Multi-Label Learning

Zhi-Hua Zhou[a] and Min-Ling Zhang[b]
[a] Nanjing University, Nanjing, China
[b] Southeast University, Nanjing, China

Definition

Multi-label learning is an extension of the standard supervised learning setting. In contrast to standard supervised learning where one training example is associated with a single class label, in multi-label learning one training example is associated with *multiple* class labels simultaneously. The multi-label learner induces a function that is able to assign multiple proper labels (from a given label set) to unseen instances. Multi-label learning reduces to standard supervised learning by restricting the number of class labels per instance to one.

Motivation and Background

Most classification learning approaches treat the class values as disjoint—each object may belong only to a single class, such as *ON* or *OFF*. Some applications, however, have categories that are not mutually exclusive—a single object may belong to multiple classes [14]. For instance, in text categorization, a news document on presidential election can cover multiple topics such as *politics*, *economics*, *diplomacy* and *TV debate* [10]; in image classification, a natural scene image can contain multiple sceneries such as *sky*, *sea*, *boat* and *beach* [2]. Actually, multi-label objects are often encountered in many applications such as bioinformatics, multimedia content annotation, information retrieval and web mining [14].

The goal of multi-label learning is to induce a function that can predict a subset of labels for an unseen instance from a given label set. Research into this important problem emerged in early 2000 and significant research progress has followed [14].

Structure of Learning System

Let $\mathcal{X} = \mathcal{R}^d$ denote the d-dimensional instance space, and $\mathcal{Y} = \{y_1, y_2, \ldots, y_q\}$ denote the label space consisting of q class labels. Given the multi-label training set $\mathcal{D} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq m\}$, the task of multi-label learning is to learn a function $h : \mathcal{X} \mapsto 2^{\mathcal{Y}}$ mapping from the instance space to the *powerset* of

the label space. For each multi-label training example $(\boldsymbol{x}_i, Y_i)$, $\boldsymbol{x}_i \in \mathcal{X}$ is a d-dimensional feature vector and $Y_i \subseteq \mathcal{Y}$ is the set of class labels associated with $\boldsymbol{x}_i$. The learned function $h(\cdot)$ predicts the proper label set for any unseen instance $\boldsymbol{x}$ as $h(\boldsymbol{x}) \subseteq \mathcal{Y}$.

An alternative model to $h(\cdot)$ returned by most multi-label learning systems corresponds to a real-valued function $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$. Here, $f(\boldsymbol{x}, y)$ can be regarded as the predictive *confidence* of $y \in \mathcal{Y}$ being a proper label for $\boldsymbol{x}$. In other words, for the multi-label example $(\boldsymbol{x}, Y)$, the predictive output $f(\boldsymbol{x}, y')$ on *relevant* label $y' \in Y$ should be larger than the predictive output $f(\boldsymbol{x}, y'')$ on *irrelevant* label $y'' \notin Y$, i.e., $f(\boldsymbol{x}, y') > f(\boldsymbol{x}, y'')$. By referring to a threshold function $t : \mathcal{X} \mapsto \mathbb{R}$, $h(\cdot)$ can be derived from the real-valued function $f(\cdot, \cdot)$ by: $h(\boldsymbol{x}) = \{y \mid f(\boldsymbol{x}, y) > t(\boldsymbol{x}), y \in \mathcal{Y}\}$.

Evaluation Measures

In standard supervised learning, popular measures used to evaluate the learning performance include *accuracy*, *precision*, *recall*, etc. In multi-label learning, however, these single-label evaluation measures cannot be adopted directly due to the multi-label nature of the data. Therefore, a number of evaluation measures have been designed for multi-label learning. These measures can be roughly categorized into two groups, i.e., *example-based* measures and *label-based* measures [14]. Example-based measures evaluate the generalization performance of the learned multi-label predictor on each test example separately, and then return the mean value across the test set; label-based measures evaluate the generalization performance of the predictor on each class label separately, and then return the macro/micro-averaging value across all class labels.

