A Review on Multi-Label Learning Algorithms

Min-Ling Zhang, *Member, IEEE* and Zhi-Hua Zhou, *Fellow, IEEE*

Abstract—Multi-label learning studies the problem where each example is represented by a *single* instance while associated with *a set of* labels simultaneously. During the past decade, significant amount of progresses have been made toward this emerging machine learning paradigm. This paper aims to provide a timely review on this area with emphasis on state-of-the-art multi-label learning algorithms. Firstly, fundamentals on multi-label learning including formal definition and evaluation metrics are given. Secondly and primarily, eight representative multi-label learning algorithms are scrutinized under common notations with relevant analyses and discussions. Thirdly, several related learning settings are briefly summarized. As a conclusion, online resources and open research problems on multi-label learning are outlined for reference purposes.

Index Terms—Multi-label learning, label correlations, problem transformation, algorithm adaptation

1 Introduction

Traditional supervised learning is one of the mostly-studied machine learning paradigms, where each real-world object (example) is represented by a single instance (feature vector) and associated with a single label. Formally, let $\mathcal{X}$ denote the instance space and $\mathcal{Y}$ denote the label space, the task of traditional supervised learning is to learn a function $f:\mathcal{X} \to \mathcal{Y}$ from the training set $\{(\boldsymbol{x}_i, y_i) \mid 1 \leq i \leq m\}$. Here, $\boldsymbol{x}_i \in \mathcal{X}$ is an instance characterizing the properties (features) of an object and $y_i \in \mathcal{Y}$ is the corresponding label characterizing its semantics. Therefore, one fundamental assumption adopted by traditional supervised learning is that each example belongs to only one concept, i.e. having *unique* semantic meaning.

Although traditional supervised learning is prevailing and successful, there are many learning tasks where the above simplifying assumption does not fit well, as real-world objects might be complicated and have multiple semantic meanings simultaneously. To name a few, in text categorization, a news document could cover several topics such as *sports*, *London Olympics*, *ticket sales* and *torch relay*; In music information retrieval, a piece of symphony could convey various messages such as *piano*, *classical music*, *Mozart* and *Austria*; In automatic video annotation, one video clip could be related to some scenarios, such as *urban* and *building*, and so on.

To account for the multiple semantic meanings that one real-world object might have, one direct solution is to assign *a set of* proper labels to the object to explicitly express its semantics. Following the above consideration, the paradigm of multi-label learning naturally emerges [95]. In contrast to traditional supervised learning, in multi-label learning each object is also represented by a single instance while associated with a set of labels instead of a single label. The task is to learn a function which can predict the proper label sets for unseen instances.[1]

Early researches on multi-label learning mainly focus on the problem of multi-label text categorization [63], [75], [97]. During the past decade, multi-label learning has gradually attracted significant attentions from machine learning and related communities and has been widely applied to diverse problems from automatic annotation for multimedia contents [5], [67], [74], [85], [102] to bioinformatics [16], [27], [107], web mining [51], [82], rule mining [84], [99], information retrieval [35], [114], tag recommendation [50], [77], etc. Specifically, in recent six years (2007-2012), there are more than 60 papers with keyword *multi-label* (or *multilabel*) in the title appearing in major machine learning-related conferences (including ICML/ECML PKDD/IJCAI/AAAI/KDD/ICDM/NIPS).

This paper serves as a timely review on the emerging area of multi-label learning, where its state-of-the-art is presented in three parts.[2] In the first part (Section 2), fundamentals on multi-label learning including formal definition (learning framework, key challenge, threshold calibration) and evaluation metrics (example-based, label-based, theoretical results) are given. In the second and primary part

- *M.-L. Zhang is with the School of Computer Science and Engineering, Southeast University, Nanjing 210096, China, and the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China. E-mail: zhangml@seu.edu.cn.*
- *Z.-H. Zhou is with the National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China. E-mail: zhouzh@lamda.nju.edu.cn.*

Manuscript received 7 June 2012; revised 31 Jan. 2013; accepted 22 Feb. 2013. Date of publication 6 Mar. 2013; date of current version 10 July 2014. Recommended for acceptance by J. Pei.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier 10.1109/TKDE.2013.39

1. In a broad sense, multi-label learning can be regarded as one possible instantiation of multi-target learning [95], where each object is associated with multiple target variables (multi-dimensional outputs) [3]. Different types of target variables would give rise to different instantiations of multi-target learning, such as multi-label learning (binary targets), multi-dimensional classification (categorical/multi-class targets), multi-output/multivariate regression (numerical targets), and even learning with combined types of target variables.

2. Note that there have been some nice reviews on multi-label learning techniques [17], [89], [91]. Compared to earlier attempts in this regard, we strive to provide an enriched version with the following enhancements: a) In-depth descriptions on more algorithms; b) Comprehensive introductions on latest progresses; c) Succinct summarizations on related learning settings.

(Section 3), technical details of up to eight representative multi-label algorithms are scrutinized under common notations with necessary analyses and discussions. In the third part (Section 4), several related learning settings are briefly summarized. To conclude this review (Section 5), online resources and possible lines of future researches on multi-label learning are discussed.

2 The Paradigm

2.1 Formal Definition

2.1.1 Learning Framework

Suppose $\mathcal{X} = \mathbb{R}^d$ (or $\mathbb{Z}^d$) denotes the d-dimensional instance space, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_q\}$ denotes the label space with q possible class labels. The task of multi-label learning is to learn a function $h{:}\mathcal{X} \to 2^{\mathcal{Y}}$ from the multi-label training set $\mathcal{D} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \le i \le m\}$. For each multi-label example $(\boldsymbol{x}_i, Y_i)$, $\boldsymbol{x}_i \in \mathcal{X}$ is a d-dimensional feature vector $(x_{i1}, x_{i2}, \ldots, x_{id})^\top$ and $Y_i \subseteq \mathcal{Y}$ is the set of labels associated with $\boldsymbol{x}_i$.[3] For any unseen instance $\boldsymbol{x} \in \mathcal{X}$, the multi-label classifier $h(\cdot)$ predicts $h(\boldsymbol{x}) \subseteq \mathcal{Y}$ as the set of proper labels for $\boldsymbol{x}$.

To characterize the properties of any multi-label data set, several useful multi-label indicators can be utilized [72], [95]. The most natural way to measure the degree of multi-labeledness is *label cardinality*: $LCard(\mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m}|Y_i|$, i.e. the average number of labels per example; Accordingly, *label density* normalizes label cardinality by the number of possible labels in the label space: $LDen(\mathcal{D}) = \frac{1}{|\mathcal{Y}|} \cdot LCard(\mathcal{D})$. Another popular multi-labeledness measure is *label diversity*: $LDiv(\mathcal{D}) = |\{Y \mid \exists \boldsymbol{x}{:}(\boldsymbol{x}, Y) \in \mathcal{D}\}|$, i.e. the number of distinct label sets appeared in the data set; Similarly, label diversity can be normalized by the number of examples to indicate the proportion of distinct label sets: $PLDiv(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \cdot LDiv(\mathcal{D})$.

In most cases, the model returned by a multi-label learning system corresponds to a real-valued function $f{:}\mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $f(\boldsymbol{x}, y)$ can be regarded as the *confidence* of $y \in \mathcal{Y}$ being the proper label of $\boldsymbol{x}$. Specifically, given a multi-label example $(\boldsymbol{x}, Y)$, $f(\cdot,\cdot)$ should yield larger output on the *relevant* label $y' \in Y$ and smaller output on the *irrelevant* label $y'' \notin Y$, i.e. $f(\boldsymbol{x}, y') > f(\boldsymbol{x}, y'')$. Note that the multi-label classifier $h(\cdot)$ can be derived from the real-valued function $f(\cdot,\cdot)$ via: $h(\boldsymbol{x}) = \{y \mid f(\boldsymbol{x}, y) > t(\boldsymbol{x}), y \in \mathcal{Y}\}$, where $t{:}\mathcal{X} \to \mathbb{R}$ acts as a *thresholding function* which dichotomizes the label space into relevant and irrelevant label sets.

For ease of reference, Table 1 lists major notations used throughout this review along with their mathematical meanings.

2.1.2 Key Challenge

It is evident that traditional supervised learning can be regarded as a degenerated version of multi-label learning if each example is confined to have only one single label. However, the generality of multi-label learning inevitably makes the corresponding learning task much more difficult to solve. Actually, the key challenge of learning from multi-label data lies in the overwhelming size of output space, i.e. the number of label sets grows *exponentially* as the number of class labels increases. For example, for a label space with 20 class labels ($q = 20$), the number of possible label sets would exceed one million (i.e. 2^{20}).

To cope with the challenge of exponential-sized output space, it is essential to facilitate the learning process by exploiting *correlations* (or dependency) among labels [95], [106]. For example, the probability of an image being annotated with label *Brazil* would be high if we know it has labels *rainforest* and *soccer*; A document is unlikely to be labeled as *entertainment* if it is related to *politics*. Therefore, effective exploitation of the label correlations information is deemed to be crucial for the success of multi-label learning techniques. Existing strategies to label correlations exploitation could among others be roughly categorized into three families, based on the *order of correlations* that the learning techniques have considered [106]:

- *First-order strategy*: The task of multi-label learning is tackled in a *label-by-label* style and thus ignoring co-existence of the other labels, such as decomposing the multi-label learning problem into a number of independent binary classification problems (one per label) [5], [16], [108]. The prominent merit of first-order strategy lies in its conceptual simplicity and high efficiency. On the other hand, the effectiveness of the resulting approaches might be suboptimal due to the ignorance of label correlations.
- *Second-order strategy*: The task of multi-label learning is tackled by considering *pairwise* relations between labels, such as the ranking between relevant label and irrelevant label [27], [30], [107], or interaction between any pair of labels [33], [67], [97], [114], etc. As label correlations are exploited to some extent by second-order strategy, the resulting approaches can achieve good generalization performance. However, there are certain real-world applications where label correlations go beyond the second-order assumption.
- *High-order strategy*: The task of multi-label learning is tackled by considering high-order relations among labels such as imposing all other labels' influences on each label [13], [34], [47], [103], or addressing connections among random subsets of labels [71], [72], [94], etc. Apparently high-order strategy has stronger correlation-modeling capabilities than first-order and second-order strategies, while on the other hand is computationally more demanding and less scalable.

In Section 3, a number of multi-label learning algorithms adopting different strategies will be described in detail to better demonstrate the respective pros and cons of each strategy.

2.1.3 Threshold Calibration

As mentioned in Subsection 2.1.1, a common practice in multi-label learning is to return some real-valued function $f(\cdot,\cdot)$ as the learned model [95]. In this case, in

3. In this paper, the term "multi-label learning" is used in equivalent sense as "multi-label classification" since labels assigned to each instance are considered to be binary. Furthermore, there are alternative multi-label settings where other than a single instance each example is represented by a bag of instances [113] or graphs [54], or extra ontology knowledge might exist on the label space such as hierarchy structure [2], [100]. To keep the review comprehensive yet well-focused, examples are assumed to adopt *single-instance* representation and possess *flat* class labels.

TABLE 1
Summary of Major Mathematical Notations

Notations	Mathematical Meanings
$\mathcal{X}$	d-dimensional instance space $\mathbb{R}^d$ (or $\mathbb{Z}^d$)
$\mathcal{Y}$	label space with q possible class labels $\{y_1, y_2, \cdots, y_q\}$
$\boldsymbol{x}$	d-dimensional feature vector $(x_1, x_2, \cdots, x_d)^\top$ ($\boldsymbol{x} \in \mathcal{X}$)
Y	label set associated with $\boldsymbol{x}$ ($Y \subseteq \mathcal{Y}$)
$\bar{Y}$	complementary set of Y in $\mathcal{Y}$
$\mathcal{D}$	multi-label training set $\{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq m\}$
$\mathcal{S}$	multi-label test set $\{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq p\}$
$h(\cdot)$	multi-label classifier $h : \mathcal{X} \to 2^{\mathcal{Y}}$, where $h(\boldsymbol{x})$ returns the set of proper labels for $\boldsymbol{x}$
$f(\cdot, \cdot)$	real-valued function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, where $f(\boldsymbol{x}, y)$ returns the confidence of y being proper label of $\boldsymbol{x}$
$rank_f(\cdot, \cdot)$	$rank_f(\boldsymbol{x}, y)$ returns the rank of y in $\mathcal{Y}$ based on the descending order induced from $f(\boldsymbol{x}, \cdot)$
$t(\cdot)$	thresholding function $t : \mathcal{X} \to \mathbb{R}$, where $h(\boldsymbol{x}) = \{y \mid f(\boldsymbol{x}, y) > t(\boldsymbol{x}), y \in \mathcal{Y}\}$
$\lvert \cdot \rvert$	$\lvert\mathcal{A}\rvert$ returns the cardinality of set $\mathcal{A}$
$[\![\cdot]\!]$	$[\![\pi]\!]$ returns 1 if predicate π holds, and 0 otherwise
$\phi(\cdot, \cdot)$	$\phi(Y, y)$ returns $+1$ if $y \in Y$, and -1 otherwise
$\mathcal{D}_j$	binary training set $\{(\boldsymbol{x}_i, \phi(Y_i, y_j)) \mid 1 \leq i \leq m\}$ derived from $\mathcal{D}$ for the j-th class label y_j
$\psi(\cdot, \cdot, \cdot)$	$\psi(Y, y_j, y_k)$ returns $+1$ if $y_j \in Y$ and $y_k \notin Y$, and -1 if $y_j \notin Y$ and $y_k \in Y$
$\mathcal{D}_{jk}$	binary training set $\{(\boldsymbol{x}_i, \psi(Y_i, y_j, y_k)) \mid \phi(Y_i, y_j) \neq \phi(Y_i, y_k), 1 \leq i \leq m\}$ derived from $\mathcal{D}$ for the label pair (y_j, y_k)
$\sigma_{\mathcal{Y}}(\cdot)$	injective function $\sigma_{\mathcal{Y}} : 2^{\mathcal{Y}} \to \mathbb{N}$ mapping from the power set of $\mathcal{Y}$ to natural numbers ($\sigma_{\mathcal{Y}}^{-1}$ being the corresponding inverse function)
$\mathcal{D}_{\mathcal{Y}}^{\dagger}$	multi-class (single-label) training set $\{(\boldsymbol{x}_i, \sigma_{\mathcal{Y}}(Y_i)) \mid 1 \leq i \leq m\}$ derived from $\mathcal{D}$
$\mathcal{B}$	binary learning algorithm [complexity: $\mathcal{F}_{\mathcal{B}}(m, d)$ for training; $\mathcal{F}'_{\mathcal{B}}(d)$ for (per-instance) testing]
$\mathcal{M}$	multi-class learning algorithm [complexity: $\mathcal{F}_{\mathcal{M}}(m, d, q)$ for training; $\mathcal{F}'_{\mathcal{M}}(d, q)$ for (per-instance) testing]

order to decide the proper label set for unseen instance $\boldsymbol{x}$ (i.e. $h(\boldsymbol{x})$), the real-valued output $f(\boldsymbol{x}, y)$ on each label should be calibrated against the thresholding function output $t(\boldsymbol{x})$.

Generally, threshold calibration can be accomplished with two strategies, i.e. setting $t(\cdot)$ as constant function or inducing $t(\cdot)$ from the training examples [44]. For the first strategy, as $f(\boldsymbol{x}, y)$ takes value in $\mathbb{R}$, one straightforward choice is to use zero as the calibration constant [5]. Another popular choice for calibration constant is 0.5 when $f(\boldsymbol{x}, y)$ represents the posterior probability of y being a proper label of $\boldsymbol{x}$ [16]. Furthermore, when all the unseen instances in the test set are available, the calibration constant can be set to minimize the difference on certain multi-label indicator between the training set and test set, notably the label cardinality [72].

