Multi-Label Co-Training*

Yuying Xing[1], Guoxian Yu[1,*], Carlotta Domeniconi[2], Jun Wang[1] and Zili Zhang[1,3]
[1]College of Computer and Information Science, Southwest University, Chongqing 400715, China
[2]Department of Computer Science, George Mason University, Fairfax 22030, USA
[3]School of Information Technology, Deakin University, Geelong, VIC 3220, Australia
{yyxing4148, gxyu, kingjun, zhangzl}@swu.edu.cn, carlotta@cs.gmu.edu

Abstract

Multi-label learning aims at assigning a set of appropriate labels to multi-label samples. Although it has been successfully applied in various domains in recent years, most multi-label learning methods require sufficient labeled training samples, because of the large number of possible label sets. Co-training, as an important branch of semi-supervised learning, can leverage unlabeled samples, along with scarce labeled ones, and can potentially help with the large labeled data requirement. However, it is a difficult challenge to combine multi-label learning with co-training. Two distinct issues are associated with the challenge: (i) how to solve the widely-witnessed *class-imbalance* problem in multi-label learning; and (ii) how to select samples with *confidence*, and communicate their predicted labels among classifiers for model refinement. To address these issues, we introduce an approach called Multi-Label Co-Training (MLCT). MLCT leverages information concerning the co-occurrence of pairwise labels to address the class-imbalance challenge; it introduces a predictive reliability measure to select samples, and applies label-wise filtering to confidently communicate labels of selected samples among co-training classifiers. MLCT performs favorably against related competitive multi-label learning methods on benchmark datasets and it is also robust to the input parameters.

1 Introduction

In multi-label learning, each sample is associated with several related class labels [Zhang and Zhou, 2014; Gibaja and Ventura, 2015]. Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ be the data matrix including n d-dimensional samples, and $\mathbf{Y} \in \mathbb{R}^{n\times q}$ be the q-dimensional label space for the samples. Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i))|1 \leq i \leq n\}$, the task of multi-label learning is to learn a predictive function $f(\mathbf{x}) \in \mathbb{R}^q$ that maps the input feature space of the samples onto the label space.

Most multi-label learning methods train the predictor using only labeled samples [Zhang and Zhou, 2007; Bucak *et al.*, 2011; Huang and Zhou, 2012; Yu *et al.*, 2014]. Given the exponential size of the *power set* of the labels, a very large number of labeled training samples is generally required. In practice, collecting sufficient labeled samples in this scenario is very expensive, and often impractical. On the other hand, with the rapid advancement of data collection and storage techniques, it has become feasible to collect a large number of unlabeled samples. Inspired by single-label semi-supervised learning [Zhu, 2008], efforts have been made to leverage both labeled and unlabeled samples for multi-label learning, giving promising results [Qian and Davidson, 2010; Yu *et al.*, 2012; Kong *et al.*, 2013; Wu *et al.*, 2015; Zhang and Yeung, 2009; Zhang *et al.*, 2015b]. The nature of this work, though, is typically *transductive*; that is, the canonical approach is to construct a graph that captures the connections between labeled and unlabeled instances, and then to predict the labels of the unlabeled instances embodied in the graph. As such, this approach cannot generalize to unseen samples. It is obviously desirable to empower the learner with an *inductive* ability that enables label prediction for new instances unseen during training.

Few attempts have been made to achieve inductive semi-supervised multi-label learning [Guo and Schuurmans, 2012; Wu and Zhang, 2013; Gönen, 2014; Tan *et al.*, 2017; Zhan and Zhang, 2017]. In [Guo and Schuurmans, 2012], the authors utilized unlabeled and labeled instances to learn a subspace representation, while simultaneously training a supervised large-margin multi-label classifier on the labeled instances, which could be directly applied to unseen instances. In [Wu and Zhang, 2013], the authors took advantage of label correlations in labeled instances and of maximum-margin regularization over unlabeled instances to optimize a collection of linear predictors for inductive multi-label classification. In [Gönen, 2014], the author proposed a Bayesian semi-supervised multilabel learning (BSSML) approach that combines linear dimensionality reduction with linear binary classification under a low-density assumption. In [Tan *et al.*, 2017], the authors introduced SMILE, which first estimates the missing labels of labeled samples and uses a graph to embody both labeled and unlabeled samples; it then trains a graph-regularized semi-supervised linear classifier, to further recover the missing labels of labeled samples, and to directly predict labels of unseen new samples. In [Zhan and Zhang, 2017], the authors introduced an inductive semi-supervised

*Guoxian Yu is the corresponding author.

multi-label learning using a co-training approach called COINs. Specifically, to enable single view co-training, COINs first optimizes two disjoint feature views from the whole feature space by maximizing the diversity between two classifiers independently trained on the two views [Chen *et al.*, 2011], and then iteratively communicates the pairwise ranking predictions of either classifier on unlabeled instances for model refinement. COINs communicates only the single predicted most relevant label and the single predicted most irrelevant label between two classifiers. However, multi-label instances are often simultaneously associated with several relevant and irrelevant labels, and not just a single one. For this reason, only communicating the two most relevant and irrelevant labels may mislead the refinement process, and may not achieve pronounced performance improvement. Furthermore, COINs only focuses on two views.