Let $\mathcal{S} = \{(\boldsymbol{x}_i, Y_i) \mid 1 \leq i \leq p\}$ denote the multi-label test set, and $h(\cdot)$ (or equivalently $f(\cdot, \cdot)$) denote the learned multi-label predictor. Typical example-based measures include:

- *Subset Accuracy*: $\frac{1}{p}\sum_{i=1}^{p}[\![h(\boldsymbol{x}_i) = Y_i]\!]$. This measure evaluates the proportion of test examples whose predicted label set coincides with the ground-truth label set. Here, $[\![\pi]\!]$ returns 1 if predicate π holds, and 0 otherwise.
- *Hamming Loss*: $\frac{1}{p}\sum_{i=1}^{p}\frac{1}{q}|h(\boldsymbol{x}_i)\Delta Y_i|$. This measure evaluates the proportion of misclassified instance-label pairs, i.e., a relevant label is missed or an irrelevant label is predicted. Here, Δ stands for the symmetric difference between two sets and $|\cdot|$ measures the cardinality of a set.
- *One-error*: $\frac{1}{p}\sum_{i=1}^{p}[\![\arg\max_{y\in\mathcal{Y}} f(\boldsymbol{x}_i, y) \notin Y_i]\!]$. This measure evaluates the proportion of test examples whose top-1 predicted label fails to be a relevant label.
- *Coverage*: $\frac{1}{p}\sum_{i=1}^{p}\max_{y\in Y_i} rank_f(\boldsymbol{x}_i, y) - 1$. This measure evaluates the number of steps needed to move down the ranked label list so as to cover all relevant labels of the test example. Here, $rank_f(\boldsymbol{x}, y)$ returns the rank of class label y within label space $\mathcal{Y}$ according to the descending order specified by $f(\boldsymbol{x}, \cdot)$.

- *Ranking Loss*: $\frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i||\bar{Y}_i|}|\{(y',y'')|f(\boldsymbol{x},y') \leq f(\boldsymbol{x}_i,y''),\ (y',y'') \in Y_i \times \bar{Y}_i\}|$. This measure evaluates the proportion of incorrectly ordered label pairs, i.e., an irrelevant label yields larger output value than a relevant label. Here, $\bar{Y}_i$ is the complementary set of Y_i in $\mathcal{Y}$.

- *Average Precision*: $\frac{1}{p}\sum_{i=1}^{p}\frac{1}{|Y_i|}\sum_{y\in Y_i}\frac{|\{y'|rank_f(\boldsymbol{x}_i,y')\leq rank_f(\boldsymbol{x}_i,y),y'\in Y_i\}|}{rank_f(\boldsymbol{x}_i,y)}$. This measure evaluates the average proportion of labels ranked higher than a relevant label $y \in Y_i$ that are also relevant.

For *hamming loss*, *one-error*, *coverage* and *ranking loss*, the smaller the value, the better the generalization performance. For the other example-based measures, the larger the value, the better the performance.

For label-based measures, to characterize the binary classification performance of the predictor on each label $y_j \in \mathcal{Y}$, four basic quantities regarding the test examples are commonly used: TP_j (#true positive), FP_j (#false positive), TN_j (#true negative), and FN_j (#false negative). It is evident that most binary classification measures can be derived based on these quantities. Let $B(TP_j, FP_j, TN_j, FN_j)$ denote a certain binary classification measure, label-based multi-label measures can be defined in either of the following ways:

- *Macro-B*: $\frac{1}{q}\sum_{j=1}^{q} B(TP_j, FP_j, TN_j, FN_j)$. This multi-label measure is derived by assuming equal importance for each label.

- *Micro-B*: $B(\sum_{j=1}^{q} TP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} TN_j, \sum_{j=1}^{q} FN_j)$. This multi-label measure is derived by assuming equal importance for each example.

Among popular choices of $B \in \{Accuracy, Precision, Recall, F\}$, the larger the macro-/micro-B value, the better the performance.

Label Correlation

The major challenge of learning from multi-label data lies in the potentially tremendous-sized output space. Here, the number of possible *label sets* to be predicted grows exponentially as the number of class labels increases. For example, a label space with a moderate number of 20 class labels will lead to more than 1 million (i.e., 2^{20}) possible label sets. Thus, many label sets will rarely have examples appearing in the training set, leading to poor performance if they are learned separately.