For the second strategy, a *stacking*-style procedure would be used to determine the thresholding function [27], [69], [107]. One popular choice is to assume a linear model for $t(\cdot)$, i.e. $t(\boldsymbol{x}) = \langle \boldsymbol{w}^*, \boldsymbol{f}^*(\boldsymbol{x}) \rangle + b^*$ where $\boldsymbol{f}^*(\boldsymbol{x}) = (f(\boldsymbol{x}, y_1), \ldots, f(\boldsymbol{x}, y_q))^{\mathrm{T}} \in \mathbb{R}^q$ is a q-dimensional stacking vector storing the learning system's real-valued outputs on each label. Specifically, to work out the q-dimensional weight vector $\boldsymbol{w}^*$ and bias b^*, the following *linear least squares* problem is solved based on the training set $\mathcal{D}$:

$$\min_{\{\boldsymbol{w}^*, b^*\}} \quad \sum_{i=1}^{m} \left(\langle \boldsymbol{w}^*, \boldsymbol{f}^*(\boldsymbol{x}_i) \rangle + b^* - s(\boldsymbol{x}_i) \right)^2. \tag{1}$$

Here, $s(\boldsymbol{x}_i) = \arg\min_{a \in \mathbb{R}} \left(\lvert\{y_j \mid y_j \in Y_i, f(\boldsymbol{x}_i, y_j) \leq a\}\rvert + \lvert\{y_k \mid y_k \in \bar{Y}_i, f(\boldsymbol{x}_i, y_k) \geq a\}\rvert \right)$ represents the *target output* of the stacking model which bipartitions $\mathcal{Y}$ into relevant and irrelevant labels for each training example with minimum misclassifications.

All the above threshold calibration strategies are *general-purpose* techniques which could be used as a post-processing step to any multi-label learning algorithm returning real-valued function $f(\cdot, \cdot)$. Accordingly, there also exist some *ad hoc* threshold calibration techniques which are specific to the learning algorithms [30], [94] and will be introduced as their inherent component in Section 3. Instead of utilizing the thresholding function $t(\cdot)$, an equivalent mechanism to induce $h(\cdot)$ from $f(\cdot, \cdot)$ is to specify the number of relevant labels for each example with $t' : \mathcal{X} \to \{1, 2, \ldots, q\}$ such that $h(\boldsymbol{x}) = \{y \mid rank_f(\boldsymbol{x}, y) \leq t'(\boldsymbol{x})\}$ [44], [82]. Here, $rank_f(\boldsymbol{x}, y)$ returns the rank of y when all class labels in $\mathcal{Y}$ are sorted in descending order based on $f(\boldsymbol{x}, \cdot)$.

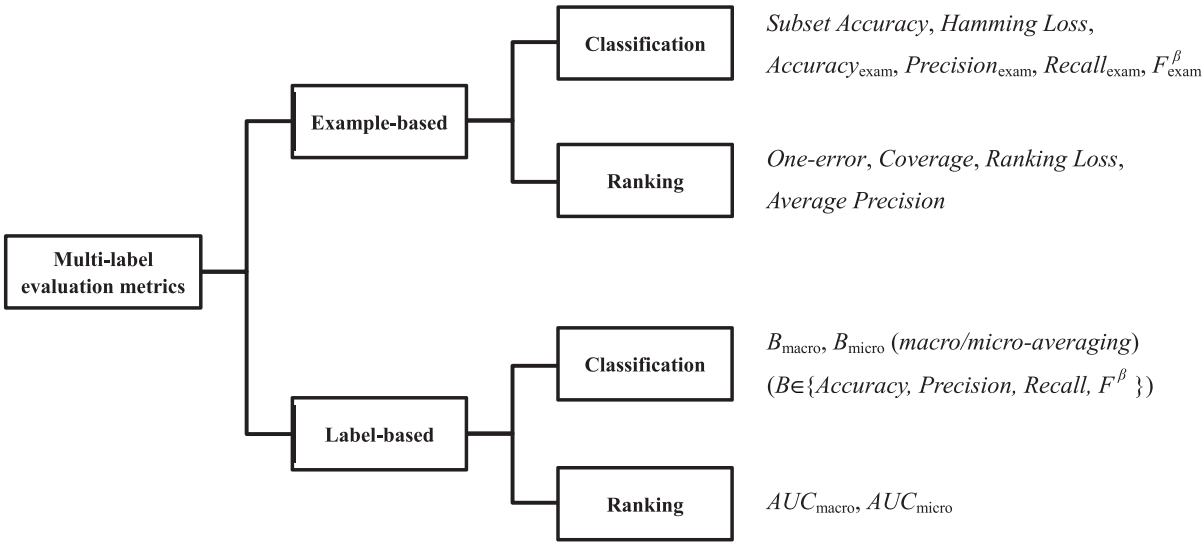

Fig. 1. Summary of major multi-label evaluation metrics.

2.2 Evaluation Metrics

2.2.1 Brief Taxonomy

In traditional supervised learning, generalization performance of the learning system is evaluated with conventional metrics such as accuracy, F-measure, area under the ROC curve (AUC), etc. However, performance evaluation in multi-label learning is much complicated than traditional single-label setting, as each example can be associated with multiple labels simultaneously. Therefore, a number of evaluation metrics specific to multi-label learning are proposed, which can be generally categorized into two groups, i.e. *example-based* metrics [33], [34], [75] and *label-based* metrics [94].

Following the notations in Table 1, let $\mathcal{S} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq p)\}$ be the test set and $h(\cdot)$ be the learned multi-label classifier. Example-based metrics work by evaluating the learning system's performance on each test example separately, and then returning the *mean value* across the test set. Different to the above example-based metrics, label-based metrics work by evaluating the learning system's performance on each class label separately, and then returning the *macro/micro-averaged value* across all class labels.

Note that with respect to $h(\cdot)$, the learning system's generalization performance is measured from *classification* perspective. However, for either example-based or label-based metrics, with respect to the real-valued function $f(\cdot,\cdot)$ which is returned by most multi-label learning systems as a common practice, the generalization performance can also be measured from *ranking* perspective. Fig. 1 summarizes the major multi-label evaluation metrics to be introduced next.

2.2.2 Example-based Metrics

Following the notations in Table 1, six example-based classification metrics can be defined based on the multi-label classifier $h(\cdot)$ [33], [34], [75]:

- *Subset Accuracy*:

$$subsetacc(h) = \frac{1}{p}\sum_{i=1}^{p} [\![h(\boldsymbol{x}_i) = Y_i]\!].$$

The subset accuracy evaluates the fraction of correctly classified examples, i.e. the predicted label set is identical to the ground-truth label set. Intuitively, subset accuracy can be regarded as a multi-label counterpart of the traditional accuracy metric, and tends to be overly strict especially when the size of label space (i.e. q) is large.

- *Hamming Loss*:

$$hloss(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{q}\,|h(\boldsymbol{x}_i)\Delta Y_i|.$$

Here, Δ stands for the symmetric difference between two sets. The hamming loss evaluates the fraction of misclassified instance-label pairs, i.e. a relevant label is missed or an irrelevant is predicted. Note that when each example in $\mathcal{S}$ is associated with only one label, $\text{hloss}_{\mathcal{S}}(h)$ will be $2/q$ times of the traditional misclassification rate.

- $Accuracy_{\text{exam}}$, $Precision_{\text{exam}}$, $Recall_{\text{exam}}$, F^{β}_{exam}:

$$Accuracy_{\text{exam}}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|Y_i \bigcap h(\boldsymbol{x}_i)|}{|Y_i \bigcup h(\boldsymbol{x}_i)|};$$

$$Precision_{\text{exam}}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|Y_i \bigcap h(\boldsymbol{x}_i)|}{|h(\boldsymbol{x}_i)|}$$

$$Recall_{\text{exam}}(h) = \frac{1}{p}\sum_{i=1}^{p}\frac{|Y_i \bigcap h(\boldsymbol{x}_i)|}{|Y_i|};$$

$$F^{\beta}_{\text{exam}}(h) = \frac{(1+\beta^2)\cdot Precision_{\text{exam}}(h)\cdot Recall_{\text{exam}}(h)}{\beta^2\cdot Precision_{\text{exam}}(h) + Recall_{\text{exam}}(h)}.$$

Furthermore, F^{β}_{exam} is an integrated version of $Precision_{\text{exam}}(h)$ and $Recall_{\text{exam}}(h)$ with balancing factor $\beta > 0$. The most common choice is $\beta = 1$ which leads to the *harmonic mean* of precision and recall.

When the intermediate real-valued function $f(\cdot,\cdot)$ is available, four example-based ranking metrics can be defined as well [75]:

- *One-error*:

$$one\text{-}error(f) = \frac{1}{p}\sum_{i=1}^{p} [\![\left[\arg\max_{y\in\mathcal{Y}} f(\boldsymbol{x}_i, y)\right] \notin Y_i]\!].$$

The one-error evaluates the fraction of examples whose top-ranked label is not in the relevant label set.

- *Coverage*:

$$coverage(f) = \frac{1}{p}\sum_{i=1}^{p} \max_{y\in Y_i} rank_f(\boldsymbol{x}_i, y) - 1.$$

The coverage evaluates how many steps are needed, on average, to move down the ranked label list so as to cover all the relevant labels of the example.

- *Ranking Loss*:

$$rloss(f) = \frac{1}{p}\sum_{i=1}^{p} \frac{1}{|Y_i||\bar{Y}_i|} |\{(y', y'') \mid f(\boldsymbol{x}_i, y') \leq f(\boldsymbol{x}_i, y''),\ (y', y'') \in Y_i \times \bar{Y}_i)\}|.$$

The ranking loss evaluates the fraction of reversely ordered label pairs, i.e. an irrelevant label is ranked higher than a relevant label.

- *Average Precision*:

$$avgprec(f) = \frac{1}{p}\sum_{i=1}^{p} \frac{1}{|Y_i|} \sum_{y\in Y_i} \frac{|\{y' \mid rank_f(\boldsymbol{x}_i, y') \leq rank_f(\boldsymbol{x}_i, y),\ y' \in Y_i\}|}{rank_f(\boldsymbol{x}_i, y)}.$$

The average precision evaluates the average fraction of relevant labels ranked higher than a particular label $y \in Y_i$.

For *one-error*, *coverage* and *ranking loss*, the smaller the metric value the better the system's performance, with optimal value of $\frac{1}{p}\sum_{i=1}^{p}|Y_i| - 1$ for *coverage* and 0 for *one-error* and *ranking loss*. For the other example-based multi-label metrics, the larger the metric value the better the system's performance, with optimal value of 1.

2.2.3 Label-based Metrics

For the j-th class label y_j, four basic quantities characterizing the binary classification performance on this label can be defined based on $h(\cdot)$:

$$\begin{aligned} TP_j &= |\{\boldsymbol{x}_i \mid y_j \in Y_i \wedge y_j \in h(\boldsymbol{x}_i), 1 \leq i \leq p\}|; \\ FP_j &= |\{\boldsymbol{x}_i \mid y_j \notin Y_i \wedge y_j \in h(\boldsymbol{x}_i), 1 \leq i \leq p\}| \\ TN_j &= |\{\boldsymbol{x}_i \mid y_j \notin Y_i \wedge y_j \notin h(\boldsymbol{x}_i), 1 \leq i \leq p\}|; \\ FN_j &= |\{\boldsymbol{x}_i \mid y_j \in Y_i \wedge y_j \notin h(\boldsymbol{x}_i), 1 \leq i \leq p\}|. \end{aligned}$$

In other words, TP_j, FP_j, TN_j and FN_j represent the number of *true positive*, *false positive*, *true negative*, and *false negative* test examples with respect to y_j. According to the above definitions, $TP_j + FP_j + TN_j + FN_j = p$ naturally holds.

Based on the above four quantities, most of the binary classification metrics can be derived accordingly. Let $B(TP_j, FP_j, TN_j, FN_j)$ represent some specific binary classification metric ($B \in \{Accuracy, Precision, Recall, F^\beta\}$[4]), the label-based classification metrics can be obtained in either of the following modes [94]:

- *Macro-averaging*:

$$B_{\text{macro}}(h) = \frac{1}{q}\sum_{j=1}^{q} B(TP_j, FP_j, TN_j, FN_j)$$

- *Micro-averaging*:

$$B_{\text{micro}}(h) = B\left(\sum_{j=1}^{q} TP_j,\ \sum_{j=1}^{q} FP_j,\ \sum_{j=1}^{q} TN_j,\ \sum_{j=1}^{q} FN_j\right)$$

Conceptually speaking, macro-averaging and micro-averaging assume "equal weights" for labels and examples respectively. It is not difficult to show that both $Accuracy_{\text{macro}}(h) = Accuracy_{\text{micro}}(h)$ and $Accuracy_{\text{micro}}(h) + hloss(h) = 1$ hold. Note that the macro/micro-averaged version ($B_{\text{macro}}/B_{\text{micro}}$) is different to the example-based version in Subsection 2.2.2.

When the intermediate real-valued function $f(\cdot,\cdot)$ is available, one label-based ranking metric, i.e. macro-averaged AUC, can be derived as:

$$\begin{aligned} AUC_{\text{macro}} &= \frac{1}{q}\sum_{j=1}^{q} AUC_j \\ &= \frac{1}{q}\sum_{j=1}^{q} \frac{|\{(\boldsymbol{x}', \boldsymbol{x}'') | f(\boldsymbol{x}', y_j) \geq f(\boldsymbol{x}'', y_j),\ (\boldsymbol{x}', \boldsymbol{x}'') \in \mathcal{Z}_j \times \bar{\mathcal{Z}}_j|}{|\mathcal{Z}_j||\bar{\mathcal{Z}}_j|}. \end{aligned} \quad (2)$$

Here, $\mathcal{Z}_j = \{\boldsymbol{x}_i \mid y_j \in Y_i, 1 \leq i \leq p\}$ $\left(\bar{\mathcal{Z}}_j = \{\boldsymbol{x}_i \mid y_j \notin Y_i, 1 \leq i \leq p\}\right)$ corresponds to the set of test instances with (without) label y_j. The second line of Eq.(2) follows from the close relation between AUC and the Wilcoxon-Mann-Whitney statistic [39]. Correspondingly, the micro-averaged AUC can also be derived as:

$$AUC_{\text{micro}} = \frac{|\{(\boldsymbol{x}', \boldsymbol{x}'', y', y'') | f(\boldsymbol{x}', y') \geq f(\boldsymbol{x}'', y''),\ (\boldsymbol{x}', y') \in \mathcal{S}^+,\ (\boldsymbol{x}'', y'') \in \mathcal{S}^-\}|}{|\mathcal{S}^+||\mathcal{S}^-|}.$$

Here, $\mathcal{S}^+ = \{(\boldsymbol{x}_i, y) \mid y \in Y_i, 1 \leq i \leq p\}$ $\left(\mathcal{S}^- = \{(\boldsymbol{x}_i, y) \mid y \notin Y_i, 1 \leq i \leq p\}\right)$ corresponds to the set of relevant (irrelevant) instance-label pairs.

For the above label-based multi-label metrics, the larger the metric value the better the system's performance, with optimal value of 1.

2.2.4 Theoretical Results

Based on the metric definitions, it is obvious that existing multi-label metrics consider the performance from diverse aspects and are thus of different natures. As shown in Section 3, most multi-label learning algorithms actually learn from training examples by explicitly or implicitly optimizing one specific metric. In light of fair and honest evaluation, performance of the multi-label learning

4. For example, $Accuracy(TP_j, FP_j, TN_j, FN_j) = \frac{TP_j + TN_j}{TP_j + FP_j + TN_j + FN_j}$, $Precision(TP_j, FP_j, TN_j, FN_j) = \frac{TP_j}{TP_j + FP_j}$, $Recall(TP_j, FP_j, TN_j, FN_j) = \frac{TP_j}{TP_j + FN_j}$, and $F^\beta(TP_j, FP_j, TN_j, FN_j) = \frac{(1+\beta^2)\cdot TP_j}{(1+\beta^2)\cdot TP_j + \beta^2 \cdot FN_j + FP_j}$.

algorithm should therefore be tested on a broad range of metrics instead of only on the one being optimized. Specifically, recent theoretical studies show that classifiers aim at maximizing the *subset accuracy* would perform rather poor if evaluated in terms of *hamming loss*, and vice versa [22], [23].

As multi-label metrics are usually non-convex and discontinuous, in practice most learning algorithms resort to optimizing (convex) *surrogate* multi-label metrics [65], [66]. Recently, the *consistency* of multi-label learning [32] has been studied, i.e. whether the expected loss of a learned classifier converges to the Bayes loss as the training set size increases. Specifically, a necessary and sufficient condition for consistency of multi-label learning based on surrogate loss functions is given, which is intuitive and can be informally stated as that for a fixed distribution over $\mathcal{X} \times 2^{\mathcal{Y}}$, the set of classifiers yielding optimal surrogate loss must fall in the set of classifiers yielding optimal original multi-label loss.

By focusing on *ranking loss*, it is disclosed that none pairwise convex surrogate loss defined on label pairs is consistent with the ranking loss and some recent multi-label approach [40] is inconsistent even for *deterministic* multi-label learning [32].[5] Interestingly, in contrast to this negative result, a complementary positive result on consistent multi-label learning is reported for ranking loss minimization [21]. By using a reduction to the bipartite ranking problem [55], simple univariate convex surrogate loss (exponential or logistic) defined on single labels is shown to be consistent with the ranking loss with explicit regret bounds and convergence rates.

