), construct a multiple class training set:

\[
D_M^{Y_j^k} = \{ (x_i, \sigma(Y_j \cap Y_j^k)) \} | 1 \leq i \leq m \}.
\]

Ibid., multiclassifier \( h^{Y_j^k}(.) \leftarrow M(D_M^{Y_j^k}) \) can be obtained. The algorithm randomly selected \( n \) multiclassifiers for ensemble learning. For any label \( y_r \) \( (r \in [1, q]) \) of a testing instance \( x \), the max value \( \max (y_r) \) that the ensemble learning system might vote for it and the actual value \( \text{act}(y_r) \) that the ensemble learning system actually voted for it were calculated as follows:

\[
\begin{align*}
\max (x, y_r) &= \sum_{j=1}^{q} \left[ y_r \in Y_j^k \right], \\
\text{act}(x, y_r) &= \sum_{j=1}^{n} \left[ y_r \in \sigma^{-1} \left( h_{Y_j^k}(x) \right) \right].
\end{align*}
\]

When \( \text{act}(y_r) \) reached half of \( \max (y_r) \), \( y_r \) could be considered to be related to \( x \), that is,

\[
Y_x = \{ > 0.5 \}.
\]

Random \( k \) label set is a high-order strategy. The value of \( k \) in the algorithm controls the relevant range among labels, so there is a critical impact on the performance of the algorithm; the experience value of \( k \) is generally set to 3, while the number of base classifiers \( n \) preferably chose \( 2q \).

3.4.4. GLOCAL (Multilabel Learning with GLObal and loCAL Correlation). In practical applications of MLL, more complex situations may occur. For example, partial labels
of some instances are complete, while others may be missing. Since the given data distribution may be different from the ground truth distribution, learning the correlation among labels on such datasets is challenging [17]. In addition, global and local label correlation may exist simultaneously in the same application. For the two cases discussed above, the GLOCAL algorithm transforms the ground truth matrix $\mathbf{Y}$ into a product of two matrices by the low-rank matrix decomposition, i.e., $\mathbf{Y} = \mathbf{U}\mathbf{V}^T$ [18], where the matrix $\mathbf{V}$ represents a latent label matrix that is more concise and semantically more abstract than the original label matrix $\mathbf{Y}$, while the matrix $\mathbf{U}$ is used to project $\mathbf{Y}$ into the latent label space. Let $Y$ denote the given label matrix (possibly missing some labels) in training dataset $D$. The local correlations of labels can be obtained by dividing training dataset $D$ into $g$-related subsets by domain knowledge or clustering. The GLOCAL algorithm alternately solves the missing labels, learns the linear classifier, and adopts both global and local label correlations by optimizing the following formula:

$$\min_{\mathbf{U}, \mathbf{V}, \mathbf{W}} \|\Pi_{\Omega}(\mathbf{Y} - \mathbf{U}\mathbf{V})\|_F^2 + \lambda_1 \|\mathbf{V} - \mathbf{W}^T\mathbf{X}\|_F^2 + \lambda_2 \mathbf{R}(U, V, W) + \sum_{m=1}^g \lambda_4 \text{tr}(\mathbf{F}_m^T\mathbf{L}_m\mathbf{F}_m),$$

(14)

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are tradeoff parameters; when the corresponding position of $Y$ is a non-0 element, $\Pi_{\Omega}(\mathbf{Y} - \mathbf{U}\mathbf{V}) = (\mathbf{Y} - \mathbf{U}\mathbf{V})$; otherwise, $\Pi_{\Omega}(\mathbf{Y} - \mathbf{U}\mathbf{V}) = 0$. Matrix $\mathbf{W}$ is used to implement mapping instance matrix $\mathbf{X}$ onto the latent label matrix $\mathbf{V}$, which can be achieved by minimizing $\|\mathbf{V} - \mathbf{W}^T\mathbf{X}\|_F^2$, $\mathbf{R}(U, V, W)$ is the regular term part, used to regularize the model using label correlation; that is, the more the two labels are positively correlated, the closer the output value of the model is. Let $\mathbf{F}_0 = \mathbf{UW}^T\mathbf{X}$. Suppose the global correlation matrix is marked as $\mathbf{S}_0$, $\mathbf{L}_0$ is the Laplacian matrix of $\mathbf{S}_0$. For any training subset $\mathbf{D}_m$, assuming that the local correlation matrix on it is $\mathbf{S}_m$, then $\mathbf{F}_m = \mathbf{UW}^T\mathbf{X}_m$ and $\mathbf{L}_m$ is the Laplacian matrix of $\mathbf{S}_m$. The alternating optimization of $\mathbf{U}, \mathbf{V}, \mathbf{W}$ is performed by fixing two of them and optimizing the last one.

3.5. Challenging Issues about Label Correlation. Although the correlation among labels has attracted the interest of scholars in recent years, at least the following challenging issues need to be further explored. First of all, the correlation among labels has been approved, but the study on formal characterization on the concept or principled mechanism on appropriate usage of the concept is not yet sufficient so far. Some researches indicate that the correlation among labels might be asymmetric; i.e., the influence of label $y_j$ to label $y_i$ might not be valid on label $y_i$ to label $y_j$. Furthermore, the correlation among labels might be local; i.e., there are seldom global correlations for all examples on the whole training dataset. Secondly, the applicable conditions of the correlation among labels are not given enough consideration. How to use label correlation to improve the performance of MLL algorithm in specific fields still needs a lot of further exploration in experimental comparison. Thirdly, why label correlation can improve the performance of MLL and how it works have not been fully understood. The last but not the least, for large scale label space, how to reduce the computational complexity of the algorithm when the correlation among labels is complex (especially for high order strategy model). In other words, how to trade-off between expressing ability of label correlation and computational complexity. In addition, for all MLL algorithms (not just for that based on label correlation), the nature of training dataset (e.g., the completeness of label or noise data) has a great influence on the performance of MLL algorithms; and the class imbalance in MLL is another challenging topic [19].