To deal with the challenge of huge output space, a common practice is to exploit the *label correlation* to facilitate the learning process [14]. For instance, the probability of an image having label *Africa* would be high if we know it already has labels *grassland* and *lions*; A document is unlikely to be labeled as *recreation* if it is related to *legislation* and *police*. Actually, the fundamental issue distinguishing multi-label learning from traditional supervised learning lies in the fact that in multi-label learning it is crucial to exploit the label relations.

A widely-used strategy is to estimate the correlation among labels directly from the training examples based on the assumed correlation model. Based on

the *order of correlations* being modeled, the estimation techniques can be roughly categorized into three categories: (a) First-order techniques tackling multi-label learning task in a *label-by-label* style and thus ignoring the co-existence of other labels, such as decomposing the multi-label learning problem into a number of independent binary classification problems (one per label) [2, 13]; (b) Second-order techniques tackling multi-label learning task by considering *pairwise* correlations between labels, such as the ranking between relevant and irrelevant labels [4, 10]; (c) High-order techniques tackling multi-label learning task by considering high-order correlations among labels, such as assuming the correlations among all labels [8] or random subsets of labels [11].

Another strategy is to adopt domain knowledge about label relations as input to the multi-label learning algorithms. One conventional source of domain knowledge corresponds to the label hierarchies (or taxonomies) available in some applications such as text classification [9]. There is also a recent strategy which tries to discover and exploit label relations during the procedure of learning the multi-label predictors [14].

Learning Algorithms

To design learning algorithms for multi-label data, two complementary philosophies naturally arise. On one hand, *algorithm adaptation* methods work by fitting algorithms to data, i.e., adapting popular standard supervised learning algorithms to deal with multi-label data. On the other hand, *problem transformation* methods work by fitting data to algorithms, i.e., transforming multi-label data to accommodate other well-established learning frameworks. During the past decade, lots of algorithms have been developed following these philosophies [14]. This section briefly introduces four representative algorithms, including algorithm adaptation methods ML-KNN (multi-label k-nearest neighbor) [13] and RANK-SVM (ranking support vector machine) [4], as well as problem transformation methods CC (classifier chain) [8] and RAKEL (random k-labelsets) [11]. These algorithms are simply chosen to manifest the essentials of two key design philosophies, which by no means exclude the importance of other multi-label learning algorithms.

ML-KNN adapts the *k-nearest neighbor* technique to deal with multi-label data [13]. Specifically, the *maximum a posteriori* (MAP) rule is utilized to make prediction for unseen instance by reasoning with the labeling information from its neighbors. Given the multi-label training set $\mathcal{D}$ and unseen instance $\boldsymbol{x}$, let $\mathcal{N}(\boldsymbol{x})$ denote the set of k nearest neighbors of $\boldsymbol{x}$ identified in $\mathcal{D}$. Accordingly, the following statistics can be calculated based on the labeling information of the neighbors in $\mathcal{N}(\boldsymbol{x})$: $C_j = \sum_{(\boldsymbol{x}_i, Y_i) \in \mathcal{N}(\boldsymbol{x})} [\![y_j \in Y_i]\!]$. Namely, C_j records the number of neighbors which take the j-th class label y_j as their relevant label. Let $P(H_j \mid C_j)$ represent the posterior probability that the event of H_j (i.e., $\boldsymbol{x}$ has y_j as its relevant label) holds under the condition of C_j (i.e., $\boldsymbol{x}$ has exactly C_j neighbors with relevant label y_j). Similarly, let $P(\neg H_j \mid C_j)$ represent the posterior probability that H_j does not hold under the same condition. Based

on the MAP rule, the label set for $\boldsymbol{x}$ is predicted by

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

According to the Bayes rule, we have $P(H_j \mid C_j) \propto P(H_j) \cdot P(C_j \mid H_j)$ and $P(\neg H_j \mid C_j) \propto P(\neg H_j) \cdot P(C_j \mid \neg H_j)$. Therefore, it suffices to make prediction by estimating the prior probabilities $\{P(H_j),\ P(\neg H_j)\}$ and the likelihoods $\{P(C_j \mid H_j),\ P(C_j \mid \neg H_j)\}$. These probabilistic terms can be estimated from the training set via the *frequency counting* strategy [13]. In general, ML-KNN assumes label independence in its learning procedure and optimizes the evaluation measure of hamming loss (or equivalently macro-/micro-accuracy).