3 Learning Algorithms

3.1 Simple Categorization

Algorithm development always stands as the core issue of machine learning researches, with multi-label learning being no exception. During the past decade, significant amount of algorithms have been proposed to learning from multi-label data. Considering that it is infeasible to go through all existing algorithms within limited space, in this review we opt for scrutinizing a total of eight representative multi-label learning algorithms. Here, the *representativeness* of those selected algorithms are maintained with respect to the following criteria: a) Broad spectrum: each algorithm has unique characteristics covering a variety of algorithmic design strategies; b) Primitive impact: most algorithms lead to a number of follow-up or related methods along its line of research; and c) Favorable influence: each algorithm is among the highly-cited works in the multi-label learning field.[6]

As we try to keep the selection less biased with the above criteria, one should notice that the eight algorithms to be detailed by no means exclude the importance of other methods. Furthermore, for the sake of notational consistency and mathematical rigor, we have chosen to rephrase and present each algorithm under common notations. In this paper, a simple categorization of multi-label learning algorithms is adopted [95]:

Problem transformation methods: This category of algorithms tackle multi-label learning problem by transforming it into other well-established learning scenarios. Representative algorithms include first-order approaches Binary Relevance [5] and high-order approach Classifier Chains [72] which transform the task of multi-label learning into the task of binary classification, second-order approach Calibrated Label Ranking [30] which transforms the task of multi-label learning into the task of label ranking, and high-order approach Random k-labelsets [94] which transforms the task of multi-label learning into the task of multi-class classification.

Algorithm adaptation methods: This category of algorithms tackle multi-label learning problem by adapting popular learning techniques to deal with multi-label data directly. Representative algorithms include first-order approach ML-kNN [108] adapting lazy learning techniques, first-order approach ML-DT [16] adapting decision tree techniques, second-order approach Rank-SVM [27] adapting kernel techniques, and second-order approach CML [33] adapting information-theoretic techniques.

Briefly, the key philosophy of problem transformation methods is to *fit data to algorithm*, while the key philosophy of algorithm adaptation methods is to *fit algorithm to data*. Fig. 2 summarizes the above-mentioned algorithms to be detailed in the rest of this section.

3.2 Problem Transformation Methods

3.2.1 Binary Relevance

The basic idea of this algorithm is to decompose the multi-label learning problem into q *independent* binary classification problems, where each binary classification problem corresponds to a possible label in the label space [5].

Following the notations in Table 1, for the j-th class label y_j, Binary Relevance firstly constructs a corresponding *binary* training set by considering the relevance of each training example to y_j:

$$\mathcal{D}_j = \{(\boldsymbol{x}_i, \phi(Y_i, y_j)) \mid 1 \le i \le m\} \tag{3}$$

$$\text{where} \quad \phi(Y_i, y_j) = \begin{cases} +1, & \text{if } y_j \in Y_i \\ -1, & \text{otherwise.} \end{cases}$$

After that, some binary learning algorithm $\mathcal{B}$ is utilized to induce a binary classifier $g_j{:}\mathcal{X} \to \mathbb{R}$, i.e. $g_j \leftarrow \mathcal{B}(\mathcal{D}_j)$. Therefore, for any multi-label training example $(\boldsymbol{x}_i, Y_i)$, instance $\boldsymbol{x}_i$ will be involved in the learning process of q binary classifiers. For relevant label $y_j \in Y_i$, $\boldsymbol{x}_i$ is regarded as one *positive* instance in inducing $g_j(\cdot)$; On the other hand, for irrelevant label $y_k \in \bar{Y}_i$, $\boldsymbol{x}_i$ is regarded as one *negative* instance. The above training strategy is termed as *cross-training* in [5].

For unseen instance $\boldsymbol{x}$, Binary Relevance predicts its associated label set Y by querying labeling relevance on each individual binary classifier and then combing relevant labels:

$$Y = \{y_j \mid g_j(\boldsymbol{x}) > 0,\ 1 \le j \le q\}. \tag{4}$$

5. Here, deterministic multi-label learning corresponds to the easier learning case where for any instance $\boldsymbol{x} \in \mathcal{X}$, there exists a label subset $Y \subseteq \mathcal{Y}$ such that the posteriori probability of observing Y given $\boldsymbol{x}$ is greater than 0.5, i.e. $\mathbb{P}(Y \mid \boldsymbol{x}) > 0.5$.

6. According to Google Scholar statistics (by January 2013), each paper for the eight algorithms has received at least 90 citations, with more than 200 citations on average.

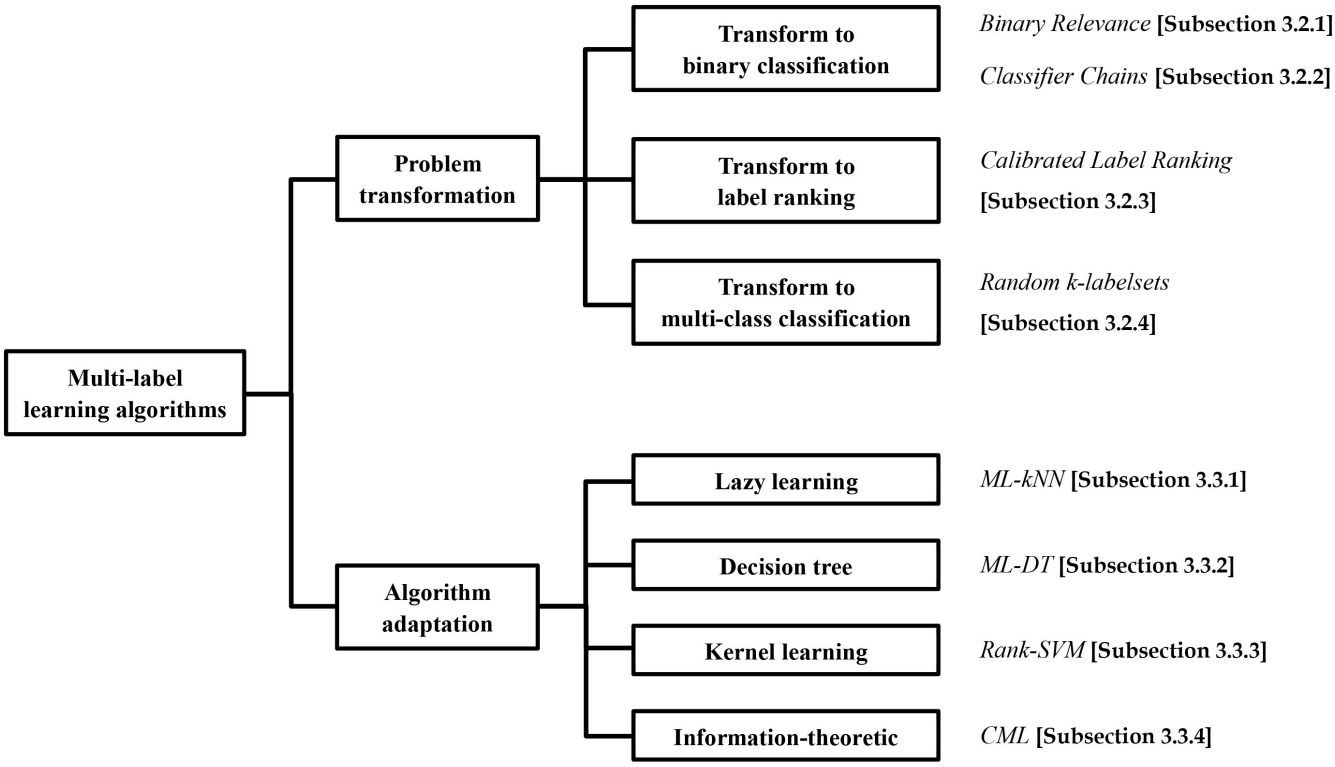

Fig. 2. Categorization of representative multi-label learning algorithms being reviewed.

Note that when all the binary classifiers yield negative outputs, the predicted label set Y would be empty. To avoid producing empty prediction, the following *T-Criterion* rule can be applied:

$$Y = \{y_j \mid g_j(\boldsymbol{x}) > 0,\ 1 \le j \le q\} \bigcup \{y_{j^*} \mid j^* = \arg\max_{1 \le j \le q} g_j(\boldsymbol{x})\}. \quad (5)$$

Briefly, when none of the binary classifiers yield positive predictions, T-Criterion rule complements Eq.(4) by including the class label with *greatest* (least negative) output. In addition to T-Criterion, some other rules toward label set prediction based on the outputs of each binary classifier can be found in [5].

`Remarks:` The pseudo-code of Binary Relevance is summarized in Fig. 3. It is a *first-order* approach which builds classifiers for each label separately and offers the natural opportunity for parallel implementation. The most prominent advantage of Binary Relevance lies in its extremely straightforward way of handling multi-label data (Steps 1-4), which has been employed as the building block of many state-of-the-art multi-label learning techniques [20], [34], [72], [106]. On the other hand, Binary Relevance completely ignores potential correlations among labels, and the binary classifier for each label may suffer from the issue of class-imbalance when q is large and label density (i.e. $LDen(\mathcal{D})$) is low. As shown in Fig. 3, Binary Relevance has computational complexity of $\mathcal{O}(q \cdot \mathcal{F}_{\mathcal{B}}(m, d))$ for training and $\mathcal{O}(q \cdot \mathcal{F}'_{\mathcal{B}}(d))$ for testing.[7]

Y=BinaryRelevance($\mathcal{D}$, $\mathcal{B}$, $\boldsymbol{x}$)
1. **for** $j = 1$ to q **do**
2. Construct the binary training set $\mathcal{D}_j$ according to Eq.(3);
3. $g_j \leftarrow \mathcal{B}(\mathcal{D}_j)$;
4. **endfor**
5. Return Y according to Eq.(5);

Fig. 3. Pseudo-code of binary relevance.

3.2.2 Classifier Chains

The basic idea of this algorithm is to transform the multi-label learning problem into a *chain* of binary classification problems, where subsequent binary classifiers in the chain is built upon the predictions of preceding ones [72], [73].

For q possible class labels $\{y_1, y_2, \ldots, y_q\}$, let τ:$\{1, \ldots, q\} \rightarrow \{1, \ldots, q\}$ be a permutation function which is used to specify an *ordering* over them, i.e. $y_{\tau(1)} \succ y_{\tau(2)} \succ \cdots \succ y_{\tau(q)}$. For the j-th label $y_{\tau(j)}$ $(1 \le j \le q)$ in the ordered list, a corresponding binary training set is constructed by appending each instance with its relevance to those labels preceding $y_{\tau(j)}$:

$$\mathcal{D}_{\tau(j)} = \{\left([\boldsymbol{x}_i, \mathbf{pre}^i_{\tau(j)}], \phi(Y_i, y_{\tau(j)})\right) \mid 1 \le i \le m\} \quad (6)$$
$$\text{where } \mathbf{pre}^i_{\tau(j)} = (\phi(Y_i, y_{\tau(1)}), \ldots, \phi(Y_i, y_{\tau(j-1)}))^\top.$$

Here, $[\boldsymbol{x}_i, \mathbf{pre}^i_{\tau(j)}]$ concatenates vectors $\boldsymbol{x}_i$ and $\mathbf{pre}^i_{\tau(j)}$, and $\mathbf{pre}^i_{\tau(j)}$ represents the binary assignment of those labels preceding $y_{\tau(j)}$ on $\boldsymbol{x}_i$ (specifically, $\mathbf{pre}^i_{\tau(1)} = \emptyset$).[8] After that,

7. In this paper, computational complexity is mainly examined with respect to three factors which are common for all learning algorithms, i.e.: m (number of training examples), d (dimensionality) and q (number of possible class labels). Furthermore, for binary (multi-class) learning algorithm $\mathcal{B}$ ($\mathcal{M}$) embedded in problem transformation methods, we denote its training complexity as $\mathcal{F}_{\mathcal{B}}(m, d)$ ($\mathcal{F}_{\mathcal{M}}(m, d, q)$) and its (per-instance) testing complexity as $\mathcal{F}'_{\mathcal{B}}(d)$ ($\mathcal{F}'_{\mathcal{M}}(d, q)$). All computational complexity results reported in this paper are the worst-case bounds.

8. In Classifier Chains [72], [73], binary assignment is represented by 0 and 1. Without loss of generality, binary assignment is represented by -1 and +1 in this paper for notational consistency.

Y=ClassifierChains($\mathcal{D}$, $\mathcal{B}$, τ, $\boldsymbol{x}$)

1. **for** $j = 1$ to q **do**
2. Construct the chaining binary training set $\mathcal{D}_{\tau(j)}$ according to Eq.(6);
3. $g_{\tau(j)} \leftarrow \mathcal{B}(\mathcal{D}_{\tau(j)})$;
4. **endfor**
5. Return Y according to Eq.(8) (in conjunction with Eq.(7));

Fig. 4. Pseudo-code of classifier chains.

some binary learning algorithm $\mathcal{B}$ is utilized to induce a binary classifier $g_{\tau(j)}$:$\mathcal{X} \times \{-1, +1\}^{j-1} \rightarrow \mathbb{R}$, i.e. $g_{\tau(j)} \leftarrow \mathcal{B}(\mathcal{D}_{\tau(j)})$. In other words, $g_{\tau(j)}(\cdot)$ determines whether $y_{\tau(j)}$ is a relevant label or not.

For unseen instance $\boldsymbol{x}$, its associated label set Y is predicted by traversing the classifier chain iteratively. Let $\lambda^{\boldsymbol{x}}_{\tau(j)} \in \{-1, +1\}$ represent the predicted binary assignment of $y_{\tau(j)}$ on $\boldsymbol{x}$, which are recursively derived as follows:

$$\lambda^{\boldsymbol{x}}_{\tau(1)} = \mathrm{sign}\left[g_{\tau(1)}(\boldsymbol{x})\right] \qquad (7)$$
$$\lambda^{\boldsymbol{x}}_{\tau(j)} = \mathrm{sign}\left[g_{\tau(j)}([\boldsymbol{x}, \lambda^{\boldsymbol{x}}_{\tau(1)}, \ldots, \lambda^{\boldsymbol{x}}_{\tau(j-1)}])\right] \quad (2 \le j \le q).$$

Here, sign[$\cdot$] is the signed function. Accordingly, the predicted label set corresponds to:

$$Y = \{y_{\tau(j)} \mid \lambda^{\boldsymbol{x}}_{\tau(j)} = +1,\ 1 \le j \le q\}. \qquad (8)$$

It is obvious that for the classifier chain obtained as above, its effectiveness is largely affected by the ordering specified by τ. To account for the effect of ordering, an *Ensemble of Classifier Chains* can be built with n random permutations over the label space, i.e. $\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(n)}$. For each permutation $\tau^{(r)}$ $(1 \le r \le n)$, instead of inducing one classifier chain by applying $\tau^{(r)}$ directly on the original training set $\mathcal{D}$, a modified training set $\mathcal{D}^{(r)}$ is used by sampling $\mathcal{D}$ *without* replacement ($|\mathcal{D}^{(r)}| = 0.67 \cdot |\mathcal{D}|$) [72] or *with* replacement ($|\mathcal{D}^{(r)}| = |\mathcal{D}|$) [73].

`Remarks:` The pseudo-code of Classifier Chains is summarized in Fig. 4. It is a *high-order* approach which considers correlations among labels in a random manner. Compared to Binary Relevance [5], Classifier Chains has the advantage of exploiting label correlations while loses the opportunity of parallel implementation due to its chaining property. During the training phase, Classifier Chains augments instance space with extra features from ground-truth labeling (i.e. $\mathbf{pre}^{i}_{\tau(j)}$ in Eq.(6)). Instead of keeping extra features binary-valued, another possibility is to set them to the classifier's probabilistic outputs when the model returned by $\mathcal{B}$ (e.g. Naive Bayes) is capable of yielding posteriori probability [20], [105]. As shown in Fig. 4, Classifier Chains has computational complexity of $\mathcal{O}(q \cdot \mathcal{F}_{\mathcal{B}}(m, d+q))$ for training and $\mathcal{O}(q \cdot \mathcal{F}'_{\mathcal{B}}(d+q))$ for testing.

3.2.3 Calibrated Label Ranking

The basic idea of this algorithm is to transform the multi-label learning problem into the *label ranking* problem, where ranking among labels is fulfilled by techniques of pairwise comparison [30].