To fully accomplish inductive multi-label classification and to leverage labeled and unlabeled instances of multiple feature views, we advocate the integration of multi-label learning with the well-established co-training paradigm [Blum and Mitchell, 1998; Zhou and Li, 2005]. Co-training has a natural inductive classification ability. It mutually communicates the labels predicted with most confidence among classifiers, which are independently trained on the respective views of data, thus augmenting the labeled training sets. The classifiers are then independently retrained on the respective augmented training sets; the communication and update iterate till convergence. Nevertheless, it is a *difficult* challenge to integrate multi-label learning with co-training. Two distinct issues should be addressed:

(i) How to solve the widely-witnessed *class-imbalance* problem in multi-label learning. For multi-label datasets, the number of samples relevant to a label is generally much smaller than the number of samples irrelevant to that label. Furthermore, the number of relevant samples for different labels can vary significantly [Zhang *et al.*, 2015a; Sun and Lee, 2017]. The class-imbalance problem can be exaggerated when communicating labels among learners during the iterative process of co-training.

(ii) How to select the samples and communicate their predicted labels with *confidence* among multiple co-training classifiers. Unlike traditional co-training, the to-be-communicated samples can be associated with several labels, and not just one.

To address the above two issues in multi-label co-training, we propose a co-training based multi-label classification method called MLCT. MLCT first independently trains predictors on different views and makes prediction on unlabeled samples. Then, it uses the co-occurrence information of labels to adjust the predicted likelihoods and to deal with the class-imbalance problem. Next, it summarizes the adjusted likelihoods across views and measures the predictive confidence of samples based on the summarized likelihoods. After this, it selects samples with the highest confidence, applies label-wise filtering on the summarized likelihoods of the selected samples, and then communicates filtered labels among learners during the iterative co-training process. MLCT repeats the above iterative process till convergence and makes the final prediction on unseen samples by combining the predictions of the classifiers via a majority vote. An extensive comparative study shows that MLCT performs favorably against the recently proposed COINs [Zhan and Zhang, 2017] and other representative multi-label learning methods (including ML-KNN [Zhang and Zhou, 2007], MLRGL[Bucak *et al.*, 2011], MLLOC[Huang and Zhou, 2012], BSSML[Gönen, 2014] and SMILE[Tan *et al.*, 2017]).

2 The MLCT Approach

The original co-training approach was applied to samples with multiple feature views [Blum and Mitchell, 1998], under the assumption that each feature view would provide sufficient and independent information to produce a classifier with a good generalization capability. In this paper, we mainly focus on mining multi-label samples naturally represented by multiple views. MLCT can also work on feature views generated by a particular view splitting technique [Chen *et al.*, 2011; Du *et al.*, 2011]. Let $\mathcal{X} = \{\mathbf{X}^v\}_{v=1}^m$ be m view representations of n samples, where each view $\mathbf{X}^v \in \mathbb{R}^{n\times d_v}$. $\mathbf{x}_j^v \in \mathbb{R}^{1\times d_v}$ is the d_v-dimensional feature vector for the j-th sample in the v-th view, and $\mathbf{y}_j \in \{\pm 1\}^q$ is the q-dimensional label vector for the j-th sample, where $\mathbf{y}_{j,c} = +1(-1)$ indicates whether the c-th $(1 \le c \le q)$ label is relevant (irrelevant) for the sample. Without loss of generality, we assume that the first l samples are labeled and the remaining $u = n - l$ ($l \ll u$) samples are unlabeled, $\mathcal{L} = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1}^l$, $\mathcal{U} = \{\mathbf{x}_j\}_{j=l+1}^n$.

The goal of MLCT is to perform multi-label co-training on $\mathcal{X}$ and $\{\mathbf{y}_j\}_{j=1}^n$, and to make accurate predictions on unseen new samples. To accomplish this goal, MLCT first uses correlations of labels to address the widely witnessed class-imbalance problem in multi-label learning, and to adjust the predicted label confidence values of samples. Next, it introduces a confidence measure to select samples, performs label-wise filtering on the predicted label confidence values of the selected samples, and communicates their labels among classifiers. The following subsections elaborate on the above two steps.

2.1 Addressing Class-imbalance via Label Correlation

In multi-label learning, labels have much fewer relevant samples than irrelevant ones, and the number of relevant samples varies significantly across labels [Zhang *et al.*, 2015a]. This *class-imbalance* issue can become even more pronounced during the iterative process of label communication in co-training. Inspired by the class-imbalance solutions for multi-label learning proposed in [Zhang *et al.*, 2015a; Sun and Lee, 2017], MLCT makes use of label correlation to address class-imbalance in multi-label co-training.

Two labels $c1$ and $c2$ are considered as positively correlated if they often co-occur as sample labels, and their correlation can be empirically estimated as follows:

$$\mathbf{C}(c2, c1) = \frac{\sum_{j=1}^{|\mathcal{L}|}[\mathbf{y}_{j,c1} = 1][\mathbf{y}_{j,c2} = 1]}{\sum_{j=1}^{|\mathcal{L}|}[\mathbf{y}_{j,c1} = 1]} \tag{1}$$

where $[\cdot]$ is true if and only if the condition is met. $\mathbf{C}(c2, c1)$ represents the probability that a sample is labeled as $c2$, given

that it's already labeled as $c1$. Although $c1$ and $c2$ might co-exist for some samples, the number of relevant samples for $c1$ might be far less than that of $c2$, or vice versa. As such, we separately compute $\mathbf{C}(c2, c1)$ and $\mathbf{C}(c1, c2)$ ($\mathbf{C}(c2, c1) \neq \mathbf{C}(c1, c2)$) to account for the imbalance phenomenon.