RANK-SVM adapts *large margin* methods to deal with multi-label data [4]. Specifically, a set of linear classifiers are optimized to minimize the empirical ranking loss. Given the learning system with q linear classifiers $\mathcal{W} = \{(\boldsymbol{w}_j, b_j) \mid 1 \le j \le q\}$, its margin over each multi-label training example $(\boldsymbol{x}_i, Y_i)$ corresponds to

$$\gamma_i = \min_{(y_j, y_k) \in Y_i \times \bar{Y}_i} \frac{\langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i \rangle + b_j - b_k}{||\boldsymbol{w}_j - \boldsymbol{w}_k||} \tag{2}$$

Here, $\langle \cdot, \cdot \rangle$ returns the inner product between two vectors. Conceptually, Eq.(2) considers the signed L_2-distance of $\boldsymbol{x}_i$ to the decision hyperplane of every relevant-irrelevant label pair (y_j, y_k): $\langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x} \rangle + b_j - b_k = 0$, and then returns the minimum as the margin on $(\boldsymbol{x}_i, Y_i)$. Accordingly, the margin of the learning system on the whole training set $\mathcal{D}$ is: $\min_{(\boldsymbol{x}_i, Y_i) \in \mathcal{D}} \gamma_i$. Under the ideal case that the learning system can properly rank every relevant-irrelevant label pair for each training example, the large margin optimization problem turns out to be

$$\begin{aligned} \min_{\mathcal{W}} \quad & \max_{1 \le j < k \le q} ||\boldsymbol{w}_j - \boldsymbol{w}_k||^2 \\ \text{s.t.}: \quad & \langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i \rangle + b_j - b_k \ge 1 \\ & 1 \le i \le m,\ (y_j, y_k) \in Y_i \times \bar{Y}_i \end{aligned} \tag{3}$$

By approximating `max` by `sum` and introducing slack variables to accommodate violated constraints, Eq.(3) can be re-formulated as

$$\begin{aligned} \min_{\{\mathcal{W}, \Xi\}} \quad & \sum_{j=1}^{q} ||\boldsymbol{w}_j||^2 + C \sum_{i=1}^{m} \frac{1}{|Y_i||\bar{Y}_i|} \sum_{(y_j, y_k) \in Y_i \times \bar{Y}_i} \xi_{ijk} \\ \text{s.t.}: \quad & \langle \boldsymbol{w}_j - \boldsymbol{w}_k, \boldsymbol{x}_i \rangle + b_j - b_k \ge 1 - \xi_{ijk} \\ & \xi_{ijk} \ge 0,\ 1 \le i \le m,\ (y_j, y_k) \in Y_i \times \bar{Y}_i \end{aligned} \tag{4}$$

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 first objective term in Eq.(4) corresponds to the margin of the learning system, whereas the second objective term corresponds to the empirical ranking loss. The solution to Eq.(4) can be found by invoking standard *quadratic programming* (QP) procedure in its primal form, or incorporating kernel trick

in its dual form. The label set for unseen instance is predicted by thresholding the output of each classifier in $\mathcal{W}$. In general, RANK-SVM assumes second-order label correlations (relevant-irrelevant label pair) in its learning procedure and optimizes the evaluation measure of ranking loss.