For q possible class labels $\{y_1, y_2, \ldots, y_q\}$, a total of $q(q-1)/2$ binary classifiers can be generated by pairwise comparison, one for each label pair (y_j, y_k) $(1 \le j < k \le q)$. Concretely, for each label pair (y_j, y_k), pairwise comparison firstly constructs a corresponding binary training set by considering the *relative* relevance of each training example to y_j and y_k:

$$\mathcal{D}_{jk} = \{(\boldsymbol{x}_i, \psi(Y_i, y_j, y_k)) \mid \phi(Y_i, y_j) \neq \phi(Y_i, y_k), \ 1 \le i \le m\}, \qquad (9)$$

$$\text{where } \psi(Y_i, y_j, y_k) = \begin{cases} +1, & \text{if } \phi(Y_i, y_j) = +1 \text{ and } \phi(Y_i, y_k) = -1 \\ -1, & \text{if } \phi(Y_i, y_j) = -1 \text{ and } \phi(Y_i, y_k) = +1. \end{cases}$$

In other words, only instances with distinct relevance to y_j and y_k will be included in $\mathcal{D}_{jk}$. After that, some binary learning algorithm $\mathcal{B}$ is utilized to induce a binary classifier g_{jk}:$\mathcal{X} \rightarrow \mathbb{R}$, i.e. $g_{jk} \leftarrow \mathcal{B}(\mathcal{D}_{jk})$. Therefore, for any multi-label training example $(\boldsymbol{x}_i, Y_i)$, instance $\boldsymbol{x}_i$ will be involved in the learning process of $|Y_i||\bar{Y}_i|$ binary classifiers. For any instance $\boldsymbol{x} \in \mathcal{X}$, the learning system votes for y_j if $g_{jk}(\boldsymbol{x}) > 0$ and y_k otherwise.

For unseen instance $\boldsymbol{x}$, Calibrated Label Ranking firstly feeds it to the $q(q-1)/2$ trained binary classifiers to obtain the overall votes on each possible class label:

$$\zeta(\boldsymbol{x}, y_j) = \sum_{k=1}^{j-1} [\![g_{kj}(\boldsymbol{x}) \le 0]\!] + \sum_{k=j+1}^{q} [\![g_{jk}(\boldsymbol{x}) > 0]\!] \qquad (1 \le j \le q). \qquad (10)$$

Based on the above definition, it is not difficult to verify that $\sum_{j=1}^{q} \zeta(\boldsymbol{x}, y_j) = q(q-1)/2$. Here, labels in $\mathcal{Y}$ can be ranked according to their respective votes (ties are broken arbitrarily).

Thereafter, some thresholding function should be further specified to bipartition the list of ranked labels into relevant and irrelevant label set. To achieve this within the pairwise comparison framework, Calibrated Label Ranking incorporates a *virtual label* y_{V} into each multi-label training example $(\boldsymbol{x}_i, Y_i)$. Conceptually speaking, the virtual label serves as an *artificial splitting point* between $\boldsymbol{x}_i$'s relevant and irrelevant labels [6]. In other words, y_{V} is considered to be ranked lower than $y_j \in Y_i$ while ranked higher than $y_k \in \bar{Y}_i$.

In addition to the original $q(q-1)/2$ binary classifiers, q auxiliary binary classifiers will be induced, one for each new label pair (y_j, y_{V}) $(1 \le j \le q)$. Similar to Eq.(9), a binary training set corresponding to (y_j, y_{V}) can be constructed as follows:

$$\mathcal{D}_{j\mathrm{V}} = \{(\boldsymbol{x}_i, \varphi(Y_i, y_j, y_{\mathrm{V}})) \mid 1 \le i \le m\} \qquad (11)$$

$$\text{where } \varphi(Y_i, y_j, y_{\mathrm{V}}) = \begin{cases} +1, & \text{if } y_j \in Y_i \\ -1, & \text{otherwise.} \end{cases}$$

Based on this, the binary learning algorithm $\mathcal{B}$ is utilized to induce a binary classifier corresponding to the virtual label $g_{j\mathrm{V}}$:$\mathcal{X} \rightarrow \mathbb{R}$, i.e. $g_{j\mathrm{V}} \leftarrow \mathcal{B}(\mathcal{D}_{j\mathrm{V}})$. After that, the overall votes specified in Eq.(10) will be updated with the newly induced classifiers:

$$\zeta^*(\boldsymbol{x}, y_j) = \zeta(\boldsymbol{x}, y_j) + [\![g_{j\mathrm{V}}(\boldsymbol{x}) > 0]\!] \qquad (1 \le j \le q). \qquad (12)$$

Y=CalibratedLabelRanking($\mathcal{D}$, $\mathcal{B}$, $\boldsymbol{x}$)
1. **for** $j = 1$ to $q - 1$ **do**
2. **for** $k = j + 1$ to q **do**
3. Construct the binary training set $\mathcal{D}_{jk}$ according to Eq.(9);
4. $g_{jk} \leftarrow \mathcal{B}(\mathcal{D}_{jk})$;
5. **endfor**
6. **endfor**
7. **for** $j = 1$ to q **do**
8. Construct the binary training set $\mathcal{D}_{j\mathrm{V}}$ according to Eq.(11);
9. $g_{j\mathrm{V}} \leftarrow \mathcal{B}(\mathcal{D}_{j\mathrm{V}})$;
10. **endfor**
11. Return Y according to Eq.(14) (in conjunction with Eqs.(10)-(13));

Fig. 5. Pseudo-code of calibrated label ranking.

Furthermore, the overall votes on virtual label can be computed as:

$$\zeta^*(\boldsymbol{x}, y_{\mathrm{V}}) = \sum_{j=1}^{q} [\![g_{j\mathrm{V}}(\boldsymbol{x}) \leq 0]\!]. \tag{13}$$

Therefore, the predicted label set for unseen instance $\boldsymbol{x}$ corresponds to:

$$Y = \{y_j \mid \zeta^*(\boldsymbol{x}, y_j) > \zeta^*(\boldsymbol{x}, y_{\mathrm{V}}),\ 1 \leq j \leq q\}. \tag{14}$$

By comparing Eq.(11) to Eq.(3), it is obvious that the training set $\mathcal{D}_{j\mathrm{V}}$ employed by Calibrated Label Ranking is identical to the training set $\mathcal{D}_j$ employed by Binary Relevance [5]. Therefore, Calibrated Label Ranking can be regarded as an enriched version of pairwise comparison, where the routine $q(q-1)/2$ binary classifiers are enlarged with the q binary classifiers of Binary Relevance to facilitate learning [30].

`Remarks:` The pseudo-code of Calibrated Label Ranking is summarized in Fig. 5. It is a *second-order* approach which builds classifiers for any pair of class labels. Compared to previously introduced algorithms [5], [72] which construct binary classifiers in a *one-vs-rest* manner, Calibrated Label Ranking constructs binary classifiers (except those for virtual label) in a *one-vs-one* manner and thus has the advantage of mitigating the negative influence of the class-imbalance issue. On the other hand, the number of binary classifiers constructed by Calibrated Label Ranking grows from linear scale to *quadratic scale* in terms of the number class labels (i.e. q). Improvements on Calibrated Label Ranking mostly focus on reducing the quadratic number of classifiers to be queried in testing phase by exact pruning [59] or approximate pruning [60], [61]. By exploiting idiosyncrasy of the underlying binary learning algorithm $\mathcal{B}$, such as dual representation for Perceptron [58], the quadratic number of classifiers can be induced more efficiently in training phase [57]. As shown in Fig. 5, Calibrated Label Ranking has computational complexity of $\mathcal{O}(q^2 \cdot \mathcal{F}_{\mathcal{B}}(m, d))$ for training and $\mathcal{O}(q^2 \cdot \mathcal{F}'_{\mathcal{B}}(d))$ for testing.

3.2.4 Random k-Labelsets

The basic idea of this algorithm is to transform the multi-label learning problem into an ensemble of *multi-class classification* problems, where each component learner in the ensemble targets a random subset of $\mathcal{Y}$ upon which a multi-class classifier is induced by the Label Powerset (LP) techniques [92], [94].

LP is a straightforward approach to transforming multi-label learning problem into multi-class (single-label) classification problem. Let $\sigma_{\mathcal{Y}}$:$2^{\mathcal{Y}} \rightarrow \mathbb{N}$ be some *injective* function mapping from the power set of $\mathcal{Y}$ to natural numbers, and $\sigma_{\mathcal{Y}}^{-1}$ be the corresponding *inverse* function. In the training phase, LP firstly converts the original multi-label training set $\mathcal{D}$ into the following multi-class training set by treating every distinct label set appearing in $\mathcal{D}$ as a *new class*:

$$\mathcal{D}_{\mathcal{Y}}^{\dagger} = \{(\boldsymbol{x}_i, \sigma_{\mathcal{Y}}(Y_i)) \mid 1 \leq i \leq m\}, \tag{15}$$

where the set of new classes covered by $\mathcal{D}_{\mathcal{Y}}^{\dagger}$ corresponds to:

$$\Gamma\left(\mathcal{D}_{\mathcal{Y}}^{\dagger}\right) = \{\sigma_{\mathcal{Y}}(Y_i) \mid 1 \leq i \leq m\}. \tag{16}$$

Obviously, $\left|\Gamma\left(\mathcal{D}_{\mathcal{Y}}^{\dagger}\right)\right| \leq \min\left(m, 2^{|\mathcal{Y}|}\right)$ holds here. After that, some multi-class learning algorithm $\mathcal{M}$ is utilized to induce a multi-class classifier $g_{\mathcal{Y}}^{\dagger}$:$\mathcal{X} \rightarrow \Gamma\left(\mathcal{D}_{\mathcal{Y}}^{\dagger}\right)$, i.e. $g_{\mathcal{Y}}^{\dagger} \leftarrow \mathcal{M}\left(\mathcal{D}_{\mathcal{Y}}^{\dagger}\right)$. Therefore, for any multi-label training example $(\boldsymbol{x}_i, Y_i)$, instance $\boldsymbol{x}_i$ will be re-assigned with the newly mapped single-label $\sigma_{\mathcal{Y}}(Y_i)$ and then participates in multi-class classifier induction.

For unseen instance $\boldsymbol{x}$, LP predicts its associated label set Y by firstly querying the prediction of multi-class classifier and then mapping it back to the power set of $\mathcal{Y}$:

$$Y = \sigma_{\mathcal{Y}}^{-1}\left(g_{\mathcal{Y}}^{\dagger}(\boldsymbol{x})\right). \tag{17}$$

Unfortunately, LP has two major limitations in terms of practical feasibility: a) Incompleteness: as shown in Eqs.(16) and (17), LP is confined to predict label sets appearing in the training set, i.e. unable to generalize to those outside $\{Y_i \mid 1 \leq i \leq m\}$; b) Inefficiency: when $\mathcal{Y}$ is large, there might be too many newly mapped classes in $\Gamma\left(\mathcal{D}_{\mathcal{Y}}^{\dagger}\right)$, leading to overly high complexity in training $g_{\mathcal{Y}}^{\dagger}(\cdot)$ and extremely few training examples for some newly mapped classes.

To keep LP's simplicity while overcoming its two major drawbacks, Random k-Labelsets chooses to combine ensemble learning [24], [112] with LP to learn from multi-label data. The key strategy is to invoke LP only on random *k-labelsets* (size-k subset in $\mathcal{Y}$) to guarantee computational efficiency, and then ensemble a number of LP classifiers to achieve predictive completeness.

Let $\mathcal{Y}^k$ represent the collection of all possible k-labelsets in $\mathcal{Y}$, where the l-th k-labelset is denoted as $\mathcal{Y}^k(l)$, i.e. $\mathcal{Y}^k(l) \subseteq \mathcal{Y}$, $\left|\mathcal{Y}^k(l)\right| = k$, $1 \leq l \leq \binom{q}{k}$. Similar to Eq.(15), a multi-class training set can be constructed as well by shrinking the original label space $\mathcal{Y}$ into $\mathcal{Y}^k(l)$:

$$\mathcal{D}_{\mathcal{Y}^k(l)}^{\dagger} = \left\{\left(\boldsymbol{x}_i,\ \sigma_{\mathcal{Y}^k(l)}(Y_i \cap \mathcal{Y}^k(l))\right) \;\middle|\; 1 \leq i \leq m\right\}, \tag{18}$$

where the set of new classes covered by $\mathcal{D}_{\mathcal{Y}^k(l)}^{\dagger}$ corresponds to:

$$\Gamma\left(\mathcal{D}_{\mathcal{Y}^k(l)}^{\dagger}\right) = \{\sigma_{\mathcal{Y}^k(l)}(Y_i \cap \mathcal{Y}^k(l)) \mid 1 \leq i \leq m\}.$$

After that, the multi-class learning algorithm $\mathcal{M}$ is utilized to induce a multi-class classifier $g_{\mathcal{Y}^k(l)}^{\dagger}$:$\mathcal{X} \rightarrow \Gamma\left(\mathcal{D}_{\mathcal{Y}^k(l)}^{\dagger}\right)$, i.e. $g_{\mathcal{Y}^k(l)}^{\dagger} \leftarrow \mathcal{M}\left(\mathcal{D}_{\mathcal{Y}^k(l)}^{\dagger}\right)$.

Y=Random$k$-Labelsets($\mathcal{D}$, $\mathcal{M}$, k, n, $\boldsymbol{x}$)

1. **for** $r = 1$ to n **do**
2. Randomly choose a k-labelset $\mathcal{Y}^k(l_r) \subseteq \mathcal{Y}$ with $|\mathcal{Y}^k(l_r)| = k$;
3. Construct the multi-class training set $\mathcal{D}^{\dagger}_{\mathcal{Y}^k(l_r)}$ according to Eq.(18);
4. $g^{\dagger}_{\mathcal{Y}^k(l_r)} \leftarrow \mathcal{M}\left(\mathcal{D}^{\dagger}_{\mathcal{Y}^k(l_r)}\right)$;
5. **endfor**
6. Return Y according to Eq.(20) (in conjunction with Eq.(19));

Fig. 6. Pseudo-code of random k-labelsets.

To create an ensemble with n component classifiers, Random k-Labelsets invokes LP on n random k-labelsets $\mathcal{Y}^k(l_r)$ $(1 \le r \le n)$ each leading to a multi-class classifier $g^{\dagger}_{\mathcal{Y}^k(l_r)}(\cdot)$. For unseen instance $\boldsymbol{x}$, the following two quantities are calculated for each class label:

$$\tau(\boldsymbol{x}, y_j) = \sum_{r=1}^{n} [\![y_j \in \mathcal{Y}^k(l_r)]\!] \qquad (1 \le j \le q) \tag{19}$$
$$\mu(\boldsymbol{x}, y_j) = \sum_{r=1}^{n} \left[\!\!\left[y_j \in \sigma^{-1}_{\mathcal{Y}^k(l_r)}\left(g^{\dagger}_{\mathcal{Y}^k(l_r)}(\boldsymbol{x})\right) \right]\!\!\right] \qquad (1 \le j \le q).$$

Here, $\tau(\boldsymbol{x}, y_j)$ counts the *maximum* number of votes that y_j can be received from the ensemble, while $\mu(\boldsymbol{x}, y_j)$ counts the *actual* number of votes that y_j receives from the ensemble. Accordingly, the predicted label set corresponds to:

$$Y = \{y_j \mid \mu(\boldsymbol{x}, y_j)/\tau(\boldsymbol{x}, y_j) > 0.5,\ 1 \le j \le q\}. \tag{20}$$

In other words, when the actual number of votes exceeds *half* of the maximum number of votes, y_j is regarded to be relevant. For an ensemble created by n k-labelsets, the maximum number of votes on each label is nk/q on average. A rule-of-thumb setting for Random k-Labelsets is $k = 3$ and $n = 2q$ [92], [94].

`Remarks`: The pseudo-code of Random k-Labelsets is summarized in Fig. 6. It is a *high-order* approach where the degree of label correlations is controlled by the size of k-labelsets. In addition to use k-labelset, another way to improve LP is to prune distinct label set in $\mathcal{D}$ appearing less than a pre-specified counting threshold [71]. Although Random k-Labelsets embeds ensemble learning as its *inherent* part to amend LP's major drawbacks, ensemble learning could be employed as a *meta-level* strategy to facilitate multi-label learning by encompassing homogeneous [72], [76] or heterogeneous [74], [83] component multi-label learners. As shown in Fig. 6, Random k-Labelsets has computational complexity of $\mathcal{O}(n \cdot \mathcal{F}_{\mathcal{M}}(m, d, 2^k))$ for training and $\mathcal{O}(n \cdot \mathcal{F}'_{\mathcal{M}}(d, 2^k))$ for testing.