4. Evaluation Metric

4.1. Common Evaluation Metrics and Related Theories. Since each instance is associated with multiple labels in MLL, the metrics for performance evaluation in MLL are more complex than those in binary classification [20]. In general, the common metrics for performance evaluation in MLL are subset accuracy, hamming loss, accuracy, one-error, coverage, ranking loss, average precision, macroaveraging, microaveraging, AUC-macro, AUC-micro, and so on.

To the best of our knowledge, existing MLL evaluation metrics often only reflect the generalization performance of MLL classifiers in some aspects, and no universal MLL evaluation metric has been found. Generally speaking, adopting metrics based on example is a good choice for classification tasks, while adopting metrics based on labels is a better choice for retrieval tasks. Cheng et al. showed that the Hamming loss and the subset could not be optimized simultaneously in 2010. When evaluating the performance of the algorithm on the test set, more metrics are chosen, and the evaluation effect should be fairer.

4.2. MLL Classification Algorithms Based on Evaluation Metrics. The MLL classification task is more complex, and the MLL evaluation metric is more challenging, so more recent studies have focused on this work. Different metrics often take care of algorithm performance from different aspects and reflect different natures. However, most MLL algorithms often only optimize one metric [21]. It should be said that different metrics are adopted for different MLL algorithms, and their generalization performances are also different. Gao et al. study the Bayes coherence of substitution loss in MLL and demonstrates that neither any convex substitution loss nor rank loss are coherent, proposing an alternative loss function for Hamming loss in specific context conditions [22].

Wu et al. analyzed the common properties of 11 MLL metrics mentioned above, proposed the concepts of instance-wise and label-wise based on margin, and gave a unified understanding about MLL:

$$\text{instance}_{\text{wise}} = \min_{a,b} \left\{ f_j(x_a) - f_j(x_b) \right\} \left\{ (a \in Y, b \in Y^c) \right\},$$

(15)
where $Y_{+}$ is the set of the positive instance index of $y_j$ and $Y_{-}$ is the set of the negative instance index of $y_j$.

\[
\text{labelwise} = \min_{u,v} \{f_u(x_i) - f_v(x_i)\} (u \in Y_{+}, v \in Y_{-}),
\]

where $Y_{+}$ is the set of the relevant label index of $x_i$ and $Y_{-}$ is the set of the irrelevant label index of $x_i$.

It was shown that the concepts based on margin can be used to optimize certain MLL metrics. The metrics that can be optimized were analyzed for ranking and classification, respectively, and the details are shown in Table 2.

Because double wise has the advantages of caring 11 metrics simultaneously, an algorithm for optimizing both instance-wise and label-wise is proposed, called the LIMO algorithm. Assuming $F = W$, the LIMO algorithm randomly assigns the weight matrix $W$ an initial value of $W_0$ that follows a normal distribution $(0, d)$, and the optimization model is solved by SGD (stochastic gradient descent):

\[
\begin{align*}
\arg\min_{w_d} & \sum_{i=1}^{m} ||w_i||^2 + \lambda_1 \sum_{i=1}^{m} \xi_i \varepsilon_i^u + \lambda_2 \sum_{j=1}^{q} \sum_{a,b} \xi_{ab}^j \\
\text{s.t.} & \quad w_i^T x_i - w_j^T x_i > 1 - \xi_i \varepsilon_i^u, \xi_i \varepsilon_i^u \geq 0, \\
& \quad w_j^T x_a - w_j^T x_b > 1 - \xi_{ab}^j, \xi_{ab}^j \geq 0,
\end{align*}
\]

for $j = 1, \ldots, q$ and $a \in Y_{+}, v \in Y_{-}$

where $\xi_i$ and $\xi_{ab}^j$ are two relaxation variables and $\lambda_1$ and $\lambda_2$ are two harmonic parameters used to adjust the instance-wise and label-wise; i.e., when $\lambda_1 = 0$, the LIMO algorithm optimizes instance-wise; when $\lambda_2 = 0$, the LIMO algorithm optimizes label-wise; when $\lambda_1 > 0$ and $\lambda_2 > 0$, the LIMO algorithm optimizes both instance-wise and label-wise. However, it is difficult to optimize both instance-wise and label-wise simultaneously in practice. An important contribution of LIMO algorithm is that it provides some reference for how to tradeoff between instance-wise and label-wise when they cannot be optimized at the same time.

### 5. Multilabel Semisupervised Learning

Traditional MLL is based on two fundamental assumptions: First, the sample in the training set has complete labels; second, the training set provides sufficient examples for the training process. However, in the practical application, annotating for multiple labels cost more than that for a single label. The correlation among unlabeled instances contains a wealth of useful information, which may contribute to reducing of the cost caused by multilabel classification and improving the classification performance [23, 24].