CC transforms the multi-label learning problem into a chain of *binary* classification problems. Specifically, subsequent classifiers in the chain are built upon the predictions of preceding ones. Without loss of generality, suppose all the class labels in $\mathcal{Y}$ are ordered in a chain: $y_1 \succ y_2 \succ \cdots \succ y_q$. For the j-th class label y_j in the chain, a corresponding binary training set can be constructed by taking the relevancy of each preceding label as an extra feature to the instance:

$$\begin{aligned} &\mathcal{D}_j = \left\{ \left([\boldsymbol{x}_i, \mathbf{pre}_j^i], \phi(Y_i, y_j) \right) \mid 1 \leq i \leq m \right\} \\ &\text{where } \ \mathbf{pre}_j^i = (\phi(Y_i, y_1), \ldots, \phi(Y_i, y_{j-1}))^{\mathrm{T}} \end{aligned} \tag{5}$$

Here, $\phi(Y, y) = [\![y \in Y]\!]$ represents the binary assignment of class label y w.r.t. label set Y. As shown in Eq.(5), each instance $\boldsymbol{x}_i$ is appended with an extra feature vector $\mathbf{pre}_j^i$ representing the relevancy of those labels preceding y_j. After that, a binary classifier $g_j : \mathcal{X} \times \{0,1\}^{j-1} \mapsto \{0,1\}$ can be induced for y_j by utilizing some binary learning algorithm $\mathcal{B}$, i.e., $g_j \leftarrowtail \mathcal{B}(D_j)$. For unseen instance $\boldsymbol{x}$, its label set is predicted by traversing the classifier chain iteratively. The predicted binary assignment of y_j on $\boldsymbol{x}$, denoted as $\lambda_j^{\boldsymbol{x}}$, are recursively determined by

$$\begin{aligned} \lambda_1^{\boldsymbol{x}} &= g_1(\boldsymbol{x}) \\ \lambda_j^{\boldsymbol{x}} &= g_j([\boldsymbol{x}, \lambda_1^{\boldsymbol{x}}, \ldots, \lambda_{j-1}^{\boldsymbol{x}}]) \ \ (2 \leq j \leq q) \end{aligned} \tag{6}$$

Therefore, the predicted label set corresponds to: $Y = \{y_j \mid \lambda_j^{\boldsymbol{x}} = 1, \ 1 \leq j \leq q\}$. Evidently, the chaining order over the class labels has significant influence on the effectiveness of CC. To account for the effect of chaining order, an *ensemble* of classifier chains can be built with diverse random chaining orders. In general, CC assumes high-order label correlations (among all labels) in its learning procedure and optimizes the evaluation measure of hamming loss (or equivalently macro-/micro-accuracy).

RAKEL transforms the multi-label learning problem into an ensemble of *multi-class* classification problems. Specifically, each component learner in the ensemble is generated by considering a random subset of $\mathcal{Y}$. Let $\mathcal{S}_k \subset \mathcal{Y}$ denote a k-labelset which contains k random class labels in $\mathcal{Y}$. Accordingly, let $\sigma_{\mathcal{S}_k} : 2^{\mathcal{S}_k} \mapsto \mathcal{N}$ denote the injective function mapping from the power set of $\mathcal{S}_k$ to natural numbers. In view of $\mathcal{S}_k$, a corresponding multi-class training set can be constructed by shrinking the original label space $\mathcal{Y}$ into $\mathcal{S}_k$:

$$\mathcal{D}_{\mathcal{S}_k} = \{(\boldsymbol{x}_i, \sigma_{\mathcal{S}_k}(Y_i \cap \mathcal{S}_k)) \mid 1 \leq i \leq m\} \tag{7}$$

Here, the set of newly-transformed labels in $\mathcal{D}_{\mathcal{S}_k}$ corresponds to $\Gamma_{\mathcal{S}_k} = \{\sigma_{\mathcal{S}_k}(Y_i \cap \mathcal{S}_k) \mid 1 \leq i \leq m\}$. As shown in Eq.(7), each instance $\boldsymbol{x}_i$ is transformed into a multi-class single-label example by mapping the intersection between Y_i and