3.3 Algorithm Adaptation Methods

3.3.1 Multi-Label k-Nearest Neighbor (ML-kNN)

The basic idea of this algorithm is to adapt *k-nearest neighbor* techniques to deal with multi-label data, where maximum a posteriori (MAP) rule is utilized to make prediction by reasoning with the labeling information embodied in the neighbors [108].

For unseen instance $\boldsymbol{x}$, let $\mathcal{N}(\boldsymbol{x})$ represent the set of its k nearest neighbors identified in $\mathcal{D}$. Generally, similarity between instances is measured with the Euclidean distance. For the j-th class label, ML-kNN chooses to calculate the following statistics:

$$C_j = \sum_{(\boldsymbol{x}^*, Y^*) \in \mathcal{N}(\boldsymbol{x})} [\![y_j \in Y^*]\!]. \tag{21}$$

Namely, C_j records the number of $\boldsymbol{x}$'s neighbors with label y_j.

Let H_j be the event that $\boldsymbol{x}$ has label y_j, and $\mathbb{P}(H_j \mid C_j)$ represents the *posterior probability* that H_j holds under the condition that $\boldsymbol{x}$ has exactly C_j neighbors with label y_j. Correspondingly, $\mathbb{P}(\neg H_j \mid C_j)$ represents the posterior probability that H_j doesn't hold under the same condition. According to the MAP rule, the predicted label set is determined by deciding whether $\mathbb{P}(H_j \mid C_j)$ is greater than $\mathbb{P}(\neg H_j \mid C_j)$ or not:

$$Y = \{y_j \mid \mathbb{P}(H_j \mid C_j)/\mathbb{P}(\neg H_j \mid C_j) > 1,\ 1 \le j \le q\}. \tag{22}$$

Based on Bayes theorem, we have:

$$\frac{\mathbb{P}(H_j \mid C_j)}{\mathbb{P}(\neg H_j \mid C_j)} = \frac{\mathbb{P}(H_j) \cdot \mathbb{P}(C_j \mid H_j)}{\mathbb{P}(\neg H_j) \cdot \mathbb{P}(C_j \mid \neg H_j)}. \tag{23}$$

Here, $\mathbb{P}(H_j)$ $\left(\mathbb{P}(\neg H_j)\right)$ represents the *prior probability* that H_j holds (doesn't hold). Furthermore, $\mathbb{P}(C_j \mid H_j)$ $\left(\mathbb{P}(C_j \mid \neg H_j)\right)$ represents the *likelihood* that $\boldsymbol{x}$ has exactly C_j neighbors with label y_j when H_j holds (doesn't hold). As shown in Eqs.(22) and (23), it suffices to estimate the prior probabilities as well as likelihoods for making predictions.

ML-kNN fulfills the above task via the *frequency counting* strategy. Firstly, the prior probabilities are estimated by counting the number training examples associated with each label:

$$\mathbb{P}(H_j) = \frac{s + \sum_{i=1}^{m} [\![y_j \in Y_i]\!]}{s \times 2 + m};$$
$$\mathbb{P}(\neg H_j) = 1 - \mathbb{P}(H_j) \qquad (1 \le j \le q). \tag{24}$$

Here, s is a smoothing parameter controlling the effect of *uniform prior* on the estimation. Generally, s takes the value of 1 resulting in Laplace smoothing.

Secondly, the estimation process for likelihoods is somewhat involved. For the j-th class label y_j, ML-kNN maintains two *frequency arrays* κ_j and $\tilde{\kappa}_j$ each containing k+1 elements:

$$\kappa_j[r] = \sum_{i=1}^{m} [\![y_j \in Y_i]\!] \cdot [\![\delta_j(\boldsymbol{x}_i) = r]\!] \quad (0 \le r \le k) \tag{25}$$
$$\tilde{\kappa}_j[r] = \sum_{i=1}^{m} [\![y_j \notin Y_i]\!] \cdot [\![\delta_j(\boldsymbol{x}_i) = r]\!] \quad (0 \le r \le k)$$
$$\text{where } \delta_j(\boldsymbol{x}_i) = \sum_{(\boldsymbol{x}^*, Y^*) \in \mathcal{N}(\boldsymbol{x}_i)} [\![y_j \in Y^*]\!].$$

Here, $\delta_j(\boldsymbol{x}_i)$ records the number of $\boldsymbol{x}_i$'s neighbors with label y_j. Therefore, $\kappa_j[r]$ counts the number of training examples which have label y_j and have exactly r neighbors with label y_j, while $\tilde{\kappa}_j[r]$ counts the number of training examples which don't have label y_j and have exactly r neighbors with label y_j. Afterwards, the likelihoods can be estimated based on κ_j and $\tilde{\kappa}_j$:

$$\mathbb{P}(C_j \mid H_j) = \frac{s + \kappa_j[C_j]}{s \times (k+1) + \sum_{r=0}^{k} \kappa_j[r]} \qquad (1 \le j \le q,\ 0 \le C_j \le k) \tag{26}$$
$$\mathbb{P}(C_j \mid \neg H_j) = \frac{s + \tilde{\kappa}_j[C_j]}{s \times (k+1) + \sum_{r=0}^{k} \tilde{\kappa}_j[r]} \qquad (1 \le j \le q,\ 0 \le C_j \le k).$$

Thereafter, by substituting Eq.(24) (prior probabilities) and Eq.(26) (likelihoods) into Eq.(23), the predicted label set in Eq.(22) naturally follows.

Y=ML-kNN($\mathcal{D}$, k, $\boldsymbol{x}$)
for $i = 1$ to m **do**
  Identify k nearest neighbors $\mathcal{N}(\boldsymbol{x}_i)$ for $\boldsymbol{x}_i$;
endfor
for $j = 1$ to q **do**
  Estimate the prior probabilities $\mathbb{P}(H_j)$ and $\mathbb{P}(\neg H_j)$ according to Eq.(24);
  Maintain frequency arrays κ_j and $\tilde{\kappa}_j$ according to Eq.(25);
endfor
Identify k nearest neighbors $\mathcal{N}(\boldsymbol{x})$ for $\boldsymbol{x}$;
for $j = 1$ to q **do**
  Calculate statistic C_j according to Eq.(21);
endfor
Return Y according to Eq.(22) (in conjunction with Eqs.(23), (24) and (26));

Fig. 7. Pseudo-code of ML-kNN.

Remarks: The pseudo-code of ML-kNN is summarized in Fig. 7. It is a *first-order* approach which reasons the relevance of each label separately. ML-kNN has the advantage of inheriting merits of both lazy learning and Bayesian reasoning: a) decision boundary can be adaptively adjusted due to the *varying neighbors* identified for each unseen instance; b) the class-imbalance issue can be largely mitigated due to the *prior probabilities* estimated for each class label. There are other ways to make use of lazy learning for handling multi-label data, such as combining kNN with ranking aggregation [7], [15], identifying kNN in a label-specific style [41], [101], expanding kNN to cover the whole training set [14], [49]. Considering that ML-kNN is ignorant of exploiting label correlations, several extensions have been proposed to provide patches to ML-kNN along this direction [13], [104]. As shown in Fig. 7, ML-kNN has computational complexity of $\mathcal{O}(m^2d+qmk)$ for training and $\mathcal{O}(md+qk)$ for testing.

3.3.2 Multi-Label Decision Tree (ML-DT)

The basic idea of this algorithm is to adopt *decision tree* techniques to deal with multi-label data, where an information gain criterion based on multi-label entropy is utilized to build the decision tree recursively [16].

Given any multi-label data set $\mathcal{T} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \le i \le n\}$ with n examples, the information gain achieved by dividing $\mathcal{T}$ along the l-th feature at splitting value ϑ is:

$$\begin{aligned}\mathrm{IG}(\mathcal{T}, l, \vartheta) &= \mathrm{MLEnt}(\mathcal{T}) \\ &\quad - \sum\nolimits_{\rho\in\{-,+\}} \frac{|\mathcal{T}^\rho|}{|\mathcal{T}|} \cdot \mathrm{MLEnt}(\mathcal{T}^\rho) \qquad (27)\\ \text{where}\quad \mathcal{T}^- &= \{(\boldsymbol{x}_i, Y_i) \mid x_{il} \le \vartheta, 1 \le i \le n\},\\ \mathcal{T}^+ &= \{(\boldsymbol{x}_i, Y_i) \mid x_{il} > \vartheta, 1 \le i \le n\}.\end{aligned}$$

Namely, $\mathcal{T}^-$ ($\mathcal{T}^+$) consists of examples with values on the l-th feature less (greater) than ϑ.[9]

Starting from the root node (i.e. $\mathcal{T} = \mathcal{D}$), ML-DT identifies the feature and the corresponding splitting value which maximizes the information gain in Eq.(27), and then generates two child nodes with respect to $\mathcal{T}^-$ and $\mathcal{T}^+$. The above process is invoked recursively by treating either $\mathcal{T}^-$ or $\mathcal{T}^+$ as the new root node, and terminates until some stopping criterion $\mathcal{C}$ is met (e.g. size of the child node is less than the pre-specified threshold).

To instantiate ML-DT, the mechanism for computing multi-label entropy, i.e. MLEnt($\cdot$) in Eq.(27), needs to be specified. A straightforward solution is to treat each subset $Y \subseteq \mathcal{Y}$ as a *new class* and then resort to the conventional single-label entropy:

$$\begin{aligned}\widehat{\mathrm{MLEnt}}(\mathcal{T}) &= -\sum\nolimits_{Y\subseteq\mathcal{Y}} \mathbb{P}(Y)\cdot\log_2(\mathbb{P}(Y)), \qquad (28)\\ \text{where}\quad \mathbb{P}(Y) &= \frac{\sum_{i=1}^{n}[\![Y_i = Y]\!]}{n}.\end{aligned}$$

However, as the number of new classes grows exponentially with respect to $|\mathcal{Y}|$, many of them might not even appear in $\mathcal{T}$ and thus only have trivial estimated probability (i.e. $\mathbb{P}(Y) = 0$). To circumvent this issue, ML-DT assumes *independence* among labels and computes the multi-label entropy in a decomposable way:

$$\begin{aligned}\mathrm{MLEnt}(\mathcal{T}) &= \sum\nolimits_{j=1}^{q} -p_j\log_2 p_j - (1-p_j)\log_2(1-p_j), \qquad (29)\\ \text{where}\quad p_j &= \frac{\sum_{i=1}^{n}[\![y_j \in Y_i]\!]}{n}.\end{aligned}$$

Here, p_j represents the fraction of examples in $\mathcal{T}$ with label y_j. Note that Eq.(29) can be regarded as a simplified version of Eq.(28) under the label independence assumption, and it holds that $\mathrm{MLEnt}(\mathcal{T}) \ge \widehat{\mathrm{MLEnt}}(\mathcal{T})$.

For unseen instance $\boldsymbol{x}$, it is fed to the learned decision tree by traversing along the paths until reaching a leaf node affiliated with a number of training examples $\mathcal{T} \subseteq \mathcal{D}$. Then, the predicted label set corresponds to:

$$Y = \{y_j \mid p_j > 0.5,\ 1 \le j \le q\}. \qquad (30)$$

In other words, if for one leaf node the majority of training examples falling into it have label y_j, any test instance allocated within the same leaf node will regard y_j as its relevant label.

Remarks: The pseudo-code of ML-DT is summarized in Fig. 8. It is a *first-order* approach which assumes label independence in calculating multi-label entropy. One prominent advantage of ML-DT lies in its high efficiency in inducing the decision tree model from multi-label data. Possible improvements on multi-label decision trees include employing pruning strategy [16] or ensemble learning techniques [52], [110]. As shown in Fig. 8, ML-DT has computational complexity of $\mathcal{O}(mdq)$ for training and $\mathcal{O}(mq)$ for testing.

3.3.3 Ranking Support Vector Machine (Rank-SVM)

The basic idea of this algorithm is to adapt *maximum margin* strategy to deal with multi-label data, where a set of linear classifiers are optimized to minimize the empirical ranking loss and enabled to handle nonlinear cases with kernel tricks [27].

Let the learning system be composed of q linear classifiers $\mathcal{W} = \{(\boldsymbol{w}_j, b_j) \mid 1 \le j \le q\}$, where $\boldsymbol{w}_j \in \mathbb{R}^d$ and $b_j \in \mathbb{R}$ are the *weight vector* and *bias* for the j-th class label y_j. Correspondingly, Rank-SVM defines the learning system's *margin* on $(\boldsymbol{x}_i, Y_i)$ by considering its ranking ability on the

9. Without loss of generality, here we assume that features are real-valued and the data set is bi-partitioned by setting splitting point along each feature. Similar to Eq.(27), information gain with respect to discrete-valued features can be defined as well.

Y=ML-DT($\mathcal{D}$, $\mathcal{C}$, $\boldsymbol{x}$)

1. Create a decision tree with root node $\mathcal{N}$ affiliated with the whole training set ($\mathcal{T} = \mathcal{D}$);
2. **if** stopping criterion $\mathcal{C}$ is met **then**
3. **break** and **go to** step 9;
4. **else**
5. Identify the feature-value pair (l, ϑ) which maximizes Eq.(27);
6. Set $\mathcal{T}^-$ and $\mathcal{T}^+$ according to Eq.(27);
7. Set $\mathcal{N}.lsubtree$ and $\mathcal{N}.rsubtree$ to the decision trees recursively constructed with $\mathcal{T}^-$ and $\mathcal{T}^+$ respectively;
8. **endif**
9. Traverse $\boldsymbol{x}$ along the decision tree from the root node until a leaf node is reached;
10. Return Y according to Eq.(30);

Fig. 8. Pseudo-code of ML-DT.

example's relevant and irrelevant labels:

$$\min_{(y_j,y_k)\in Y_i\times\bar{Y}_i} \frac{\langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i\rangle + b_j - b_k}{\|\boldsymbol{w}_j - \boldsymbol{w}_k\|}. \tag{31}$$

Here, $\langle \boldsymbol{u}, \boldsymbol{v}\rangle$ returns the inner product $\boldsymbol{u}^\top\boldsymbol{v}$. Geometrically speaking, for each relevant-irrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$, their discrimination boundary corresponds to the hyperplane $\langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}\rangle + b_j - b_k = 0$. Therefore, Eq.(31) considers the *signed* L_2-distance of $\boldsymbol{x}_i$ to hyperplanes of every relevant-irrelevant label pair, and then returns the minimum as the margin on $(\boldsymbol{x}_i, Y_i)$. Therefore, the learning system's margin on the whole training set $\mathcal{D}$ naturally follows:

$$\min_{(\boldsymbol{x}_i,Y_i)\in\mathcal{D}} \min_{(y_j,y_k)\in Y_i\times\bar{Y}_i} \frac{\langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i\rangle + b_j - b_k}{\|\boldsymbol{w}_j - \boldsymbol{w}_k\|}. \tag{32}$$

When the learning system is capable of properly ranking every relevant-irrelevant label pair for each training example, Eq.(32) will return *positive* margin. In this ideal case, we can rescale the linear classifiers to ensure: a) $\forall\ 1 \le i \le m$ and $(y_j, y_k) \in Y_i \times \bar{Y}_i$, $\langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i\rangle + b_j - b_k \ge 1$; b) $\exists\, i^* \in \{1, \ldots, m\}$ and $(y_{j^*}, y_{k^*}) \in Y_{i^*} \times \bar{Y}_{i^*}$, $\langle \boldsymbol{w}_{j^*} - \boldsymbol{w}_{k^*}, \boldsymbol{x}_{i^*}\rangle + b_{j^*} - b_{k^*} = 1$. Thereafter, the problem of maximizing the margin in Eq.(32) can be expressed as:

$$\begin{aligned} \max_{\mathcal{W}} \quad & \min_{(\boldsymbol{x}_i,Y_i)\in\mathcal{D}} \min_{(y_j,y_k)\in Y_i\times\bar{Y}_i} \frac{1}{\|\boldsymbol{w}_j - \boldsymbol{w}_k\|^2} \\ \text{subject to:} \quad & \langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i\rangle + b_j - b_k \ge 1 \\ & (1 \le i \le m,\ (y_j, y_k) \in Y_i \times \bar{Y}_i). \end{aligned} \tag{33}$$

Suppose we have sufficient training examples such that for each label pair (y_j, y_k) $(j \ne k)$, there exists $(\boldsymbol{x}, Y) \in \mathcal{D}$ satisfying $(y_j, y_k) \in Y \times \bar{Y}$. Thus, the objective in Eq.(33) becomes equivalent to $\max_{\mathcal{W}} \min_{1\le j<k\le q} \frac{1}{\|\boldsymbol{w}_j - \boldsymbol{w}_k\|^2}$ and the optimization problem can be re-written as:

$$\begin{aligned} \min_{\mathcal{W}} \quad & \max_{1\le j<k\le q} \|\boldsymbol{w}_j - \boldsymbol{w}_k\|^2 \\ \text{subject to:} \quad & \langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i\rangle + b_j - b_k \ge 1 \\ & (1 \le i \le m,\ (y_j, y_k) \in Y_i \times \bar{Y}_i). \end{aligned} \tag{34}$$

To overcome the difficulty brought by the max operator, Rank-SVM chooses to simplify Eq.(34) by approximating the max operator with the sum operator:

$$\begin{aligned} \min_{\mathcal{W}} \quad & \textstyle\sum_{j=1}^{q} \|\boldsymbol{w}_j\|^2 \\ \text{subject to:} \quad & \langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i\rangle + b_j - b_k \ge 1 \\ & (1 \le i \le m,\ (y_j, y_k) \in Y_i \times \bar{Y}_i). \end{aligned} \tag{35}$$

To accommodate real-world scenarios where constraints in Eq.(35) can not be fully satisfied, *slack variables* can be incorporated into Eq.(35):

$$\begin{aligned} \min_{\{\mathcal{W},\,\Xi\}} \quad & \textstyle\sum_{j=1}^{q} \|\boldsymbol{w}_j\|^2 + C\sum_{i=1}^{m} \frac{1}{|Y_i||\bar{Y}_i|} \sum_{(y_j,y_k)\in Y_i\times\bar{Y}_i} \xi_{ijk} \\ \text{subject to:} \quad & \langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i\rangle + b_j - b_k \ge 1 - \xi_{ijk} \\ & \xi_{ijk} \ge 0\ \ (1 \le i \le m,\ (y_j, y_k) \in Y_i \times \bar{Y}_i). \end{aligned} \tag{36}$$

Here, $\Xi = \{\xi_{ijk} \mid 1 \le i \le m,\ (y_j, y_k) \in Y_i \times \bar{Y}_i\}$ is the set of slack variables. The objective in Eq.(36) consists of two parts balanced by the trade-off parameter C. Specifically, the first part corresponds to the *margin* of the learning system, while the second parts corresponds to the surrogate *ranking loss* of the learning system implemented in hinge form. Note that surrogate ranking loss can be implemented in other ways such as the exponential form for neural network's global error function [107].