Multilabel semisupervised learning is an effective method that can make full use of these useful information. The training sets in multilabel semisupervised learning can be described as

\[ D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m), (x_t, y_t), \ldots, (x_p, y_p), \ldots, (x_u, y_u)\}, \]

in which $D\{\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ represents data with complete labels; $D_p = \{(x_{p+1}, y_{p+1}), \ldots, (x_u, y_u)\}$ indicate data with partial labels; and $D_u = \{(x_1, y_1), (x_2, y_2), \ldots, (x_u, y_u)\}$ say the data with no label. The numbers of the elements which they contain are, respectively, $l = |D_p|, p = |D_p|, u = |D_u|$ and $l + p + u = m = |D|$. For easy description herein, $D_p$ and $D_u$ collectively refer to $D_u$.

The goal of the multilabel semisupervised learning is to model on a given training set and to predict the labels assigned to an unknown instance $x$ by learning. If $x \in D_u$, that is, the testing instance is closed for the training set, this learning strategy is called transductive semisupervised learning; otherwise, known as inductive semisupervised learning. The typical transductive semisupervised learning includes Tram (Transductive multilabel classification) and DMMS (normalized dependence maximization multilabel semisupervised learning method). Tram is built on the following assumption: the labels related to an instance are smooth for the aspect of attribute manifold, and then, a MLL model like random walk can be obtained. While DMMS is built on the basis of the dependence of the statistical theory, take the labels in $D_u$ as constraint. It estimates the normalized dependence to attribute set and label set on the entire dataset, and the optimization target is to maximum the estimated value of dependence. DMMS learning outcomes remained significant when the data $D_u$ is sparse [25, 26].

It is worth mentioning that in the learning frameworks with unlabeled information to improve learning performance, active learning is an effective way, resulting in a set of learning frameworks called MLAL (multilabel active learning). Active learning submits some unlabeled data to the field experts for annotating in iteration ways according to some kind of inquiry criterion and then submits the labeled data to the training model to improve its generalization performance. The study focuses on how to filter the
Typical instances to experts to annotate and, as much as possible, to reduce the number of labeling. Currently, active learning query criteria can be divided into two types, which are information content and representation. Information content (such as uncertainty and the degree of unity) describes the performance of the selected unlabeled instances in terms of reducing the uncertainty of learning model. Representation (such as density and cluster centers) describes the ability that the selected instances represent the overall data distribution. Huang et al. proposed AUDI (Active query driven by Uncertainty and Diversity for Incremental multilabel learning), QUIRE (Querying Informative and Representative Examples) which are a certain breakthrough on reducing the labeling cost by active learning using relationships between labels [27, 28].

AUDI constructs a classification model for any label $y_j (j \in [1, q])$ in the label space:

$$f_j(x) = \omega_j^T W_\theta x,$$

(20)

in which $W_\theta$ is the $q \times d$ matrix, with the role of mapping the feature vectors ($d$ dimension) in training set $D_t$ to a low-dimensional ($q$ dimensional) shared subspace, and $\omega_j^T$ is a $q$-dimensional weight vector, which corresponds to a linear model based on a shared subspace. For a $x_i$ in $D_t$ and a label $y_j (y_j \in Y_j)$ assigned to $x_i$, define an error of ranking:

$$e(x_i, y_k) = \frac{R(x_i, y_k)}{q},$$

(21)

in which $R(x_i, y_k)$ indicates the number of the misarranged labels with $y_k$ as a center, that is, the number of the unrelated labels before $y_k$. The purpose of learning is to minimize $e(x_i, y_k)$ on the entire training set. However, this objective function is a nonconvex function. In order to facilitate optimization, AUDI selected the hamming loss as an approximate alternative loss function:

$$e(x_i, y_k) \leq \sum_{y \in Y} e(x_i, y_k) \max \left(0, 1 + f_j(x_i) - f_j(x_i)\right).$$

(22)

In fact, the wrong order on $x_i$ may also be considered as the sum of the irrelevant labels before the relevant label $y_k$. So it can be decomposed into each $y \in Y$ to get

$$L(x_i, y_k, \tilde{y}_w) = e(x_i, y_k) \max \left(0, 1 + f_j(x_i) - f_j(x_i)\right).$$

(23)

Using the stochastic gradient descent method to solve the objective function, we can obtain classification model.

After the model being obtained, it predicted for all instances in $D_u$, and AUDI wants to select the most valuable instances from them to give the experts in the fields to annotate. The most valuable points are reflected in two aspects: First, the instance contains most information which is used to distinguish uncertainty whether or not being related; second, the diversity is the biggest between the selected instance and the previously queried instances. It is the typical representative of the unselected instances. That is,

$$C_1(x_i) = \frac{\|\sum_{k=1}^{q} I[y_{ik} > 0] - (1/l) \sum_{j=1}^{l} \sum_{k=1}^{q} I[y_{ik} > 0]\|}{\max (0.5, q - |U(x_i)|)}.$$

(24)

in which $\|\sum_{k=1}^{q} I[y_{ik} > 0] - (1/l) \sum_{j=1}^{l} \sum_{k=1}^{q} I[y_{ik} > 0]\|$ describes the disunity of label set cardinality (LCI) and represents the difference between the number of related labels that the model output and the average number of ground truth-related labels in $D_t$. $|U(x_i)|$ represents a indicator function. $U(x_i)$ represents the label set with no query assigned to $x_i$.

Suppose the most valuable instances selected by (24) are $x^*$, and AUDI queried the most valuable label once for every time to the labels of $x^*$; AUDI screens the most valuable label of $x^*$ using the following formula:

$$C_2(x_i) = \left| f_j(x^*) - f_j(x^*) \right|.$$

(25)

Here, $y^*$ stands for the virtual label between relevant and irrelevant label. $y^*$ which makes $C_2(x_i)$ minimum is the most valuable label.