$\mathcal{S}_k$ into a new label in $\Gamma_{\mathcal{S}_k}$. After that, a multi-class classifier $g_{\mathcal{S}_k} : \mathcal{X} \mapsto \Gamma_{\mathcal{S}_k}$ can be induced for $\mathcal{S}_k$ by utilizing some multi-class learning algorithm $\mathcal{M}$, i.e., $g_{\mathcal{S}_k} \leftarrow\!\shortmid \mathcal{M}(\mathcal{D}_{\mathcal{S}_k})$. To entirely explore the original label space $\mathcal{Y}$ with k-labelsets, an ensemble of n random k-labelsets $\mathcal{S}_k^{(r)}$ $(1 \leq r \leq n)$ can be created where each of them leads to a multi-class classifier $g_{\mathcal{S}_k^{(r)}}(\cdot)$. For unseen instance $\boldsymbol{x}$, its label set is predicted by referring to the following two quantities:

$$\begin{aligned} \tau(\boldsymbol{x}, y_j) &= \sum_{r=1}^{n} \left[\!\!\left[y_j \in \mathcal{S}_k^{(r)} \right]\!\!\right] \\ \mu(\boldsymbol{x}, y_j) &= \sum_{r=1}^{n} \left[\!\!\left[y_j \in \sigma_{\mathcal{S}_k^{(r)}}^{-1}\left(g_{\mathcal{S}_k^{(r)}}(\boldsymbol{x}) \right) \right]\!\!\right] \end{aligned} \tag{8}$$

Conceptually, $\tau(\boldsymbol{x}, y_j)$ counts the *maximum* number of votes that y_j can receive from the ensemble, whereas $\mu(\boldsymbol{x}, y_j)$ counts the *actual* number of votes that y_j does receive from the ensemble. Therefore, the predicted label set corresponds to: $Y = \{y_j \mid \mu(\boldsymbol{x}, y_j)/\tau(\boldsymbol{x}, y_j) > 0.5,\ 1 \leq j \leq q\}$. In general, CC assumes high-order label correlations (among subsets of labels) in its learning procedure and optimizes the evaluation measure of subset accuracy (measured w.r.t. k-labelset).

It is worth mentioning that many multi-label learning algorithms mainly work under the scenarios where the label space $\mathcal{Y}$ contains moderate number (tens or hundreds) of class labels. Nonetheless, in many applications the number of class labels in $\mathcal{Y}$ can be huge. For instance, a web page may be annotated with relevant labels from the pool of more than one million Wikipedia categories. In such case, the computational complexity of many multi-label learning algorithms might be prohibitively high. Even for binary decomposition, which is the simplest way to learn from multi-label data, building one independent classifier for each label is still too computational demanding given the huge number of class labels. Therefore, specific strategies need to be employed to handle huge number of labels. One feasible strategy is to find a low-dimensional embedding of the original label space by exploiting the sparsity of relevant labels, where the classification model is built within the embedded label space [12]. Another strategy is to partition the original label space into different clusters based on tree structure, where the classification model is built within each leaf node [1].

Theory

Multi-label loss functions are usually *non-convex* and *discontinuous*, making them difficult to optimize directly. Therefore, in practice most learning algorithms resort to optimizing (convex) *surrogate* loss functions. There are several theoretical studies about the *consistency* of surrogate loss functions, i.e., whether the expected risk of surrogate loss of a learner converges to the Bayes risk of multi-label loss as the training set size increases. Recently, a necessary and sufficient condition has been provided for the consistency of multi-label learning based on surrogate loss functions [5].

For *hamming loss*, the state-of-the-art multi-label learning approaches are proven to be inconsistent [5]. for *ranking loss*, it has been shown that none pairwise convex surrogate loss defined on label pairs can be consistent; therefore, the *partial ranking loss* is introduced for multi-label learning, and some pairwise consistent surrogate loss function are provided [5]. The univariate convex surrogate loss defined on single label can be consistent with partial ranking loss based on a reduction to the bipartite ranking problem [3] although the reduction relaxes the original target.

Extensions

Multi-instance multi-label learning (MIML) [15] tries to induce a function $h_{\mathrm{MIML}} : 2^{\mathcal{X}} \mapsto 2^{\mathcal{Y}}$ from a training set $\{(X_i, Y_i) \mid 1 \leq i \leq m\}$, where $X_i \subseteq \mathcal{X}$ is a set of instances and Y_i is the set of class labels associated with X_i. The major difference between MIML and multi-label learning lies in the fact that each example in MIML is represented by a set of instances rather than a single instance. This framework is suitable to tasks involving complicated objects with inherent structures; e.g., a text document can be represented by a set of instances each corresponds to a section or paragraph. In addition to exploit the structural information for learning the predictor, MIML also offers the possibility of discovering the relation between semantic meanings and input patterns; e.g., it is possible to discover that the document owes a specific tag because of its several special paragraphs.