Suppose $\mathbf{f}_j^v = [\mathbf{f}_{j,1}^v, \cdots, \mathbf{f}_{j,q}^v] \in \mathbb{R}^q$ is the likelihood of $\mathbf{x}_j^v$ with respect to q labels in the v-th view, *initially* predicted by a specific multi-label classifier (i.e., ML-KNN [Zhang and Zhou, 2007]) and trained on the l labeled samples. To address the class-imbalance problem, MLCT makes use of label correlation to adjust the predictive confidence values of labels as follows:

$$\mathbf{p}_{j,c}^v = \frac{1}{1 + e^{-(\mathbf{w}_{j,c}^+ - \mathbf{w}_{j,c}^-)}} \tag{2}$$

$\mathbf{p}_{j,c}^v$ is the updated likelihood of the c-th label for $\mathbf{x}_j^v$. It follows the form of a logistic regression function to enforce that the adjusted value is within the range $(0, 1)$. $\mathbf{w}_{j,c}^-$ and $\mathbf{w}_{j,c}^+$ are computed as follows:

$$\mathbf{w}_{j,c}^- = 1 - \mathbf{w}_{j,c}^+ \tag{3}$$

$$\mathbf{w}_{j,c}^+ = \mathbf{f}_{j,c}^v + \frac{\sum_{c2=1, c2\neq c}^q \mathbf{f}_{j,c2}^v \mathbf{C}(c, c2)}{\sum_{c2=1}^q \triangle\,(\mathbf{C}(c, c2)) - 1} \tag{4}$$

where $\mathbf{w}_{j,c}^+(\mathbf{w}_{j,c}^-)$ reflects the confidence that c is a relevant (irrelevant) label for $\mathbf{x}_j^v$. $\triangle\,(a)$ is an indicator function; it's equal to 1 when $a > 0$, and to 0 otherwise. $\sum_{c2=1}^q \triangle\,(\mathbf{C}(c, c2)) - 1$ counts the number of labels positively correlated with the c-th label (c excluded). $\sum_{c2=1, c2\neq c}^q \mathbf{f}_{j,c2}^v \mathbf{C}(c, c2)$ indicates how much information the other labels, which are correlated with the c-th label, contributes to the relevance between c and $\mathbf{x}_j^v$. It is evident that the larger the margin $\mathbf{w}_{j,c}^+ - \mathbf{w}_{j,c}^-$ is, the more confident the prediction is. The margin indirectly reflects the relevance of the c-th label to $\mathbf{x}_j^v$.

Eq. (2) is used to solve the class-imbalance problem via co-occurrence between labels, which not only considers the condition where $\mathbf{x}_j^v$ contains the c-th label, but also takes the reverse case into account (i.e., the c-th label does not belong to the label set of $\mathbf{x}_j^v$). With the correlations between labels as extra information, the impact of the class-imbalance issue can be effectively reduced.

2.2 Communicating Label Information

It's crucial to be able to communicate with confidence samples and labels among classifiers during the iterative process of co-training. A good communication strategy helps obtaining a pronounced and stable performance [Du *et al.*, 2011]. Traditional co-training methods directly select samples with the most confident predictions for communication and model refinement [Blum and Mitchell, 1998; Zhou and Li, 2005; Levati *et al.*, 2017]. However, for co-training with multi-label samples, since a sample may be annotated with several correlated labels, how to communicate labels with confidence is more challenging. As an inductive multi-label co-training algorithm, COINs [Zhan and Zhang, 2017] deals with this challenge by communicating the single positive and single negative labels of an unlabeled example predicted with the largest confidence. As such, the refined model may be misled by the two communicated labels, and may result in performance degeneration.

A sample in different views should share the same relevant labels. To select samples and labels to be propagated with confidence, MLCT first summarizes the prediction reliability based on $m-1$ views (the v-th view is excluded), and measures the overall prediction reliability of the j-th sample with respect to the c-th label as follows:

$$\tilde{\mathbf{h}}_{j,c}^v = \frac{1}{m-1} \sum_{v'=1, v'\neq v}^m |\mathbf{p}_{j,c}^{v'} - (1 - \mathbf{p}_{j,c}^{v'})| \tag{5}$$

where $|\mathbf{p}_{j,c}^{v'} - (1 - \mathbf{p}_{j,c}^{v'})|$ is the prediction reliability of the c-th label of $\mathbf{x}_j^{v'}$. It's straightforward to see that a larger $\tilde{\mathbf{h}}_{j,c}^v$ value indicates that the $m-1$ classifiers are more in agreement on whether c should, or should not, be a relevant label for the j-th sample.

By extending the estimation in Eq. (5) to q labels, MLCT measures the overall prediction reliability of the j-th sample as follows:

$$\tilde{\mathbf{r}}_j^v = \frac{1}{q} \sum_{c=1}^q \tilde{\mathbf{h}}^v(j, c) \tag{6}$$

where larger $\tilde{\mathbf{r}}_j^v$ values imply more consistent predictions across the classifiers, and this makes the j-th sample a good candidate for communication. As such, MLCT selects u_b ($u_b \ll u$) samples corresponding to the largest $\tilde{\mathbf{r}}_j^v$ values as the candidate sample set $\mathcal{B}^v$ to be communicated for classifier refinement.

Next, to identify confident labels of the selected samples during co-training, MLCT defines two threshold values for each label on each view as follows:

$$\begin{aligned} \theta_+^v(c) &= \frac{\sum_{\mathbf{x}_j^v \in \mathcal{B}^v} \mathbf{f}_{j,c}^v [\mathbf{f}_{j,c}^v >= 0.5]}{\sum_{\mathbf{x}_j^v \in \mathcal{B}^v} [\mathbf{f}_{j,c}^v >= 0.5]} \\ \theta_-^v(c) &= \frac{\sum_{\mathbf{x}_j^v \in \mathcal{B}^v} \mathbf{f}_{j,c}^v [\mathbf{f}_{j,c}^v < 0.5]}{\sum_{\mathbf{x}_j^v \in \mathcal{B}^v} [\mathbf{f}_{j,c}^v < 0.5]} \end{aligned} \tag{7}$$