AUDI obtains a predicting model with the labeled training set firstly, and each round filters a most valuable instance-label pair based on the predicated value of the model to the experts of field to annotate. The model filters out redundant labels in the maximum extent using the relationship between the labels, and therefore, it belongs to using the relationship between the labels indirectly. When screening instance-label pair, it takes overall consideration of the two factors: amount of information and representation, which reduces the query cost substantially and enhances the classification performance. But the amount of information and representation is connected with each other in a fixed way, and this way may be less flexible in certain applications.

QUIRE trains a classifier $f_{j, i}$, $j \in [1, q]$ for each label on the labeled dataset $D_l$, and the predicted value that $f_j$ output for each instance consists of a predicted vector $f_{j, i}$ on the label, and predicted vectors for all labels form a predicted matrix $F = \{f_{j, i} | 1 \leq j \leq q\}$. Calculating nuclear matrix $K$ the entire dataset, entering the relation matrix between labels $R$, and objective function could be defined as

$$\min_{F \in \mathbb{R}^{q \times q}} \text{tr}(R^{-1} F K^{-1} F) + |F - Y|_2^2,$$

(26)

in which $\text{tr}()$ is the matrix trace function and $Y$ is a ground-truth label matrix of all examples. Based on min–max view framework, the optimal solution is obtained. On this basis, QUIRE defines a matrix $L$ based on $R, K$, which is used to select guidelines which contributes to selecting the most
valuable example-label pair:

\[ L = \lambda [R \otimes K + \lambda I]^{-1}. \]  

\( \otimes \) is the tensor product of the matrix.

Because \( L \) is defined on \( D_t \) and \( D_u \) (the entire training set), the predicted confidence for the selected instances to be queried using \( D_t \) cloud reflect the factor of amount of information; for \( D_u \), if the selected instance is very representative, it should be highly similar to other instances, and if labels conform to uniform distribution, lots of unlabeled instances in \( D_u \) will result in a lower degree of confidence. And therefore QUIRE can choose the most valuable instance-label pair by minimizing both the confidences. The matrix \( L \) constructed on \( R \) and \( K \) contains two relationships: the relationship between labels and the relationship between instances, and the two factors, the amount of information and co-ordinate representation, are overall planed.

The QUIRE method makes use of the label relations directly in the algorithm, and reduces the number of computing inverse matrix when solving unlabeled instance-label and is suitable for processing massive tasks with more unlabeled instances. However, the QUIRE method requires the relationship between the labels in \( D_t \) as the known knowledge to input to algorithm, and regards the labels in \( D_u \) lines with uniform distribution.

Relations between the labels play a key role in multilabel active learning tasks. When submitting a query to experts in the field, the type of query and the exploit of the relationship between the labels in \( D_t \) are equally important. Huang et al. firstly proposed a new multilabel active learning framework AURO to query correlation order between the label pair, which requires experts in the field to mark the importance of the correlation between labels [29]. Different with the traditional multilabel active learning, AURO submits the query information to unlabeled instances to the staff in the form alternative versions. Suppose \( y_i, y_j \) correspond to the two labels to be queried of the instance \( x_o \), and design the options as follows: (1) for instance \( x_o, y_i \) is more relevant than \( y_j \); (2) for instance \( x_o, y_i \) is not more relevant than \( y_j \); and (3) for instance \( x_o, y_i \) and \( y_j \) are not relevant. With the simple selection strategy and sequence label model, the framework helps annotator with less professional knowledge to obtain more label information. The framework opens up a new perspective of multilabel active learning, which gives the researchers a good way to design and explore more extensive query options to exploit the relationship between the labels, or in conjunction with a better selection strategy to achieve better classification performance. Compared with the traditional multilabel active learning algorithm, the time cost on label annotation is reasonable, but the original frame on option design is simpler; for example, the similar relevant cases of the two labels are not considered.

The notable feature of the multilabel active learning requires human-computer interaction. In multilabel semisupervised learning methods which do not require human-computer interaction, the inductive semisupervised learning method has better generalization performance. Compared with a transductive method, it can naturally predict unknown instances outside the testing set. Currently, the amount of researches on inductive semisupervised learning is relatively small. Taking traditional empirical risk minimization principle as theoretical guarantee, Li et al. first proposed inductive semisupervised learning method MASS (multilabel semisupervised learning) aimed at MLL. Similar with a traditional multilabel learning algorithm, MASS assumes that decision function is resolved to each label \( F = \{f_1, f_2, \cdots, f_q\} \) and assumes that the decision function \( f \) on each label is a linear model, namely, \( f_j(x) = <\omega_j, \phi(x)> \), in which \( \phi(x) \) stands for the attribute mapping function derived by the kernel function and \( <\cdot, \cdot> \) stands for the inner product in reproducing kernel Hilbert space \( H \) spanned by the kernel function. It is need to guarantee theoretically the vector \( F \) combined of \( f \) belongs to the reproducing kernel Hilbert space which is signed as \( H_F \). The objective function is solved by MASS:

\[
F^* = \arg \min_{F \in H_F} \frac{1}{l} \sum_{i=1}^{l} \sum_{j=1}^{q} V(x_i, y_{ij}, F) + \frac{r_q}{q} \Omega(F) + \frac{r_a}{(l+m)^2} I(F). 
\]