Note that Eq.(36) is a standard *quadratic programming* (QP) problem with convex objective and linear constraints, which can be tackled with any off-the-shelf QP solver. Furthermore, to endow Rank-SVM with nonlinear classification ability, one popular way is to solve Eq.(36) in its dual form via *kernel trick*. More details on the dual formulation can be found in [26].

As discussed in Subsection 2.1.3, Rank-SVM employs the stacking-style procedure to set the thresholding function $t(\cdot)$, i.e. $t(\boldsymbol{x}) = \langle \boldsymbol{w}^*, \boldsymbol{f}^*(\boldsymbol{x})\rangle + b^*$ with $\boldsymbol{f}^*(\boldsymbol{x}) = (f(\boldsymbol{x}, y_1), \ldots, f(\boldsymbol{x}, y_q))^{\mathrm{T}}$ and $f(\boldsymbol{x}, y_j) = \langle \boldsymbol{w}_j, \boldsymbol{x}\rangle + b_j$. For unseen instance $\boldsymbol{x}$, the predicted label set corresponds to:

$$Y = \{y_j \mid \langle \boldsymbol{w}_j, \boldsymbol{x}\rangle + b_j > \langle \boldsymbol{w}^*, \boldsymbol{f}^*(\boldsymbol{x})\rangle + b^*,\ 1 \le j \le q\}. \tag{37}$$

`Remarks`: The pseudo-code of Rank-SVM is summarized in Fig. 9. It is a *second-order* approach which defines the margin over hyperplanes for relevant-irrelevant label pairs. Rank-SVM benefits from kernels to handle nonlinear classification problems, and further variants can be achieved. Firstly, as shown in [37], the empirical

Y=Rank-SVM($\mathcal{D}$, C, $\boldsymbol{x}$)

1. Induce the classification system $\mathcal{W} = \{(\boldsymbol{w}_j, b_j) \mid 1 \le j \le q\}$ by solving the QP problem in Eq.(36);
2. Induce $(\boldsymbol{w}^*, b^*)$ for the thresholding function by solving the linear least square problem in Eq.(1);
3. Return Y according to Eq.(37);

Fig. 9. Pseudo-code of rank-SVM.

ranking loss considered in Eq.(36) can be replaced with other loss structures such as hamming loss, which can be cast as a general form of *structured output* classification [86], [87]. Secondly, the thresholding strategy can be accomplished with techniques other than stacking-style procedure [48]. Thirdly, to avoid the problem of kernel selection, multiple kernel learning techniques can be employed to learn from multi-label data [8], [46], [81]. As shown in Fig. 9, let $\mathcal{F}_{\mathrm{QP}}(a, b)$ represent the time complexity for a QP solver to solve Eq.(36) with a variables and b constraints, Rank-SVM has computational complexity of $\mathcal{O}(\mathcal{F}_{\mathrm{QP}}(dq + mq^2, mq^2) + q^2(q + m))$ for training and $\mathcal{O}(dq)$ for testing.

3.3.4 Collective Multi-Label Classifier (CML)

The basic idea of this algorithm is to adapt *maximum entropy* principle to deal with multi-label data, where correlations among labels are encoded as constraints that the resulting distribution must satisfy [33].

For any multi-label example $(\boldsymbol{x}, Y)$, let $(\boldsymbol{x}, \boldsymbol{y})$ be the corresponding *random variables* representation using binary label vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_q)^\top \in \{-1, +1\}^q$, whose j-th component indicates whether Y contains the j-th label ($y_j = +1$) or not ($y_j = -1$). Statistically speaking, the task of multi-label learning is equivalent to learn a joint probability distribution $p(\boldsymbol{x}, \boldsymbol{y})$.

Let $\mathcal{H}_p(\boldsymbol{x}, \boldsymbol{y})$ represent the *information entropy* of $(\boldsymbol{x}, \boldsymbol{y})$ given their distribution $p(\cdot, \cdot)$. The *principle of maximum entropy* [45] assumes that the distribution best modeling the current state of knowledge is the one maximizing $\mathcal{H}_p(\boldsymbol{x}, \boldsymbol{y})$ subject to a collection $\mathcal{K}$ of given facts:

$$\begin{aligned} \max_{p} \quad & \mathcal{H}_p(\boldsymbol{x}, \boldsymbol{y}) \\ \text{subject to:} \quad & \mathbb{E}_p[f_k(\boldsymbol{x}, \boldsymbol{y})] = F_k \quad (k \in \mathcal{K}). \end{aligned} \tag{38}$$

Generally, the fact is expressed as constraint on the *expectation* of some function over $(\boldsymbol{x}, \boldsymbol{y})$, i.e. by imposing $\mathbb{E}_p[f_k(\boldsymbol{x}, \boldsymbol{y})] = F_k$. Here, $\mathbb{E}_p[\,\cdot\,]$ is the expectation operator with respect to $p(\cdot, \cdot)$, while F_k corresponds to the expected value estimated from training set, e.g. $\frac{1}{m}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} f_k(\boldsymbol{x}, \boldsymbol{y})$.

Together with the normalization constraint on $p(\cdot, \cdot)$ (i.e. $\mathbb{E}_p[1] = 1$), the constrained optimization problem of Eq.(38) can be carried out with standard *Lagrange Multiplier* techniques. Accordingly, the optimal solution is shown to fall within the *Gibbs* distribution family [1]:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{Z_\Lambda(\boldsymbol{x})} \exp\left(\sum\nolimits_{k\in\mathcal{K}} \lambda_k \cdot f_k(\boldsymbol{x}, \boldsymbol{y})\right). \tag{39}$$

Here, $\Lambda = \{\lambda_k \mid k \in \mathcal{K}\}$ is the set of parameters to be determined, and $Z_\Lambda(\boldsymbol{x})$ is the *partition function* serving as the normalization factor, i.e. $Z_\Lambda(\boldsymbol{x}) = \sum_{\boldsymbol{y}} \exp\left(\sum_{k\in\mathcal{K}} \lambda_k \cdot f_k(\boldsymbol{x}, \boldsymbol{y})\right)$.

By assuming Gaussian prior (i.e. $\lambda_k \sim \mathcal{N}(0, \varepsilon^2)$), parameters in Λ can be found by maximizing the following *log-posterior* probability function:

$$\begin{aligned} l(\Lambda \mid \mathcal{D}) &= \log\left(\prod\nolimits_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} p(\boldsymbol{y} \mid \boldsymbol{x})\right) - \sum\nolimits_{k\in\mathcal{K}} \frac{\lambda_k^2}{2\varepsilon^2} \\ &= \sum\nolimits_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \left(\sum\nolimits_{k\in\mathcal{K}} \lambda_k \cdot f_k(\boldsymbol{x}, \boldsymbol{y}) - \log Z_\Lambda(\boldsymbol{x})\right) \\ &\quad - \sum\nolimits_{k\in\mathcal{K}} \frac{\lambda_k^2}{2\varepsilon^2}. \end{aligned} \tag{40}$$

Note that Eq.(40) is a convex function over Λ, whose global maximum (though not in closed-form) can be found by any off-the-shelf unconstrained optimization method such as BFGS [9]. Generally, gradients of $l(\Lambda \mid \mathcal{D})$ are needed by most numerical methods:

$$\begin{aligned} \frac{\partial l(\Lambda \mid \mathcal{D})}{\partial \lambda_k} &= \sum\nolimits_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \left(f_k(\boldsymbol{x}, \boldsymbol{y}) - \sum\nolimits_{\boldsymbol{y}} f_k(\boldsymbol{x}, \boldsymbol{y}) p(\boldsymbol{y} \mid \boldsymbol{x})\right) \\ &\quad - \frac{\lambda_k}{\varepsilon^2} \quad (k \in \mathcal{K}). \end{aligned} \tag{41}$$

For CML, the set of constraints consists of two parts $\mathcal{K} = \mathcal{K}_1 \bigcup \mathcal{K}_2$. Concretely, $\mathcal{K}_1 = \{(l, j) \mid 1 \le l \le d,\ 1 \le j \le q\}$ specifies a total of $d \cdot q$ constraints with $f_k(\boldsymbol{x}, \boldsymbol{y}) = x_l \cdot [\![y_j = 1]\!]$ ($k = (l, j) \in \mathcal{K}_1$). In addition, $\mathcal{K}_2 = \{(j_1, j_2, b_1, b_2) \mid 1 \le j_1 < j_2 \le q,\ b_1, b_2 \in \{-1, +1\}\}$ specifies a total of $4 \cdot \binom{q}{2}$ constraints with $f_k(\boldsymbol{x}, \boldsymbol{y}) = [\![y_{j_1} = b_1]\!] \cdot [\![y_{j_2} = b_2]\!]$ ($k = (j_1, j_2, b_1, b_2) \in \mathcal{K}_2$). Actually, constraints in $\mathcal{K}$ can be specified in other ways yielding variants of CML [33], [114].

For unseen instance $\boldsymbol{x}$, the predicted label set corresponds to:

$$Y = \arg\max_{\boldsymbol{y}} p(\boldsymbol{y} \mid \boldsymbol{x}). \tag{42}$$

Note that exact inference with arg max is only tractable for small label space. Otherwise, pruning strategies need to be applied to significantly reduce the search space of arg max, e.g. only considering label sets appearing in the training set [33].

`Remarks:` The pseudo-code of CML is summarized in Fig. 10. It is a *second-order* approach where correlations between every label pair are considered via constraints in $\mathcal{K}_2$. The second-order correlation studied by CML is more general than that of Rank-SVM [27] as the latter only considers relevant-irrelevant label pairs. As a conditional random field (CRF) model, CML is interested in using the *conditional* probability distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$ in Eq.(39) for classification. Interestingly, $p(\boldsymbol{y} \mid \boldsymbol{x})$ can be factored in various ways such as $p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{j=1}^{q} p(y_j \mid \boldsymbol{x}, y_1, \ldots, y_{j-1})$ where each term in the product can be modeled by one classifier in the classifier chain [20], [72], or $p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{j=1}^{q} p(y_j \mid \boldsymbol{x}, \mathbf{pa}_j)$ where each term in the product can be modeled by node y_j and its parents $\mathbf{pa}_j$ in a directed graph [36], [105], [106], and efficient algorithms exist when the directed graph corresponds to multi-dimensional Bayesian network

Y=CML($\mathcal{D}$, ε^2, $\boldsymbol{x}$)

1. **for** $l = 1$ to d **do**
2. **for** $j = 1$ to q **do**
3. Set constraint $f_k(\boldsymbol{x}, \boldsymbol{y}) = x_l \cdot [\![y_j = 1]\!]$ ($k = (l, j) \in \mathcal{K}_1$);
4. **endfor**
5. **endfor**
6. **for** $j_1 = 1$ to $q - 1$ **do**
7. **for** $j_2 = j_1 + 1$ to q **do**
8. Set constraint $f_k(\boldsymbol{x}, \boldsymbol{y}) = [\![y_{j_1} = b_1]\!] \cdot [\![y_{j_2} = b_2]\!]$ ($b_1, b_2 \in \{-1, +1\}$, $k = (j_1, j_2, b_1, b_2) \in \mathcal{K}_2$);
9. **endfor**
10. **endfor**
11. Determine parameters $\Lambda = \{\lambda_k \mid k \in \mathcal{K}_1 \bigcup \mathcal{K}_2\}$ by maximizing Eq.(40) (in conjunction with Eq.(41));
12. Return Y according to Eq.(42);

Fig. 10. Pseudo-code of CML.

TABLE 2
Summary of Representative Multi-Label Learning Algorithms Being Reviewed

Algorithm	Basic Idea	Order of Correlations	Complexity [Train/Test]	Tested Domains	Optimized Metric
Binary Relevance [5]	Fit multi-label data to q binary classifiers	first-order	$\mathcal{O}(q \cdot \mathcal{F}_{\mathcal{B}}(m, d))/\mathcal{O}(q \cdot \mathcal{F}'_{\mathcal{B}}(d))$	image	classification (hamming loss)
Classifier Chains [72]	Fit multi-label data to a chain of binary classifiers	high-order	$\mathcal{O}(q \cdot \mathcal{F}_{\mathcal{B}}(m, d+q))/\mathcal{O}(q \cdot \mathcal{F}'_{\mathcal{B}}(d+q))$	image, video text, biology	classification (hamming loss)
Calibrated Label Ranking [30]	Fit multi-label data to $\frac{q(q+1)}{2}$ binary classifiers	second-order	$\mathcal{O}(q^2 \cdot \mathcal{F}_{\mathcal{B}}(m, d))/\mathcal{O}(q^2 \cdot \mathcal{F}'_{\mathcal{B}}(d))$	image, text biology	Ranking (ranking loss)
Random k-Labelsets [94]	Fit multi-label data to n multi-class classifiers	high-order	$\mathcal{O}(n \cdot \mathcal{F}_{\mathcal{M}}(m, d, 2^k))/\mathcal{O}(n \cdot \mathcal{F}'_{\mathcal{M}}(d, 2^k))$	image, text biology	classification (subset accuracy)
ML-kNN [108]	Fit k-nearest neighbor to multi-label data	first-order	$\mathcal{O}(m^2 d + qmk)/\mathcal{O}(md + qk)$	image, text biology	classification (hamming loss)
ML-DT [16]	Fit decision tree to multi-label data	first-order	$\mathcal{O}(mdq)/\mathcal{O}(mq)$	biology	classification (hamming loss)
Rank-SVM [27]	Fit kernel learning to multi-label data	second-order	$\mathcal{O}(\mathcal{F}_{\mathrm{QP}}(dq + mq^2, mq^2) + q^2(q+m))/\mathcal{O}(dq)$	biology	Ranking (ranking loss)
CML [33]	Fit conditional random field to multi-label data	second-order	$\mathcal{O}(\mathcal{F}_{\mathrm{UNC}}(dq + q^2, m))/\mathcal{O}((dq + q^2) \cdot 2^q)$	text	classification (subset accuracy)

with restricted topology [4], [18], [98]. Directed graphs are also found to be useful for modeling *multiple fault diagnosis* where y_j indicates the good/failing condition of one of the device's components [3], [19]. On the other hand, there have been some multi-label generative models which aim to model the *joint* probability distribution $p(\boldsymbol{x}, \boldsymbol{y})$ [63], [80], [97]. As shown in Fig. 10, let $\mathcal{F}_{\mathrm{UNC}}(a, m)$ represent the time complexity for an unconstrained optimization method to solve Eq.(40) with a variables, CML has computational complexity of $\mathcal{O}(\mathcal{F}_{\mathrm{UNC}}(dq + q^2, m))$ for training and $\mathcal{O}((dq + q^2) \cdot 2^q)$ for testing.