$\theta_+^v(c)$ is the average predicted likelihood of the c-th label on the v-th view, estimated using plausible relevant samples; similarly, $\theta_-^v(c)$ is the average predicted likelihood of the c-th label on the v-th view, estimated using plausible irrelevant samples. Since the sample distribution of each label is different, the above two threshold values are computed separately for each label. MLCT then uses the above threshold values to convert $\mathbf{f}_{j,c}^v$ into a binary label as follows:

$$\mathbf{b}_{j,c}^v = \begin{cases} 1, \text{ if } \mathbf{f}_{j,c}^v > \theta_+^v(c) \\ -1, \text{ if } \mathbf{f}_{j,c}^v < \theta_-^v(c) \\ 0, \text{otherwise} \end{cases} \tag{8}$$

MLCT then uses $\{\mathbf{b}_{j,c}^v\}_{v=1}^m$, and the feature information of the respective samples in the v-th view, to augment the labeled training set and to refine the classifier in the corresponding view. We do not apply sample-wise filtering here because different samples have a different number of relevant labels, and an appropriate filter threshold is difficult to pursue [Quevedo *et al.*, 2012]. The pseudo-code of MLCT is summarized in

Algorithm 1. MLCT first computes label correlations based on available labels (step 2). Then, it randomly selects u_a unlabeled samples and puts them in the buffer pool $\mathcal{B}$ (step 3), and independently predicts label likelihoods of these samples in each view (step 4-7). Next, it summarizes adjusted likelihoods and the overall predictive reliability of these samples in each view, chooses u_b samples with the largest reliability, applies label-wise filtering on the summarized predicted likelihoods, and forms the communication label sets (step 8-12). After this, it appends the communication label sets obtained from the respective views to the labeled sets, and removes it from the unlabeled set (step 13). After the maximum number of iterations is reached, MLCT finally returns the iteratively refined classifier of each view, and makes ensemble predictions for new samples via majority vote of the classifiers.

Algorithm 1 MLCT pseudo-code

Input:
$\mathcal{L}$: labeled samples set in m views;
$\mathcal{U}$: unlabeled samples set in m views;
$\mathcal{B}$: buffered unlabeled samples for co-training;
t: maximum number of iterations for co-training;
u_a, u_b: buffer size and number of communication samples.
Output:
H^v: the prediction model on the v-th $(1 \leq v \leq m)$ view.
1: **For** $iter = 1 : t$
2: Estimate label correlation $\mathbf{C}(c2, c1)(1 \leqslant c1, c2 \leqslant q)$ via Eq. (1);
3: Randomly pick u_a samples from $\mathcal{U}$ and put them into $\mathcal{B}$;
4: **For** $v = 1 : m$
5: Update classifier H^v based on $\mathcal{L}$ and make predictions on $\mathcal{B}$;
6: Adjust the initially predicted likelihoods $\mathbf{f}^v_{j,c}$ $(\mathbf{x}^v_j \in \mathcal{B})$ via Eq. (2);
7: **End For**
8: **For** $v = 1 : m$
9: Calculate $\tilde{\mathbf{r}}^v_j$ $(1 \leq j \leq u_a)$ via Eq. (6), then select u_b samples from $\mathcal{B}$ with the largest $\tilde{\mathbf{r}}^v_j$ and form the set $\mathcal{B}^v$;
10: Compute $\theta^v_+(c)$, $\theta^v_-(c)(1 \leqslant c \leqslant q)$ via Eq. (7);
11: Apply label-wise filtering on $\mathbf{f}^v_{j,c}$ $(\mathbf{x}^v_j \in \mathcal{B}^v)$ via Eq. (8), and form the communication label set $\Delta^v = \{\mathbf{b}^v_{j,c}\}$;
12: **End For**
13: Communicate $\Delta = \{\Delta^v\}^m_{v=1}$ to $\{H^v\}^m_{v=1}$, augment the labeled training set $\mathcal{L} = \mathcal{L} \bigcup \Delta$ and reduce the unlabeled training set $\mathcal{U} = \mathcal{U} - \Delta$.
14: **End For**
15: Return $\{H^v\}^m_{v=1}$.

3 Experiments

3.1 Experimental Setup

We assess the effectiveness of MLCT on four publicly accessible multi-label datasets from different domains, with different numbers of views and of samples [Gibaja *et al.*, 2016; Guillaumin *et al.*, 2010]. These datasets are described in Table 1. We also have computed the average imbalanced ratio (*ImR*) for all labels of each dataset. *ImR* reflects the degree of class-imbalance, and it is defined as follows [Sun and Lee, 2017]:

$$ImR = \frac{1}{q}\sum_{c=1}^{q}\frac{\sum_{j=1}^{n} max([\mathbf{y}_{j,c} = 1], [\mathbf{y}_{j,c} \neq 1])}{\sum_{j=1}^{n} min([\mathbf{y}_{j,c} = 1], [\mathbf{y}_{j,c} \neq 1])} \quad (9)$$

Data set	n	q	m	Avg	Min(ImR)	Max(ImR)	ImR
Emotions	593	6	2	1.87	1.24	3.00	2.32
Yeast	2417	14	2	4.23	1.32	70.08	8.95
Core15k	4999	260	6	1.47	3.46	2498.50	327.39
Pascal	9963	20	6	3.40	1.45	50.62	19.67

Table 1: Statistics of the datasets used for the experiments. n, q, and m are the number of examples, labels, and views, respectively. Avg is the average number of labels per sample, and ImR is the average imbalanced ratio for all labels in a dataset.