\( I \) is the labeled data scale; \( m \) is the total size of the training set; \( q \) is the size of labeled space; \( V(x_i, y_{ij}, F) \) represents empirical risk on datasets. Similar with classic MLL algorithms, MASS chosen hamming loss function to measure it, namely,

\[
V(x_i, y_{ij}, F) = \sum_{j=1}^{q} I(x_i, y_{ij}, f_j). 
\]

\( I(\cdot) \) was the hinge loss function in SVM; \( \Omega(F) \) was a regular item used to control the complexity of \( F \); \( I(F) \) was a manifold regular item used to constrain that “similar instances have similar multilabels with structure”; \( r_q \) and \( r_a \) were two coefficients used to make a trade-off between the importance of the two regular items, respectively. MASS defined the regular item \( \Omega(F) \) as follows:

\[
\Omega(F) = \mu ||\omega_0||^2 + \sum_{j=1}^{q} ||\omega_j||^2 \frac{q}{q} - \mu \omega_0 \omega_0^T, 
\]

in which \( \omega_0 = (1/q) \sum_{j=1}^{q} \omega_j \), \( \mu \) was a regularization coefficient in \([-1,0] \) used to adjust the force of the before and after items. \( ||\omega_0||^2 \) reflected the common property, while \( \sum_{j=1}^{q} ||\omega_j||^2 \) reflected the unique properties of the \( j \)th label. When \( \mu = -1 \) minimizing the formula (30) would drive \( \omega_0 \) of all the labels to be consistent. Instead, when \( \mu = 0 \), minimizing the formula (30) would drive \( \omega_j \) of all the labels to be independent.
MASS defined the regular item $I(F)$ as follows:

$$I(F) = \sum_{i=1}^{m} (F(x_i) - F(x_j))^T B^{-1} (F(x_i) - F(x_j)) W_{ij}$$  \hspace{1cm} (31)$$

in which $B = P^T \cdot P$, $P$ was a $q \times q$ projection matrix which projected label space into a new space that could describe the difference among multi labels. And $W$ was a $m \times m$ similarity matrix which described the similarity among the instances. As for the matrix $B$ used to describe the relation among labels, MASS gave two solving ways: one was given by the domain knowledge; the other was learned from $D_l$ and $D_u$ by alternating optimization techniques.

Wu et al. used the relation among the labels in $D_l$ and the accuracy degree inferring unlabeled instances in $D_u$ to model the learning progress and provided a novel inductive multilabel semisupervised learning algorithm iMLCU (inductive Multilabel Classification with Unlabeled data) [30]. iMLCU assumed multilabel classification model was combined by $q$ independent classifiers; each label $y_j (j \in [1, q])$ in the label space corresponded to a linear classifier $f_j(x, y_j) = \omega_j^T x + b_j$, in which $\omega_j$ was a $d$ dimensional real valued weight vector, $b_j$ was a real valued bias, and marked $W = \{(\omega_j, b_j) | 1 \leq j \leq q\}$. For an unknown instance $x$, the classifier predicted its label was

$$\hat{y} = \text{sign} \left( f_{i_1}(x, y_{i_1}), \ldots, f_{i_l}(x, y_{i_l}) \right) = \text{sign} \left( \omega_{i_1}^T x + b_{i_1}, \ldots, \omega_{i_l}^T x + b_{i_l} \right).$$  \hspace{1cm} (32)$$

And then we utilized the relation between the data in $D_l$ and $D_u$ to obtain an optimal $W$ that made the formula (32) to predict a more accurate value for the new unknown instance. Firstly, let us consider the labeled data, for any example $(x_i, Y_i)$ in $D_l$, $y_j \in Y_i$ was a related label assigned to $x_i$, and $y_k \in \bar{Y}_i$ was an irrelevant label assigned to $x_i$. Considering the relevance ranking among the labels, the decision boundary between related and irrelevant label of $x_i$ could be defined by the hyperplane between $y_j$ and $y_{j''}$, that was $(\omega_j - \omega_{j''})^T x_i + b_j - b_{j''}$. Taking maximum margin metric as the theoretical guarantee in the labeled dataset $D_l$, the objective function can be obtained:

$$\min_{\omega, b} \sum_{j=1}^{q} ||\omega_j||^2 + C \sum_{l=1}^{L} \frac{1}{|Y_l|} \sum_{y_i \in Y_l, y_j \in \bar{Y}_l} \xi_{ijk}$$  \hspace{1cm} (33)$$

s.t. \quad (\omega_j - \omega_{j''})^T x_i + b_j - b_{j''} \geq 1 - \xi_{ijk},$$  \hspace{1cm} (34)$$

$$\xi_{ijk} \geq 0 \quad i \in [1, l], \quad (y_j, y_k) \in Y_i \times \bar{Y}_i,$$  \hspace{1cm} (35)$$

in which $\Xi = \{\xi_{ijk} | (i \in [1, l], (y_j, y_k) \in Y_i \times \bar{Y}_i\}$. The two items in formula (35) were used to control model complexity and empirical loss in $D_l$ respectively, and the constant term $C$ was used to make a trade-off between them.

And then think about unlabeled data. For the unlabeled data in $D_u$, we still expect the predicted labels by using classification model can be farther away from the boundary and the misclassified instances should be punished. The difficulty for achieving this expectation lies in how to know the ground-truth labels of the data in learning process. Therefore, we cannot directly give the predicted labels some reward or punishment. For this goal, S3VM provides an idea for reference: borrowing hinge loss function.