3.4 Summary

Table 2 summarizes properties of the eight multi-label learning algorithms investigated in Subsections 3.2 and 3.3, including their basic idea, label correlations, computational complexity, tested domains, and optimized (surrogate) metric. As shown in Table 2, (surrogate) *hamming loss* and *ranking loss* are among the most popular metrics to be optimized and theoretical analyses on them [21]–[23], [32] have been discussed in Subsection 2.2.4. Furthermore, note that the *subset accuracy* optimized by Random k-Labelsets is only measured with respect to the k-labelset instead of the whole label space.

Domains reported in Table 2 correspond to data types on which the corresponding algorithm is shown to work well in the *original* literature. However, all those representative multi-label learning algorithms are *general-purpose* and can be applied to various data types. Nevertheless, the computational complexity of each learning algorithm

TABLE 3
Online Resources for Multi-Label Learning

Resource Type	Resource URL and Descriptions
Tutorial	*http://www.ecmlpkdd2009.net/program/tutorials/learning-from-multi-label-data/* (In conjunction with ECML PKDD 2009)
	http://cig.fi.upm.es/index.php/presentations?download=4 (In conjunction with TAMIDA 2010)
Workshops	*http://lpis.csd.auth.gr/workshops/mld09/* (**MLD'09**: in conjunction with ECML PKDD 2009)
	http://cse.seu.edu.cn/conf/MLD10/ (**MLD'10**: in conjunction with ICML/COLT 2010)
	http://cse.seu.edu.cn/conf/LAWS12/ (**LAWS'12**: in conjunction with ACML 2012)
Special Issue	*http://mlkd.csd.auth.gr/events/ml2010si.html* (**Machine Learning Journal** Special Issue on Learning from Multi-Label Data [96])
Software	*http://mulan.sourceforge.net/index.html* (The MULAN [93] open-source Java library)
	http://meka.sourceforge.net/ (The MEKA project based on WEKA [38])
	http://cse.seu.edu.cn/people/zhangml/Resources.htm#codes_mll (Matlab codes for multi-label learning)
Data Sets	*http://mulan.sourceforge.net/datasets.html* (Data sets from MULAN)
	http://meka.sourceforge.net/#datasets (Data sets from MEKA)
	http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html (Data sets from LIBSVM [11])

does play a key factor on its suitability for different scales of data. Here, the data *scalability* can be studied in terms of three main aspects including the number of training examples (i.e. m), the dimensionality (i.e. d), and the number of possible class labels (i.e. q). Furthermore, algorithms which append class labels as extra features to instance space [20], [72], [106] might not benefit too much from this strategy when instance dimensionality is much larger than the number of class labels (i.e. $d \gg q$).

As arguably the mostly-studied supervised learning framework, several algorithms in Table 2 employ *binary classification* as the intermediate step to learn from multi-label data [5], [30], [72]. An initial and general attempt towards binary classification transformation comes from the famous AdaBoost.MH algorithm [75], where each multi-label training example $(\boldsymbol{x}_i, Y_i)$ is converted into q binary examples $\{([\boldsymbol{x}_i, y_j], \phi(Y_i, y_j)) \mid 1 \leq j \leq q\}$. It can be regarded as a *high-order* approach where labels in $\mathcal{Y}$ are treated as appending feature to $\mathcal{X}$ and would be related to each other via the shared instance $\boldsymbol{x}$, as far as the binary learning algorithm $\mathcal{B}$ is capable of capturing dependencies among features. Other ways towards binary classification transformation can be fulfilled with techniques such as stacked aggregation [34], [64], [88] or Error-Correcting Output Codes (ECOC) [28], [111].

In addition, first-order algorithm adaptation methods can not be simply regarded as Binary Relevance [5] combined with specific binary learners. For example, ML-kNN [108] is more than Binary Relevance combined with kNN as Bayesian inference is employed to reason with neighboring information, and ML-DT [16] is more than Binary Relevance combined with decision tree as a single decision tree instead of q decision trees is built to accommodate all class labels (based on multi-label entropy).

4 Related Learning Settings

There are several learning settings related to multi-label learning which are worth some discussion, such as multi-instance learning [25], ordinal classification [29], multi-task learning [10], and data streams classification [31].

Multi-instance learning [25] studies the problem where each example is described by a bag of instances while associated with a single (binary) label. A bag is regarded to be positive iff at least one of its constituent instances is positive. In contrast to multi-label learning which models the object's ambiguities (complicated semantics) in output (label) space, multi-instance learning can be viewed as modeling the object's ambiguities in input (instance) space [113]. There are some initial attempt towards exploiting multi-instance representation for learning from multi-label data [109].

Ordinal classification [29] studies the problem where a natural ordering exists among all the class labels. In multi-label learning, we can accordingly assume an ordering of relevance on each class label to generalize the crisp membership ($y_j \in \{-1, +1\}$) into the graded membership ($y_j \in \{m_1, m_2, \ldots, m_k\}$ where $m_1 < m_2 < \cdots < m_k$). Therefore, graded multi-label learning accommodates the case where we can only provide vague (ordinal) instead of definite judgement on the label relevance. Existing work shows that graded multi-label learning can be solved by transforming it into a set of ordinal classification problems (one for each class label), or a set of standard multi-label learning problems (one for each membership level) [12].

Multi-task learning [10] studies the problem where multiple tasks are trained in parallel such that training information of related tasks are used as an inductive bias to help improve the generalization performance of other tasks. Nonetheless, there are some essential differences between multi-task learning and multi-label learning to be noticed. Firstly, in multi-label learning all the examples share the same feature space, while in multi-task learning the tasks can be in the same feature space or different feature spaces. Secondly, in multi-label learning the goal is to predict the label subset associated with an object, while the purpose of multi-task learning is to have multiple tasks to be learned well simultaneously, and it does not concern on which task subset should be associated with an object (if we take a label as a task) since it generally assumes that every object is involved by all tasks. Thirdly, in multi-label learning it is not rare (yet demanding) to deal with large label space [90], while in multi-task learning it is not reasonable to consider a large number of tasks. Nevertheless, techniques for multi-task learning might be used to benefit multi-label learning [56].

Data streams classification [31] studies the problem where real-world objects are generated online and processed in a real-time manner. Nowadays, streaming data with multi-label nature widely exist in real-world scenarios such as instant news, emails, microblogs, etc [70]. As a usual challenge for streaming data analysis, the key factor for effectively classifying multi-label data streams is how to deal with the concept drift problem. Existing works model concept drift by updating the classifiers significantly whenever a new batch of examples arrive [68], taking the fading assumption that the influence of past data gradually declines as time evolves [53], [78], or maintaining a change detector alerting whenever a concept drift is detected [70].

5 Conclusion

In this paper, the state-of-the-art of multi-label learning is reviewed in terms of paradigm formalization, learning algorithms and related learning settings. In particular, instead of trying to go through all the learning techniques within confined space, which would lead to only abridged introductions, we choose to elaborate the algorithmic details of eight representative multi-label learning algorithms with references to other related works. Some online resources for multi-label learning are summarized in Table 3, including academic activities (tutorial, workshops, special issue), publicly-available software and data sets.

As discussed in Section 2.1.2, although the idea of exploiting label correlations have been employed by various multi-label learning techniques, there has not been any formal characterization on the underlying concept or any principled mechanism on the appropriate usage of label correlations. Recent researches indicate that correlations among labels might be *asymmetric*, i.e. the influence of one label to the other one is not necessarily be the same in the inverse direction [42], or *local*, i.e. different instances share different label correlations with few correlations being globally applicable [43]. Nevertheless, full understanding on label correlations, especially for scenarios with large output space, would remain as the holy grail for multi-label learning.

As reviewed in Section 3, multi-label learning algorithms are introduced by focusing on their algorithmic properties. One natural complement to this review would be conducting thorough experimental studies to get insights on the pros and cons of different multi-label learning algorithms. A recent attempt towards extensive experimental comparison can be found in [62] where 12 multi-label learning algorithms are compared with respect to 16 evaluation metrics. Interestingly while not surprisingly, the best-performing algorithm for both classification and ranking metrics turns out to be the one based on *ensemble learning* techniques (i.e. random forest of predictive decision trees [52]). Nevertheless, empirical comparison across a broad range or within a focused type (e.g. [79]) are worthwhile topic to be further explored.

Acknowledgments

The authors would like to thank the associate editor and the anonymous reviewers for their insightful comments and suggestions. This research was supported by the National Science Foundation of China (61073097, 61175049, 61222309), the National Fundamental Research Program of China (2010CB327903), the MOE Program for New Century Excellent Talents in University (NCET-13-0130), and the Fundamental Research Funds for the Central Universities (the Cultivation Program for Young Faculties of Southeast University).

References

[1] A. L. Berger, V. J. Della Pietra, and S. A. Della Pietra, "A maximum entropy approach to natural language processing," *Comput. Linguist.*, vol. 22, no. 1, pp. 39–71, 1996.
[2] W. Bi and J. T. Kwok, "Multi-label classification on tree- and DAG-structured hierarchies," in *Proc. 28th Int. Conf. Mach. Learn.*, Bellevue, WA, USA, 2011, pp. 17–24.
[3] C. Bielza, G. Li, and P. Larrañaga, "Multi-dimensional classification with Bayesian networks," *Int. J. Approx. Reason.*, vol. 52, no. 6, pp. 705–727, 2011.
[4] H. Borchani, C. Bielza, C. Toro, and P. Larrañaga, "Predicting human immunodeficiency virus type 1 inhibitors using multi-dimensional Bayesian network classifiers," *Artif. Intell. Med.*, vol. 57, no. 3, pp. 219–229, Mar. 2013.
[5] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognit.*, vol. 37, no. 9, pp. 1757–1771, 2004.
[6] K. Brinker, J. Fürnkranz, and E. Hüllermeier, "A unified model for multilabel classification and ranking," in *Proc. 17th Eur. Conf. Artif. Intell.*, Riva del Garda, Italy, 2006, pp. 489–493.
[7] K. Brinker and E. Hüllermeier, "Case-based multilabel ranking," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, Hyderabad, India, 2007, pp. 702–707.
[8] S. S. Bucak, R. Jin, and A. K. Jain, "Multi-label multiple kernel learning by stochastic approximation: Application to visual object recognition," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Cambridge, MA, USA: MIT Press, 2010, pp. 325–333.
[9] R. H. Byrd, J. Nocedal, and R. B. Schnabel, "Representations of quasi-newton matrices and their use in limited memory methods," *Math. Program.*, vol. 63, no. 1–3, pp. 129–156, 1994.
[10] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[11] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, Article 27, 2011 [Online]. Available: http://www.csie.ntu.edu.tw/~{}cjlin/libsvm

[12] W. Cheng, K. Dembczyński, and E. Hüllermeier, "Graded multilabel classification: The ordinal case," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 223–230.

[13] W. Cheng and E. Hüllermeier, "Combining instance-based learning and logistic regression for multilabel classification," *Mach. Learn.*, vol. 76, no. 2–3, pp. 211–225, 2009.

[14] W. Cheng and E. Hüllermeier, "A simple instance-based approach to multilabel classification using the Mallows model," in *Proc. Work. Notes 1st Int. Workshop Learn. MLD*, Bled, Slovenia, 2009, pp. 28–38.

[15] T.-H. Chiang, H.-Y. Lo, and S.-D. Lin, "A ranking-based KNN approach for multi-label classification," in *Proc. 4th Asian Conf. Mach. Learn.*, Singapore, 2012, pp. 81–96.

[16] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *Lecture Notes in Computer Science 2168*, L. De Raedt and A. Siebes, Eds. Berlin, Germany: Springer, 2001, pp. 42–53.

[17] A. de Carvalho and A. A. Freitas, "A tutorial on multi-label classification techniques," in *Studies in Computational Intelligence 205*, A. Abraham, A. E. Hassanien, and V. Snásel, Eds. Berlin, Germany: Springer, 2009, pp. 177–195.

[18] P. R. de Waal and L. C. van der Gaag, "Inference and learning in multi-dimensional Bayesian network classifiers," in *Lecture Notes in Artificial Intelligence 4724*, K. Mellouli, Ed. Berlin, Germany: Springer, 2007, pp. 501–511.

[19] V. Delcroix, M.-A. Maalej, and S. Piechowiak, "Bayesian networks versus other probabilistic models for the multiple diagnosis of large devices," *Int. J. Artif. Intell. Tools*, vol. 16, no. 3, pp. 417–433, 2007.

[20] K. Dembczyński, W. Cheng, and E. Hüllermeier, "Bayes optimal multilabel classification via probabilistic classifier chains," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 279–286.

[21] K. Dembczyński, W. Kotłowski, and E. Hüllermeier, "Consistent multilabel ranking through univariate loss minization," in *Proc. 29th Int. Conf. Mach. Learn.*, Edinburgh, U.K., 2012, pp. 1319–1326.

[22] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, "Regret analysis for performance metrics in multi-label classification: The case of hamming loss and subset zero-one loss," in *Lecture Notes in Artificial Intelligence 6321*, J. Balcázar, F. Bonchi, A. Gionis, and M. Sebag, Eds. Berlin, Germany: Springer, 2010, pp. 280–295.

[23] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, "On label dependence and loss minimization in multi-label classification," *Mach. Learn.*, vol. 88, no. 1–2, pp. 5–45, 2012.

[24] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. 1st Int. Workshop MCS*, Cagliari, Italy, 2000, pp. 1–15.

[25] 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.

[26] A. Elisseeff and J. Weston, "Kernel methods for multi-labelled classification and categorical regression problems," Biowulf Technologies, Tech. Rep., 2001.

[27] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA, USA: MIT Press, 2002, pp. 681–687.

[28] C. S. Ferng and H. T. Lin, "Multi-label classification with error-correcting codes," in *Proc. 3rd Asian Conf. Mach. Learn.*, Taoyuan, Taiwan, 2011, pp. 281–295.

[29] E. Frank and M. Hall, "A simple approach to ordinal classification," in *Lecture Notes in Computer Science 2167*, L. De Raedt and P. Flach, Eds. Berlin, Germany: Springer, 2001, pp. 145–156.

[30] J. Fürnkranz, E. Hüllermeier, E. Loza Mencía, and K. Brinker, "Multilabel classification via calibrated label ranking," *Mach. Learn.*, vol. 73, no. 2, pp. 133–153, 2008.

[31] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "A survey of classification methods in data streams," in *Data Streams: Models and Algorithms*, C. C. Aggarwal, Ed. Berlin, Germany: Springer, 2007, pp. 39–59.

[32] W. Gao and Z.-H. Zhou, "On the consistency of multi-label learning," in *Proc. 24th Annu. Conf. Learn. Theory*, Budapest, Hungary, 2011, pp. 341–358.

[33] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Proc. 14th ACM Int. Conf. Inform. Knowl. Manage.*, Bremen, Germany, 2005, pp. 195–200.

[34] S. Godbole and S. Sarawagi, "Discriminative methods for multi-labeled classification," in *Lecture Notes in Artificial Intelligence 3056*, H. Dai, R. Srikant, and C. Zhang, Eds. Berlin, Germany: Springer, 2004, pp. 22–30.

[35] S. Gopal and Y. Yang, "Multilabel classification with meta-level features," in *Proc. 33rd SIGIR*, Geneva, Switzerland, 2010, pp. 315–322.

[36] Y. Guo and S. Gu, "Multi-label classification using conditional dependency networks," in *Proc. 22nd IJCAI*, Barcelona, Spain, 2011, pp. 1300–1305.

[37] Y. Guo and D. Schuurmans, "Adaptive large margin training for multilabel classification," in *Proc. 25th AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2011, pp. 374–379.

[38] M. Hall *et al.*, "The WEKA data mining software: An update," *SIGKDD Explor.*, vol. 11, no. 1, pp. 10–18, 2009.

[39] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[40] B. Hariharan, L. Zelnik-Manor, S. V. N. Vishwanathan, and M. Varma, "Large scale max-margin multi-label classification with priors," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 423–430.

[41] K.-W. Huang and Z. Li, "A multilabel text classification algorithm for labeling risk factors in SEC form 10-K," *ACM Trans. Manage. Inform. Syst.*, vol. 2, no. 3, Article 18, 2011.