Max(ImR) and *Min(ImR)* represent the largest and the smallest imbalanced ratios of q labels, respectively. The larger the difference between Max(ImR), Min(ImR), and ImR, the more imbalanced the dataset is. From the statistics in Table 1, we can see that the labels of the last three datasets are quite imbalanced.

We compare the performance of MLCT against six representative and related multi-label learning algorithms: MLKNN [Zhang and Zhou, 2007], MLRGL [Bucak *et al.*, 2011], MLLOC [Huang and Zhou, 2012]), BSSML [Gönen, 2014], SMILE [Tan *et al.*, 2017] and COINs [Zhan and Zhang, 2017]. To enable experimental comparisons with multi-label learning methods on a single view, we concatenate the feature vectors of different views for each sample into a single vector, and use the latter as the input of the comparing methods. COINs performs the feature view splitting and classifier refinement during the iterative process, and optimizes two views using the concatenated feature vectors. MLCT directly refines m ML-KNN classifiers on the naturally split views during the iterative process.

We use five widely-used multi-label evaluation metrics: Hamming Loss (*HammLoss*), Average AUC (Area Under receiver operating Curve) (*AvgAUC*), Ranking Loss (*RankLoss*), One Error (*OneError*), and Average Precision (*AvgPrec*). Due to space limitation, we omit the formal definitions of these metrics; they can be found in [Zhang and Zhou, 2014; Gibaja and Ventura, 2015]. *HammLoss* requires transforming the predicted probabilistic label vector of a testing sample into a binary vector. Following the setting of ML-KNN, a label is considered relevant to the sample if its predicted probability is above 0.5, otherwise is considered irrelevant. The smaller the values of *HammLoss*, *RankLoss*, and *OneError* are, the better the performance is. As such, to be consistent with the other evaluation metrics, we report *1-HammLoss*, *1-RankLoss*, and *1-OneError* instead. With the latter measures, larger values indicate a better performance.

3.2 Experimental Results and Analysis

To compute the performance of MLCT, we randomly partition samples of each dataset into a training set (70%) and a testing set (30%). For the training set, we again randomly select 10% samples as the initial labeled data ($\mathcal{L}$) and the remaining as unlabeled data ($\mathcal{U}$) for co-training. We independently repeat the above partition 10 times, and report the average results and standard deviations. For co-training based methods, the maximum number of iterations (t) is fixed to 30, the number of samples (u_a) in the buffer pool $\mathcal{B}$ is fixed to $\lfloor u/t \rfloor$, and the number of samples (u_b) to be shared during the co-training process is fixed to $\lfloor 5\% u_a \rfloor$. The input parameters of the competitive methods are specified (or optimized) as indicated by

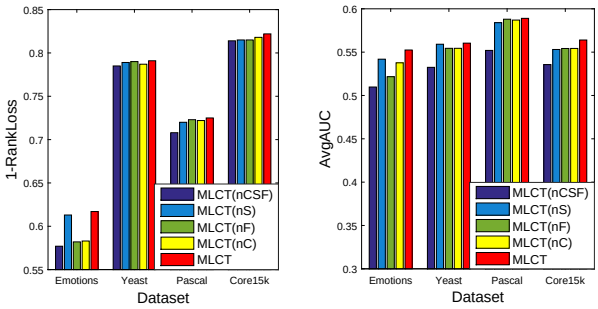
authors in the codes or papers. Table 2 shows the results.

From Table 2, we can see that MLCT generally outperforms comparing methods on different datasets and across the used metrics. We used the signed rank test [Demšar, 2006] to check for statistical significance between MLCT and the other methods (except SMILE and MLLOC). All the p-values are smaller than 0.05. Both MLCT and COINs are multi-label co-training methods, and MLCT frequently outperforms the latter. This is because COINs only communicates the single positive and negative labels predicted with highest confidence during co-training. Given the multi-label characteristics of multi-label samples, the two shared positive and negative labels may mislead the classifier update and thus degenerate the performance. In practice, we also randomly divided the concatenated view of the Pascal dataset into two views, and then applied MLCT on each view. Again, MLCT shows a better performance than COINs. MLCT runs much faster than COINs: their average runtimes on the first two datasets are 155.180 and 708.225 seconds, respectively. MLCT also outperforms two supervised multi-label solutions (ML-KNN and MLRGL), which uses different techniques for multi-label data classification. The performance margin shows the advantage of using unlabeled data for training. MLLOC explores label correlations locally and it holds comparable performance to MLCT, which employs label correlation globally. BSSML, SMILE, and MLCT use unlabeled data for training; BSSML loses to MLCT, and SMILE obtains comparable performance to MLCT. This fact suggests that co-training is an alternative and effective paradigm for using unlabeled data for semi-supervised multi-label learning. SMILE has higher AvgAUC than MLCT because the predicted likelihood vectors of SMILE are less sparse than MLCT. SMILE uses label correlation to replenish missing labels of instances before training the linear classifier, whereas MLCT directly uses the available label information for prediction.

Component Analysis

To further analyze the effect of the individual components of MLCT, we introduce four variants of MLCT:
(i) MLCT(nC): does not adjust the initially predicted likelihoods; in other words, it does not explicitly tackle the class-imbalance problem, and directly uses the initially predicted likelihoods during the whole iterative process.
(ii) MLCT(nS): first adjusts the initially predicted likelihoods, but randomly selects u_b samples; it then follows the same process as MLCT.
(iii) MLCT(nF): first adjusts the initially predicted likelihoods and selects samples based on the summarized likelihoods $\tilde{\mathbf{r}}_j^v$; it then communicates labels whose $\mathbf{f}_{j,c}^v >= 0.5$ (as done by ML-KNN), without applying label-wise filtering on the summarized likelihoods.
(iv) MLCT(nCSF): does not adjust the initially predicted likelihoods, randomly selects u_b samples, and then communicates labels whose $\mathbf{f}_{j,c}^v >= 0.5$.