For any unlabeled instance, the predicted labels through the previous way just only were considered putative labels but not official predicted labels. In addition, we use hinge loss function to punish the loss for putative labels.

$$c_i(x, f_i(x, y_i)) = \max \left(1 - \text{sign}(\omega_i^T x + b_i) \omega_i^T x + b_i, 0\right) = \max \left(1 - |\omega_i^T x + b_i|, 0\right).$$  \hspace{1cm} (36)$$

And then minimized penalty function in formula (36) on all labels of all the instances in $D_u$, combing with formula (35), the optimization function can be obtained:

$$\min_{\omega, b} \sum_{j=1}^{q} ||\omega_j||^2 + C \sum_{l=1}^{L} \frac{1}{|Y_l|} \sum_{y_i \in Y_l, y_j \in \bar{Y}_l} \xi_{ijk} + C_l \sum_{l=1}^{L} \sum_{y_i \in Y_l} \max(1 - |\omega_i^T x + b_i|, 0)$$

s.t. \quad (\omega_j - \omega_{j''})^T x_i + b_j - b_{j''} \geq 1 - \xi_{ijk},$$  \hspace{1cm} (37)$$

$$\xi_{ijk} \geq 0 \quad i \in [1, l], \quad (y_j, y_k) \in Y_i \times \bar{Y}_i,$$  \hspace{1cm} (38)$$

$$\frac{1}{l} \sum_{i=1}^{l} \omega_i^T x_i + b_j = \frac{1}{l} \sum_{i=1}^{l} y_{ij}.$$  \hspace{1cm} (39)$$

Comparing formula (40) with formula (35), there is an additional regular item in formula (40), that is, the total loss function used to constrain all the labels of all instances in $D_u$. In addition, compared with formula (35), the formula (40) also contains another constraint, that is, $(1/l) \sum_{i=1}^{l} (\omega_i^T x_i + b_j) = (1/l) \sum_{i=1}^{l} y_{ij}$ which is used to ensure that the predicted label diversity in $D_u$ is consistent with those in $D_l$.

iMLCU takes two parameters $C_1$ and $C_2$ to weigh the loss over $D_u$ and $D_l$. It will encounter a problem for direct optimize the formula (34): it is nonconvex function. Hence, iMLCU uses ConCave Convex Procedure (CCCP) to solve the globally optimal solution of the nonconvex function.

iMLCU tries to maximize the metric between relevant label and irrelevant label, so it belongs to the second order learning strategies.

For multilabel semisupervised learning, some researchers proposed S3VM (Semisupervised Support Vector Machine) and graph-based semisupervised learning methods. S3VM was based on the basic assumption of low-density separation; try to learn a classification hyperplane which can separate two kinds of examples and be able to pass through the low-density regions [31]. There are a series of S3VMs achievements on reducing the computational complexity, but many of these approaches do not take
full advantage of the correlation among labels. TSVM (Transductive Support Vector Machine) is the most famous one among them. For the problem that there are some harmful unlabeled instances in S3VMs training set which would reduce instead of improve the performance of the classifier, Li et al. proposed a more secure S4VMs that used a plurality of low-density slicer to approximate ground-truth boundary. It maximized the performance of the algorithm in the protection of algorithm safety. Semisupervised learning algorithms based on graph once attracted the interest of many researchers. The basic idea of them is to construct a graph by taking a labeled data in $D_l$ and an unlabeled data in $D_u$ as a node in the graph; then estimate a function on the basis of this. This function must meet two conditions: First, the dataset has been labeled with the existing labels coincide; the second is that it should be smooth enough in the entire graph. The difference between the various algorithms based on graph is the defining on the estimated function. Graph-based concept is very clear, but its large storage is overhead and difficult to use on large-scale data [32, 33].

### 6. Label Distribution Learning

Traditional multilabel learning assumes that the importance of all the related (or irrelated) labels to the instance is similar; however, label distribution learning is to dispose of the learning task in which the importance of the labels to the instance is different from each other. For example, facial expressions might contain a lot of basic expressions such as “joy” and “emotional,” but the description degree of each basic expression is very different. These learning tasks could not be finished based on the traditional MLL framework [34].

LDL defines $d^*_y \in [0, 1]$ that represents the description degree of the label $y$ to instance $x$ and assumes that the description degree of all the labels is complete, that is, $\sum_y d^*_y = 1$. The description degree of all the labels assigned to $x$ is composed of a data formation in the form of probability distributions, so it is called label distribution; the learning process by the label distribution on the given dataset is called label distribution learning. In many applications, the description degree or correlation of the labels assigned to an instance $x$ might not be same; however, the traditional MLL could only to express related relation (marked 1) or irrelated relation (marked 0); therefore, LDL is superior to MLL in general in many practical applications.

Compared with MLL, each instance in LDL datasets is assigned to one distribution about a label, instead of one label (or one label set). And the label distribution comes from nature feature of initial data in the application. Many MLL algorithms divide labels into two parts: related and irrelated labels, by label ranking based on certain threshold value. This method is only concerned with label sequence that can be distinguished but does not care about value as long as it is a specific predicted value. Instead, LDL cares for the overall distribution, each predicted value to the description degree is very important. Due to the different learning process, LDL performance assessment criteria are different from MLL. LDL measures performance of its algorithms by calculating the similarity between predicted distribution and ground-truth distribution on labels [34].