[42] S.-J. Huang, Y. Yu, and Z.-H. Zhou, "Multi-label hypothesis reuse," in *Proc. 18th ACM SIGKDD Conf. KDD*, Beijing, China, 2012, pp. 525–533.

[43] S.-J. Huang and Z.-H. Zhou, "Multi-label learning by exploiting label correlations locally," in *Proc. 26th AAAI Conf. Artif. Intell.*, Toronto, Canada, 2012, pp. 949–955.

[44] M. Ioannou, G. Sakkas, G. Tsoumakas, and I. Vlahavas, "Obtaining bipartition from score vectors for multi-label classification," in *Proc. 22nd IEEE ICTAI*, Arras, France, 2010, pp. 409–416.

[45] E. T. Jaynes, "Information theory and statistical mechanics," *Phys. Rev.*, vol. 106, no. 4, pp. 620–630, 1957.

[46] S. Ji, L. Sun, R. Jin, and J. Ye, "Multi-label multiple kernel learning," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Cambridge, MA, USA: MIT Press, 2009, pp. 777–784.

[47] S. Ji, L. Tang, S. Yu, and J. Ye, "Extracting shared subspace for multi-label classification," in *Proc. 14th ACM SIGKDD Conf. KDD*, Las Vegas, NV, USA, 2008, pp. 381–389.

[48] A. Jiang, C. Wang, and Y. Zhu, "Calibrated rank-SVM for multi-label image categorization," in *Proc. IJCNN*, Hong Kong, China, 2008, pp. 1450–1455.

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

[50] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Multilabel text classification for automated tag suggestion," in *Proc. ECML/PKDD Discovery Challenge*, Antwerp, Belgium, 2008, pp. 75–83.

[51] H. Kazawa, T. Izumitani, H. Taira, and E. Maeda, "Maximal margin labeling for multi-topic text categorization," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA, USA: MIT Press, 2005, pp. 649–656.

[52] D. Kocev, C. Vens, J. Struyf, and S. Džeroski, "Ensembles of multi-objective decision trees," in *Proc. 18th ECML*, Warsaw, Poland, 2007, pp. 624–631.

[53] X. Kong and P. S. Yu, "An ensemble-based approach to fast classification of multi-label data streams," in *Proc. 7th Int. Conf. CollaborateCom*, Orlando, FL, USA, 2011, pp. 95–104.

[54] X. Kong and P. S. Yu, "gMLC: A multi-label feature selection framework for graph classification," *Knowl. Inform. Syst.*, vol. 31, no. 2, pp. 281–305, 2012.

[55] W. Kotłowski, K. Dembczyński, and E. Hüllermeier, "Bipartite ranking through minimization of univariate loss," in *Proc. 28th Int. Conf. Mach. Learn.*, Washington, DC, USA, 2011, pp. 1113–1120.

[56] E. Loza Mencía, "Multilabel classification in parallel tasks," in *Proc. Work. Notes 2nd Int. Workshop Learn. MLD*, Haifa, Israel, 2010, pp. 20–36.

[57] E. Loza Mencía and J. Fürnkranz, "Efficient pairwise multilabel classification for large-scale problems in the legal domain," in *Lecture Notes in Artificial Intelligence 5212*, W. Daelemans, B. Goethals, and K. Morik, Eds. Berlin, Germany: Springer, 2008, pp. 50–65.

[58] E. Loza Mencía and J. Fürnkranz, "Pairwise learning of multilabel classifications with perceptrons," in *Proc. IJCNN*, Hong Kong, China, 2008, pp. 2899–2906.

[59] E. Loza Mencía, S.-H. Park, and J. Fürnkranz, "Efficient voting prediction for pairwise multilabel classification," *Neurocomput.*, vol. 73, no. 7–9, pp. 1164–1176, 2010.

[60] G. Madjarov, D. Gjorgjevikj, and T. Delev, "Efficient two stage voting architecture for pairwise multi-label classification," in *Lecture Notes in Computer Science 6464*, J. Li, Ed. Berlin, Germany: Springer, 2011, pp. 164–173.

[61] G. Madjarov, D. Gjorgjevikj, and S. Džeroski, "Two stage architecture for multi-label learning," *Pattern Recognit.*, vol. 45, no. 3, pp. 1019–1034, 2012.

[62] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognit.*, vol. 45, no. 9, pp. 3084–3104, 2012.

[63] A. McCallum, "Multi-label text classification with a mixture model trained by EM," in *Proc. Work. Notes AAAI Workshop Text Learn.*, Orlando, FL, USA, 1999.

[64] E. Montañés, J. R. Quevedo, and J. J. del Coz, "Aggregating independent and dependent models to learn multi-label classifiers," in *Lecture Notes in Artificial Intelligence 6912*, D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, Eds. Berlin, Germany: Springer, 2011, pp. 484–500.

[65] J. Petterson and T. Caetano, "Reverse multi-label learning," in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Cambridge, MA, USA: MIT Press, 2010, pp. 1912–1920.

[66] J. Petterson and T. Caetano, "Submodular multi-label learning," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds. Cambridge, MA, USA: MIT Press, 2011, pp. 1512–1520.

[67] G.-J. Qi *et al.*, "Correlative multi-label video annotation," in *Proc. 15th ACM Int. Conf. Multimedia*, Augsburg, Germany, 2007, pp. 17–26.

[68] W. Qu, Y. Zhang, J. Zhu, and Q. Qiu, "Mining multi-label concept-drifting data streams using dynamic classifier ensemble," in *Lecture Notes in Artificial Intelligence 5828*, Z.-H. Zhou and T. Washio, Eds. Berlin, Germany: Springer, 2009, pp. 308–321.

[69] J. R. Quevedo, O. Luaces, and A. Bahamonde, "Multilabel classifiers with a probabilistic thresholding strategy," *Pattern Recognit.*, vol. 45, no. 2, pp. 876–883, 2012.

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

[71] J. Read, B. Pfahringer, and G. Holmes, "Multi-label classification using ensembles of pruned sets," in *Proc. 8th IEEE ICDM*, Pisa, Italy, 2008, pp. 995–1000.

[72] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," in *Lecture Notes in Artificial Intelligence 5782*, W. Buntine, M. Grobelnik, and J. Shawe-Taylor, Eds. Berlin, Germany: Springer, 2009, pp. 254–269.

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

[74] C. Sanden and J. Z. Zhang, "Enhancing multi-label music genre classification through ensemble techniques," in *Proc. 34th SIGIR*, Beijing, China, 2011, pp. 705–714.

[75] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Mach. Learn.*, vol. 39, no. 2/3, pp. 135–168, 2000.

[76] C. Shi, X. Kong, P. S. Yu, and B. Wang, "Multi-label ensemble learning," in *Lecture Notes in Artificial Intelligence 6913*, D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, Eds. Berlin, Germany: Springer, 2011, pp. 223–239.

[77] Y. Song, L. Zhang, and L. C. Giles, "A sparse Gaussian processes classification framework for fast tag suggestions," in *Proc. 17th ACM Conf. Inform. Knowl. Manage.*, Napa Valley, CA, USA, 2008, pp. 93–102.

[78] E. Spyromitros-Xioufis, M. Spiliopoulou, G. Tsoumakas, and I. Vlahavas, "Dealing with concept drift and class imbalance in multi-label stream classification," in *Proc. 22nd IJCAI*, Barcelona, Spain, 2011, pp. 1583–1588.

[79] E. Spyromitros-Xioufis, G. Tsoumakas, and I. Vlahavas, "An empirical study of lazy multilabel classification algorithms," in *Proc. 5th Hellenic Conf. Artif. Intell.*, Syros, Greece, 2008, pp. 401–406.

[80] A. P. Streich and J. M. Buhmann, "Classification of multi-labeled data: A generative approach," in *Lecture Notes in Artificial Intelligence 5212*, W. Daelemans, B. Goethals, and K. Morik, Eds. Berlin, Germany: Springer, 2008, pp. 390–405.

[81] L. Tang, J. Chen, and J. Ye, "On multiple kernel learning with multiple labels," in *Proc. 21st IJCAI*, Pasadena, TX, USA, 2009, pp. 1255–1266.

[82] L. Tang, S. Rajan, and V. K. Narayanan, "Large scale multi-label classification via metalabeler," in *Proc. 19th Int. Conf. WWW*, Madrid, Spain, 2009, pp. 211–220.

[83] L. Tenenboim-Chekina, L. Rokach, and B. Shapira, "Identification of label dependencies for multi-label classification," in *Proc. Work. Notes 2nd Int. Workshop Learn. MLD*, Haifa, Israel, 2010, pp. 53–60.

[84] F. A. Thabtah, P. Cowling, and Y. Peng, "MMAC: A new multi-class, multi-label associative classification approach," in *Proc. 4th IEEE ICDM*, Brighton, U.K., 2004, pp. 217–224.

[85] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotions," in *Proc. 9th Int. Conf. Music Inform. Retrieval*, Philadephia, PA, USA, 2008, pp. 325–330.

[86] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, AB, Canada, 2004.

[87] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, Sept. 2005.

[88] G. Tsoumakas *et al.*, "Correlation-based pruning of stacked binary relevance models for multi-label learning," in *Proc. Work. Notes 1st Int. Workshop Learn. MLD*, Bled, Slovenia, 2009, pp. 101–116.

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

[90] G. Tsoumakas and I. Katakis, "Effective and efficient multilabel classification in domains with large number of labels," in *Proc. Work. Notes ECML PKDD Workshop MMD*, Antwerp, Belgium, 2008.

[91] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Berlin, Germany: Springer, 2010, pp. 667–686.

[92] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multi-label classification," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 7, pp. 1079–1089, Jul. 2011.

[93] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "MULAN: A Java library for multi-label learning," *J. Mach. Learn. Res.*, vol. 12, pp. 2411–2414, Jul. 2011.

[94] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *Lecture Notes in Artificial Intelligence 4701*, J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, Eds. Berlin, Germany: Springer, 2007, pp. 406–417.

[95] G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou, "Tutorial on learning from multi-label data," in *ECML PKDD*, Bled, Slovenia, 2009 [Online]. Available: http://www.ecmlpkdd2009.net/wp-content/uploads/2009/08/learning-from-multi-label-data.pdf

[96] G. Tsoumakas, M.-L. Zhang, and Z.-H. Zhou, "Introduction to the special issue on learning from multi-label data," *Mach. Learn.*, vol. 88, no. 1–2, pp. 1–4, 2012.

[97] N. Ueda and K. Saito, "Parametric mixture models for multi-label text," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA, USA: MIT Press, 2003, pp. 721–728.
[98] L. C. van der Gaag and P. R. de Waal, "Multi-dimensional Bayesian network classifiers," in *Proc. 3rd Eur. Workshop Probab. Graph. Models*, Prague, Czech Republic, 2006, pp. 107–114.
[99] A. Veloso, W. Meira, Jr., M. Gonçalves, and M. Zaki, "Multi-label lazy associative classification," in *Lecture Notes in Artificial Intelligence 4702*, J. N. Kok, J. Koronacki, R. L. de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, Eds. Berlin, Germany: Springer, 2007, pp. 605–612.
[100] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Mach. Learn.*, vol. 73, no. 2, pp. 185–214, 2008.
[101] H. Wang, C. Ding, and H. Huang, "Multi-label classification: Inconsistency and class balanced *k*-nearest neighbor," in *Proc. 24th AAAI Conf. Artif. Intell.*, Atlanta, GA, USA, 2010, pp. 1264–1266.
[102] M. Wang, X. Zhou, and T.-S. Chua, "Automatic image annotation via local multi-label classification," in *Proc. 7th ACM Int. Conf. Image Video Retrieval*, Niagara Falls, ON, Canada, 2008, pp. 17–26.
[103] R. Yan, J. Tešić, and J. R. Smith, "Model-shared subspace boosting for multi-label classification," in *Proc. 13th ACM SIGKDD*, San Jose, CA, USA, 2007, pp. 834–843.
[104] Z. Younes, F. Abdallah, T. Denoeux, and H. Snoussi, "A dependent multilabel classification method derived from the *k*-nearest neighbor rule," *EURASIP J. Adv. Sig. Process.*, vol. 2011, Article 645964, Apr. 2011.
[105] J. H. Zaragoza, L. E. Sucar, E. F. Morales, C. Bielza, and P. Larrañaga, "Bayesian chain classifiers for multidimensional classification," in *Proc. 22nd IJCAI*, Barcelona, Spain, 2011, pp. 2192–2197.
[106] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proc. 16th ACM SIGKDD Int. Conf. KDD*, Washington, DC, USA, 2010, pp. 999–1007.
[107] 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, Oct. 2006.
[108] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, 2007.
[109] M.-L. Zhang and Z.-H. Zhou, "Multi-label learning by instance differentiation," in *Proc. 22nd AAAI Conf. Artif. Intell.*, Vancouver, BC, Canada, 2007, pp. 669–674.
[110] X. Zhang *et al.*, "Multi-label classification without the multi-label cost," in *Proc. 10th SIAM Int. Conf. Data Mining*, Columbus, OH, USA, 2010, pp. 778–789.
[111] Y. Zhang and J. Schneider, "Maximum margin output coding," in *Proc. 29th Int. Conf. Mach. Learn.*, Edinburgh, U.K., 2012, pp. 1575–1582.
[112] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: Chapman & Hall/CRC, 2012.
[113] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li, "Multi-instance multi-label learning," *Artif. Intell.*, vol. 176, no. 1, pp. 2291–2320, 2012.
[114] S. Zhu, X. Ji, W. Xu, and Y. Gong, "Multi-labelled classification using maximum entropy method," in *Proc. 28th ACM SIGIR*, Salvador, Brazil, 2005, pp. 274–281.

Min-Ling Zhang received the B.Sc., M.Sc., and the Ph.D. degrees in computer science from Nanjing University, China, in 2001, 2004, and 2007, respectively. He is currently an Associate Professor at the Southeast University, China. His current research interests include machine learning and data mining. He has won the Microsoft Fellowship Award in 2004 and the Excellent Doctoral Dissertation Award of Chinese Computer Federation in 2008. In recent years, Dr. Zhang has served as Senior PC or PC for various conferences, including IJCAI'13, SDM'13, ACML'12 (SPC), and AAAI'13/'12, KDD'11/'10, ECML PKDD'12/'11, ICML'10 (PC), etc. He is the Program Co-Chair of the LAWS'12 (in conjunction with ACML'12) workshop on learning with weak supervision, and the MLD'09 (in conjunction with ECML/PKDD'09) and MLD'10 (in conjunction with ICML/COLT'10) workshops on learning from multi-label data. He is also one of the Guest Editors for the *Machine Learning Journal's* special issue on learning from multi-label data. He is a member of the IEEE.

Zhi-Hua Zhou (S'00-M'01-SM'06-F'13) received the B.Sc., M.Sc., and the Ph.D. degrees in computer science from Nanjing University, China, in 1996, 1998, and 2000, respectively, all with the highest honors. He joined the Department of Computer Science & Technology at Nanjing University as an Assistant Professor in 2001, and is currently a Professor and a Director of the LAMDA Group. His current research interests include artificial intelligence, machine learning, data mining, pattern recognition, and multimedia information retrieval. In these areas, he has published more than 90 papers in leading international journals or conference proceedings, and holds 12 patents. He has won various awards/honors including the National Science & Technology Award for Young Scholars of China, the Fok Ying Tung Young Professorship 1st-Grade Award, the Microsoft Young Professorship Award and nine international journals/conferences paper awards and competition awards. He serves as an Associate Editor-in-Chief of the *Chinese Science Bulletin*, Associate Editor of the *ACM Transactions on Intelligent Systems and Technology*, and on the Editorial Boards of various other journals. He also served as an Associate Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING and *Knowledge and Information Systems*. He is the Founder and Steering Committee Chair of ACML, Steering Committee Member of PAKDD and PRICAI. He serves/ed as General Chair/Co-Chair of ACML'12 and ADMA'12, Program Chair/Co-Chair for PAKDD'07, PRICAI'08, ACML'09, and SDM'13, Workshop Chair of KDD'12, Tutorial Co-Chair of KDD'13, Program Vice-Chair or Area Chair of various conferences, and chaired various domestic conferences in China. He is the Chair of the Machine Learning Technical Committee of the Chinese Association of Artificial Intelligence, Chair of the Artificial Intelligence & Pattern Recognition Technical Committee of the China Computer Federation, Vice-Chair of the Data Mining Technical Committee of IEEE Computational Intelligence Society and the Chair of the IEEE Computer Society Nanjing Chapter. He is a Fellow of the IAPR, the IEEE, and the IET/IEE.