Figure 1 gives the results obtained with MLCT and its variants. Overall MLCT outperforms its variants, and MLCT(nCSF) usually has the lowest performance. MLCT(nF), MLCT(nS), and MLCT(nC) outperform MLCT(nCSF). MLCT(nC) is always better than MLCT(nCSF), and is worse than MLCT on various datasets,



(a) *1-RankLoss* (b) *AvgAUC*

Figure 1: *1-RankLoss* and *AvgAUC* of MLCT and its variants on different datasets.

especially on Core15k, which is the most imbalanced dataset. This observation indirectly reflects the effectiveness of addressing class-imbalance during the co-training process. Moreover, we can see that MLCT(nS), which randomly selects the same number of samples for communication, is always outperformed by MLCT. This fact shows the effectiveness of MLCT in selecting samples with high confidence. We used a signed rank test to verify the statistical significance of the results of MLCT and its variants, and all p-values are smaller than 0.02. These results support the fact that MLCT is effective in addressing class-imbalance, and is capable of communicating labels with confidence for multi-label co-training.

Parameter Sensitivity Analysis

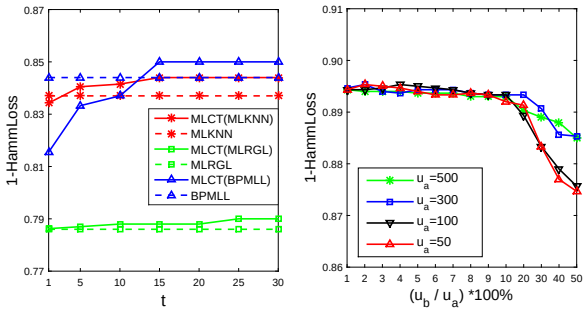
As with other co-training methods [Blum and Mitchell, 1998; Zhou and Li, 2005; Zhan and Zhang, 2017], two input parameters (t and u_b) may affect the performance of MLCT. We conduct additional experiments to study the sensitivity of MLCT with respect to these parameters. Due to space limitation, we report only the average results with respect to *1-HammLoss*. In fact, results with respect to other metrics provide similar patterns and conclusions.

Figure 2(a) shows the results for MLCT under different values of t; u_a and u_b were set to $\lfloor u/t \rfloor$ and $\lfloor 5\% u_a \rfloor$, respectively. MLCT has an increasing *1-HammLoss* as t increases, and it reaches a plateau after 15 iterations. Initially, MLCT has a lower performance than ML-KNN; that is because ML-KNN is trained on the integrated view, whereas MLCT works on separate views. To further study the generalization ability of MLCT, we also investigate the performance of MLCT with MLRGL [Bucak *et al.*, 2011] and BPMLL [Zhang and Zhou, 2006] as base classifiers (instead of ML-KNN), and report these results in Figure 2(a). Again, MLCT has a performance that is superior to that of the adopted base classifiers.

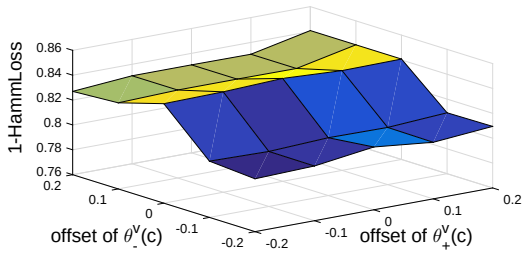
Figure 2(b) exhibits the *1-HammLoss* of MLCT for different number of selected samples (u_b) for communication, with u_a fixed to 50, 100, 300, and 500, respectively. The results are average on three datasets. *Emotions* is excluded from this experiment, since its small number of samples prevents the same setting of u_a as for the other datasets. MLCT holds a stable performance when the ratio u_b/u_a increases from 1% to 10%, and then shows a decreasing trend as $u_b/u_a > 10\%$. This indicates that a reasonable number of samples can be easily selected to achieve an effective co-training for MLCT.