Suppose $D = \{(x_1, d^*_1), (x_2, d^*_2), \ldots, (x_m, d^*_m)\}$ represents the training set, in which $d^*_y = \{d^*_y \mid 1 \leq j \leq q\}$ represents the distribution of labels assigned to $x_i$, the goal of LDL is to learn (or train) a conditional probability mass function $p(y \mid x; \theta)$ in which $\theta$ is a parameter, so that $p(y \mid x; \theta)$ is as similar as possible to the ground-truth distribution of labels $d^*_y$. LDL using KL divergence as a similarity criterion between predicted distribution and ground-truth distribution on labels, the goal of LDL can be transformed into the following optimization problem:

$$\theta^* = \arg \max_\theta \sum_i \sum_j \left( d^*_j \ln \frac{d^*_j}{p(y_j \mid x_i; \theta)} \right)$$

(41)

A variety of learning algorithms can solve the above optimization problem. For example, $(x_i, d^*_y)$ can be split into $q$ single-labels $(x_i, y_j)(i \in [1, m], j \in [1, q])$ whose weight is $D^*_{y_j}$, so PT-Bayes used posterior probability which is calculated based on Bayes theorem on the resampled standard single-label datasets as a predicted distribution of an unknown instance; PT-SVM used SVM to solve a single label learning task. It calculated output probability of SVM on each label, through Platt posterior probability [35–36]. The framework of these two algorithms is shown in Figure 6.

In addition to resampling on datasets, directly improved traditional algorithms can also handle the above optimization problem, typical algorithms such as AA-kNN and AA-BP. AA-kNN was a lazy learning algorithm, it did not associate a graph by taking a labeled data in $D_l$ and an unlabeled data in $D_u$ as a node in the graph; then estimate a function on the basis of this. This function must meet two conditions: First, the dataset has been labeled with the existing labels coincide; the second is that it should be smooth enough in the entire graph. The difference between the various algorithms based on graph is the defining on the estimated function. Graph-based concept is very clear, but its large storage is overhead and difficult to use on large-scale data [32, 33].
ISS-LLD supposed \( p(y|X; \theta) \) model was the maximum entropy model \( (y | x; \theta) = (1/Z) \exp \left( \sum_k \beta_k g_k(x) \right) \), in which \( Z \) was a normalization factor, \( \beta_k \) was a element of vector \( \theta \), and \( g_k(x) \) was the \( k^{th} \) feature of \( x \). Thereby, the optimization problem in formula (41) was transformed to

\[
T(\theta) = \sum_{i,j} d_i^j \sum_k g_k(x_i) - \sum_i \ln \sum_j \exp \left( \sum_k \beta_k g_k(x_i) \right). \tag{43}
\]

The optimization process in formula (43) used the thought (or idea, formulation) of IIS (Improved Iterative Scanning) that was \( \theta \) which would be updated to \( +\Delta \), \( T(\theta) \) would be updated to \( T(\theta + \Delta) \) in each round; until \( T(\theta + \Delta) - T(\theta) \) would be stabilized in a smaller range.

A literature confirmed that the optimize process of IIS-LLD algorithm was ineffective. SA-BFGS improved the aforementioned optimization process by BFGS Quasi-Newton method [38]. SA-BFGS expanded formula (43) into second-order Taylor expansion at \( \theta^{(l)} \) of \( l^{th} \) round, obtaining

\[
T' \left( \theta^{(l+1)} \right) \approx T' \left( \theta^{(l)} \right) + \nabla T' \left( \theta^{(l)} \right) T + \frac{1}{2} \Delta T H(\theta^{(l)}) \Delta,
\]

in which \( \Delta = \theta^{(l+1)} - \theta^{(l)} \), \( \nabla T' \left( \theta^{(l)} \right) \) and \( H(\theta^{(l)}) \) were gradient and Hansen matrix of \( T'(\theta) \) at \( \theta^{(l)} \), respectively.

Therefore, the minimum value of formula (44) can be obtained as follows:

\[
\Delta^{(l)} = -H^{-1} \left( \theta^{(l)} \right) \nabla T' \left( \theta^{(l)} \right).
\tag{45}
\]

SA-BFGS took \( \Delta^{(l)} \) as a linear search direction by Newton method:

\[
\theta^{(l+1)} = \theta^{(l)} + \lambda \Delta^{(l)},
\tag{46}
\]

in which \( \lambda \) was step length. Due to the large computational cost when solving inverse Hansen matrix, SA-BFGS selected its approximate matrix \( B \) as a substitution. \( B \) was initialized to a random value, with each iteration as follows:

\[
B^{(l+1)} = \left( I - \rho^{(l)} \right) \left( \theta^{(l)} \right)^T B^{(l)} \left( \theta^{(l)} \right) \left( \theta^{(l)} \right)^T + \rho^{(l)} \left( \xi^{(l)} \right)^T \left( \xi^{(l)} \right). \tag{47}
\]

in which \( \nabla T' \left( \theta^{(l+1)} \right) \) could be obtained by solving the partial derivatives of \( \theta_{j,k} \) in formula (43).

The parameter vector \( \theta^{(l+1)} \) in next round can be obtained by bringing \( \nabla T' \left( \theta^{(l+1)} \right) \) and \( B^{(l+1)} \) into formula (45) and (46). Repeated the above-described process until \( \nabla T' \left( \theta^{(l)} \right) \) was convergent.