Superset label learning (SLL) [7] tries to induce a function $h_{SLL} : \mathcal{X} \mapsto \mathcal{Y}$ from a training set $\{(\boldsymbol{x}_i, S_i) \mid 1 \leq i \leq m)\}$, where $\boldsymbol{x}_i \in \mathcal{X}$ is an instance and $S_i \subseteq \mathcal{Y}$ is the set of candidate labels associated with $\boldsymbol{x}_i$ such that the (unknown) ground-truth label y_i belongs to S_i. The major difference between SLL and multi-label learning lies in the fact that each example in SLL is associated with multiple *candidate* labels among which only one label is valid. This framework is suitable to tasks where superset labeling information is readily available; e.g., a face in an image can be associated with all the names mentioned in the image's surrounding texts where only one name is valid.

Label distribution learning (LDL) [6] tries to induce a function $f_{\mathrm{LDL}} : \mathcal{X} \mapsto \mathcal{P}(\mathcal{Y})$ from a training set $\{(\boldsymbol{x}_i, \mathcal{D}_i) \mid 1 \leq i \leq m\}$, where $\boldsymbol{x}_i$ is an instance and $\mathcal{D}_i = \{d_i^1, d_i^2, \cdots, d_i^q\}$ is the probability mass of the q labels associated with $\boldsymbol{x}_i$ such that $d_i^j \geq 0$ $(1 \leq j \leq q)$ and $\sum_{j=1}^{q} d_i^j = 1$. The major difference between LDL and multi-label learning lies in the fact that the associated labeling information for each example in LDL is real-valued probability mass rather than discrete-valued binary labels. This framework is suitable to tasks where the degree of labeling importance is inherently different; e.g., entities appearing in a natural scene have different importance in implying its scenic concepts.

Future Challenges

There are many research challenges to be addressed in the future. Firstly, label relations play a critical role in multi-label learning; however, there lacks principled mechanism for label relation exploitation. Secondly, it is generally difficult to get accurate and complete label annotations, particularly when each example has many labels. Thus, it is important to develop multi-label learning approaches that can learn from partially labeled data. Moreover, multi-label data usually suffers from inherent class imbalance and unequal misclassification costs; taking these properties into full consideration is desirable.

References

[1] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 13–24, Rio de Janeiro, Brazil, 2013.

[2] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004.

[3] K. Dembczyński, W. Kotłowski, and E. Hüllermeier. Consistent multilabel ranking through univariate loss minimization. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1319–1326, Edinburgh, UK, 2012.

[4] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press, Cambridge, MA, 2002.

[5] W. Gao and Z.-H. Zhou. On the consistency of multi-label learning. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 341–358, Budapest, Hungary, 2011.

[6] X. Geng, C. Yin, and Z.-H. Zhou. Facial age estimation by label distribution learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(10):2401–2412, 2013.

[7] L. Liu and T. Dietterich. A conditional multinomial mixture model for superset label learning. In P. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 557–565. MIT Press, Cambridge, MA, 2012.

[8] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

[9] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Learning hierarchical multi-category text classification models. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 774–751, Bonn, Germany, 2005.

[10] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[11] G. Tsoumakas, I. Katakis, and I. Vlahavas. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089, 2011.

[12] J. Weston, S. Bengio, and N. Usunier. WSABIE: Scaling up to large vocabulary image annotation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 2764–2770, Barcelona, Spain, 2011.

[13] M.-L. Zhang and Z.-H. Zhou. ML-kNN: A lazy learning approach to multilabel learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[14] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

[15] Z.-H. Zhou, M.-L. Zhang, S.-J. Huang, and Y.-F. Li. Multi-instance multilabel learning. *Artificial Intelligence*, 176(1):2291–2320, 2012.