Metric	BSSML	SMILE	MLLOC	MLRGL	MLKNN	COINs	MLCT
				Emotions			
1-HammLoss	0.699 ± 0.020○	0.640 ± 0.004●	0.694 ± 0.007○	0.594 ± 0.003●	0.660 ± 0.023●	0.650 ± 0.014●	0.691 ± 0.007
1-RankLoss	0.345 ± 0.036●	0.614 ± 0.008●	0.712 ± 0.016○	0.577 ± 0.015●	0.547 ± 0.029●	0.588 ± 0.022●	0.617 ± 0.029
AvgPrec	0.477 ± 0.017●	0.601 ± 0.004●	0.548 ± 0.025●	0.562 ± 0.015●	0.559 ± 0.021●	0.582 ± 0.012●	0.608 ± 0.027
1-OneError	0.272 ± 0.028●	0.466 ± 0.015○	0.430 ± 0.018●	0.412 ± 0.049●	0.412 ± 0.057	0.451 ± 0.023	0.456 ± 0.048
AvgAUC	0.602 ± 0.029○	0.647 ± 0.025○	0.674 ± 0.022○	0.542 ± 0.011●	0.525 ± 0.045●	0.557 ± 0.043○	0.552 ± 0.009
				Yeast			
1-HammLoss	0.724 ± 0.005●	0.732 ± 0.008●	0.711 ± 0.003●	0.722 ± 0.008●	0.776 ± 0.007○	0.768 ± 0.004●	0.772 ± 0.003
1-RankLoss	0.499 ± 0.024●	0.755 ± 0.011●	0.700 ± 0.013●	0.787 ± 0.004●	0.797 ± 0.006○	0.789 ± 0.005●	0.791 ± 0.005
AvgPrec	0.505 ± 0.017●	0.686 ± 0.015●	0.504 ± 0.009●	0.699 ± 0.007●	0.715 ± 0.011	0.715 ± 0.006	0.715 ± 0.007
1-OneError	0.728 ± 0.018●	0.696 ± 0.027●	0.814 ± 0.084○	0.748 ± 0.011●	0.731 ± 0.014●	0.714 ± 0.010●	0.749 ± 0.017
AvgAUC	0.550 ± 0.053	0.594 ± 0.010○	0.604 ± 0.009○	0.625 ± 0.003○	0.581 ± 0.018○	0.615 ± 0.009○	0.562 ± 0.011
				Pascal07			
1-HammLoss	0.836 ± 0.025●	0.888 ± 0.000●	0.926 ± 0.000●	0.882 ± 0.000●	0.927 ± 0.000	0.826 ± 0.013●	0.927 ± 0.000
1-RankLoss	0.661 ± 0.034●	0.752 ± 0.007○	0.775 ± 0.008○	0.695 ± 0.008●	0.721 ± 0.006●	0.701 ± 0.008●	0.725 ± 0.006
AvgPrec	0.205 ± 0.017●	0.450 ± 0.006●	0.227 ± 0.005●	0.424 ± 0.005●	0.439 ± 0.005●	0.412 ± 0.003●	0.453 ± 0.005
1-OneError	0.356 ± 0.037●	0.406 ± 0.007●	0.250 ± 0.010●	0.405 ± 0.008●	0.399 ± 0.008●	0.362 ± 0.013●	0.410 ± 0.007
AvgAUC	0.553 ± 0.056●	0.664 ± 0.007○	0.807 ± 0.009○	0.535 ± 0.005●	0.577 ± 0.008●	0.587 ± 0.005●	0.589 ± 0.009
				Core15k			
1-HammLoss	0.986 ± 0.000●	0.956 ± 0.000●	0.974 ± 0.000●	0.948 ± 0.002●	0.987 ± 0.000	0.943 ± 0.000●	0.987 ± 0.000
1-Rankloss	0.606 ± 0.017●	0.788 ± 0.007●	0.827 ± 0.005○	0.744 ± 0.005●	0.821 ± 0.003●	0.794 ± 0.000●	0.822 ± 0.005
AvgPrec	0.122 ± 0.011●	0.288 ± 0.008○	0.248 ± 0.010●	0.162 ± 0.021●	0.255 ± 0.006●	0.203 ± 0.000●	0.256 ± 0.006
1-OneError	0.138 ± 0.015●	0.284 ± 0.015○	0.356 ± 0.038○	0.054 ± 0.084●	0.271 ± 0.021●	0.318 ± 0.000○	0.275 ± 0.012
AvgAUC	0.641 ± 0.029○	0.658 ± 0.010○	0.815 ± 0.006○	0.594 ± 0.004○	0.567 ± 0.005○	0.574 ± 0.000○	0.564 ± 0.005

Table 2: Results on different datasets. ●/○ indicates whether MLCT is statistically (according to pairwise t-test at 95% significance level) superior/inferior to the other methods.



(a) Sensitivity of t (b) Sensitivity of u_b/u_a



(c) Sensitivity of offsets for $\theta^v_+(c)$ and $\theta^v_-(c)$

Figure 2: *1-HammLoss* of MLCT under different input values of t (maximum number of iterations), u_b (number of selected samples for communication), and different offsets of $\theta^v_+(c)$ and $\theta^v_-(c)$ (label-wise thresholds).

MLCT also applies label-wise thresholds ($\theta^v_+(c)$ and $\theta^v_-(c)$) to filter positive and negative labels. Figure 2(c) shows the *1-HammLoss* of MLCT under different offsets of these two thresholds. Particularly, +0.2 (-0.2) means increasing (decreasing) the respective threshold by 0.2. From the Figure, we can observe that directly using $\theta^v_+(c)$ and $\theta^v_-(c)$ (both with offset as 0) gives a better performance than other values. In practice, we also tested a threshold fixed to 0.5, and the obtained performance is lower than that of MLCT. These results demonstrate the importance of label-wise filtering. Irrespective of the offset for $\theta^v_+(c)$, either decreasing or increasing $\theta^v_-(c)$ downgrades the performance. This is because decreasing $\theta^v_-(c)$ results in a more stringent rule for negative samples, whereas increasing $\theta^v_-(c)$ causes the detection of positive samples as negative ones. On the other hand, when the offset for $\theta^v_-(c)$ is 0, MLCT shows a stable performance for different offsets of $\theta^v_+(c)$. This is because the number of relevant samples for label c is generally smaller than the number of irrelevant samples for this label; as such, MLCT can identify the relevant samples of the c-th label even with a moderately decreased or increased $\theta^v_+(c)$. In practice, we investigated the margin $(\theta^v_+(c) - \theta^v_-(c))$ and found it is generally larger than 0.5 across all labels. This investigation shows that label-wise filtering is important for multi-label co-training and the adaptive threshold values are effective. In summary, from these results, we can conclude that MLCT is robust to key input parameters.

4 Conclusions

In this paper, we study the multi-label co-training problem, an interesting but seldom studied learning paradigm. We introduce a solution to address the issue of class-imbalance, and to communicate confident labels of multi-label samples during the co-training process. Experimental results show that the proposed solution works better than other related methods. Several avenues remain to be explored, including how to accurately estimate label correlation from limited labeled data, and how to filter relevant labels more reliably during multi-label co-training. The code of MLCT is available at: http://mlda.swu.edu.cn/codes.php?name=MLCT.