There are some other novel LDL algorithms in real application recently. COS-LDL uses a distance mapping function during training to exploit the correlation between labels and then yields the objective function. Similarly, LDL-SCL is another LDL algorithm that utilizes label correlations locally. LDL-SCL designs a distance-mapping function to map the label correlation to a distance, and then measures the similarity between labels using the distance [39]. LDL-LRR adapts the ranking loss function applying the cross-entropy and uses two metrics—Spearman’s rank and Kendall tau correlation coefficient as the similarity metrics, to obtain the objective function. The improved LDL-LRR algorithm reflects the correlation distribution of either label \( y_i \) assigned to instance \( x \); meanwhile, it reflects the relevance intensities of each related label-pairs \( (y_i, y_j) \) and enriches the expression abilities of the LDL model. ENN-LDL calculates the dominant labels in the training set \( D \) through hyperparameters \( \delta \) and then divides the training set into \( k \) training subsets according to the dominant labels [40]. A weak classifier is trained on each training subsets based on the neural network rather than the maximum information entropy. Finally, these weak classifiers are combined with weight to become an integrated classifier using ensemble learning. ENN-LDL uses local correlation between labels in the proceeding of dividing the training set with the dominant labels. On the other hand, the division changes the original distribution of the training set, so the weak classifiers trained on the subset will be bias for unknown test examples. LDLFS consists of a set of differentiable decision trees, each one consisting of a series of split nodes and leaf nodes. The split nodes are used to distinguish whether current instance should be divided into left or right subtree,
and the leaf nodes are used to represent labels distribution of current instance. Each decision tree is obtained by defining a distribution-based loss function, which uses the Kullback-Leibler divergence (K-L) between ground-truth label distribution and the label distribution predicted by decision trees. The mean of loss function in all decision trees is taken as the loss function value of forest, and the split nodes from different decision trees are allowed to be connected to the same output node in a fully connected form, so that the parameters in the split nodes can be obtained by joint-learning in the form of backpropagation. In the optimization phase, minimizing the loss function of the forest is converted to iteratively reduce its upper bound. It is worth mentioning that although LDLFS is built based on traditional machine learning models, it can still be used in deep learning models with its fully-connected architecture [41].

Since the type of output in LDL is different from those of SLL and MLL, the evaluation criteria between them is also different correspondingly. The evaluation criteria of LDL algorithms are mainly based on distance or similarity. Currently, the mainstream method is to calculate the mean of distance or similarity between ground-truth distribution and predicted distribution. Thus, the criteria of distance or similarity in probability distribution could be introduced to LDL, such as Chebyshev distance, Clark distance, KL divergence, and cosine coefficients. 41 kinds of distance (or similarity) criterion are compared by computing in 30 independent experiments.

LDL provides a universal learning framework for classification, which could meet more demand in practical applications. For example, when some labeling experts have different views on the same instance, LDL could easily solve these problems by setting different description degree on the label assigned to the instance. In addition, LDL could generate a label distribution by utilizing the correlation among labels. Consequently, LDL has opened up a new way to enrich the MLL learning theory.

7. Conclusions and Discussion

Multilabel learning has attracted great interests among researchers in the fields of machine learning and data mining because it can describe richer class information in the output space and brings new challenges as the dimension of the label space increases. In recent years, many outstanding achievements have been made in supporting theories and improving algorithms in top international conferences and journals. Many mature algorithms or learning frameworks are applied to practical applications. This paper briefly discusses four major research hotspots in MLL: correlation among labels, evaluation metric, multilabel semisupervised learning, and label distribution learning (LDL) from a theoretical and algorithmic perspective, which might be useful in conducting further research by scholars interested in MLL.

The contribution of this paper is mainly reflected in four aspects. Firstly, although the current working principle of label association and how to find it in practical applications still lacks sufficient theoretical support, many scholars have done a lot of work on it. Based on this work, this paper discusses the categories of label correlations, analyzes why label correlations work, gives four methods on how to obtain the interlabel correlations, reviews some classic MLL algorithms, and points out several challenging issues about label correlations. Secondly, the evaluation metrics of MLL is more complex than that of single-label classification. Different MLL algorithms using different metrics have different generalization performance. This paper analyzes theories and representative algorithms related to MLL evaluation metrics. Thirdly, in practical MLL applications, there are many unlabeled instances due to their labeling cost. Multilabel semisupervised learning provides a solution for such problems. A summary of its advanced algorithms is given according to transductive and inductive categories, respectively. Fourthly, the LDL framework can describe the importance of each label to an instance in output space. The paper discusses the relationship between LDL and traditional MLL framework as well as formulaic paradigm of LDL.

Due to space constraints, this paper does not cover all areas of research in multilabel learning (which is not possible). For example, semisupervised learning is not fully discussed, and many representative algorithms are not discussed. It is empirical that the problem of missing labels or incorrect labels in MLL can greatly affect its classification performance and such related issues can be more sufficiently discussed in the future. In contrast to binary classification, the imbalance in MLL is reflected in at least two aspects: (1) the imbalance in the number of instances contained within different classes and (2) the imbalance between class labels. The class imbalance problem in MLL is more difficult to address, especially for extreme imbalance. Therefore, we hope that more research work will emerge in this area in the future. In addition, combining the MLL framework with other techniques may achieve better results, such as using deep learning techniques to solve some MLL problems, or using MLL methods in knowledge graph.

Although the algorithms covered in this article all attempt to design the original algorithms as general-purpose algorithms that can be broadly applied, each tends to be biased towards the data according to the No Free Lunch (NFL) principle. Where do these MLL algorithms, including representative algorithms mentioned above, play the best performance? There are still many experiments to be done in a future work, which is also a work worthy of further exploration.

Data Availability

The datasets used during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declare that they have no conflict of interest.

Acknowledgments

This work was supported by Shandong Provincial Key Research and Development Project, China.
References


[40] Y. Zhai, Research on label distribution learning combining dominant label information, Tianjin University, 2019.