Acknowledgments

This work is supported by NSFC (61741217, 61402378 and 61732019), Open Research Project of Hubei Key Laboratory of Intelligent Geo-Information Processing (KLIGIP-2017A05).

References

[Blum and Mitchell, 1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, pages 92–100, 1998.

[Bucak *et al.*, 2011] Serhat Selcuk Bucak, Rong Jin, and Anil K Jain. Multi-label learning with incomplete class assignments. In *CVPR*, pages 2801–2808, 2011.

[Chen *et al.*, 2011] Minmin Chen, Yixin Chen, and Kilian Q Weinberger. Automatic feature decomposition for single view co-training. In *ICML*, pages 953–960, 2011.

[Demšar, 2006] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[Du *et al.*, 2011] Jun Du, Charles X Ling, and Zhihua Zhou. When does cotraining work in real data? *TKDE*, 23(5):788–799, 2011.

[Gibaja and Ventura, 2015] Eva Gibaja and Sebastián Ventura. A tutorial on multilabel learning. *ACM Computing Surveys*, 47(3):52, 2015.

[Gibaja *et al.*, 2016] Eva Gibaja, Jose Moyano, and Sebastián Ventura. An ensemble-based approach for multi-view multi-label classification. *Progress in Artificial Intelligence*, 5(4):251–259, 2016.

[Gönen, 2014] Mehmet Gönen. Coupled dimensionality reduction and classification for supervised and semi-supervised multilabel learning. *Pattern Recognition Letters*, 38:132–141, 2014.

[Guillaumin *et al.*, 2010] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multimodal semi-supervised learning for image classification. In *CVPR*, pages 902–909, 2010.

[Guo and Schuurmans, 2012] Yuhong Guo and Dale Schuurmans. Semi-supervised multi-label classification: a simultaneous large-margin, subspace learning approach. In *ECML/PKDD*, pages 355–370, 2012.

[Huang and Zhou, 2012] Shengjun Huang and Zhihua Zhou. Multi-label learning by exploiting label correlations locally. In *AAAI*, pages 949–955, 2012.

[Kong *et al.*, 2013] Xiangnan Kong, Michael K Ng, and Zhihua Zhou. Transductive multilabel learning via label set propagation. *TKDE*, 25(3):704–719, 2013.

[Levati *et al.*, 2017] Jurica Levati, Michelangelo Ceci, Dragi Kocev, and Sao Deroski. Self-training for multi-target regression with tree ensembles. *Knowledge-Based Systems*, 123(C):41–60, 2017.

[Qian and Davidson, 2010] Buyue Qian and Ian Davidson. Semi-supervised dimension reduction for multi-label classification. In *AAAI*, pages 569–574, 2010.

[Quevedo *et al.*, 2012] José Ramón Quevedo, Oscar Luaces, and Antonio Bahamonde. Multilabel classifiers with a probabilistic thresholding strategy. *Pattern Recognition*, 45(2):876–883, 2012.

[Sun and Lee, 2017] Kaiwei Sun and Chong Ho Lee. Addressing class-imbalance in multi-label learning via two-stage multi-label hypernetwork. *Neurocomputing*, 266:375–389, 2017.

[Tan *et al.*, 2017] Qiaoyu Tan, Yanming Yu, Guoxian Yu, and Jun Wang. Semi-supervised multi-label classification using incomplete label information. *Neurocomputing*, 260:192–202, 2017.

[Wu and Zhang, 2013] Le Wu and Minling Zhang. Multi-label classification with unlabeled data: An inductive approach. In *ACML*, pages 197–212, 2013.

[Wu *et al.*, 2015] Baoyuan Wu, Siwei Lyu, and Bernard Ghanem. Ml-mg: multi-label learning with missing labels using a mixed graph. In *ICCV*, pages 4157–4165, 2015.

[Yu *et al.*, 2012] Guoxian Yu, Carlotta Domeniconi, Huzefa Rangwala, Guoji Zhang, and Zhiwen Yu. Transductive multi-label ensemble classification for protein function prediction. In *KDD*, pages 1077–1085, 2012.

[Yu *et al.*, 2014] Hsiang-Fu Yu, Prateek Jain, Purushottam Kar, and Inderjit Dhillon. Large-scale multi-label learning with missing labels. In *ICML*, pages 593–601, 2014.

[Zhan and Zhang, 2017] Wang Zhan and Minling Zhang. Inductive semi-supervised multi-label learning with co-training. In *KDD*, pages 1305–1314, 2017.

[Zhang and Yeung, 2009] Yu Zhang and Dit Yan Yeung. Semi-supervised multi-task regression. In *ECML/PKDD*, pages 617–631, 2009.

[Zhang and Zhou, 2006] Minling Zhang and Zhihua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *TKDE*, 18(10):1338–1351, 2006.

[Zhang and Zhou, 2007] Minling Zhang and Zhihua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

[Zhang and Zhou, 2014] Minling Zhang and Zhihua Zhou. A review on multi-label learning algorithms. *TKDE*, 26(8):1819–1837, 2014.

[Zhang *et al.*, 2015a] Minling Zhang, Yukun Li, and Xuying Liu. Towards class-imbalance aware multi-label learning. In *IJCAI*, pages 4041–4047, 2015.

[Zhang *et al.*, 2015b] Xiang Zhang, Naiyang Guan, Zhigang Luo, and Xuejun Yang. Constrained projective nonnegative matrix factorization for semi-supervised multi-label learning. In *ICMLA*, pages 588–593, 2015.

[Zhou and Li, 2005] Zhihua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *TKDE*, 17(11):1529–1541, 2005.

[Zhu, 2008] Xiaojin Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, Tech. Rep.: 1530, 2008.