ML-MG: Multi-label Learning with Missing Labels Using a Mixed Graph

Baoyuan Wu  
KAUST  
Saudi Arabia  
baoyuan.wu@kaust.edu.sa

Siwei Lyu  
University at Albany, SUNY  
Albany, NY, USA  
slyu@albany.edu

Bernard Ghanem  
KAUST  
Saudi Arabia  
bernard.ghanem@kaust.edu.sa

Abstract

This work focuses on the problem of multi-label learning with missing labels (MLML), which aims to label each test instance with multiple class labels given training instances that have an incomplete/partial set of these labels (i.e. some of their labels are missing). To handle missing labels, we propose a unified model of label dependencies by constructing a mixed graph, which jointly incorporates (i) instance-level similarity and class co-occurrence as undirected edges and (ii) semantic label hierarchy as directed edges. Unlike most MLML methods, We formulate this learning problem transductively as a convex quadratic matrix optimization problem that encourages training label consistency and encodes both types of label dependencies (i.e. undirected and directed edges) using quadratic terms and hard linear constraints. The alternating direction method of multipliers (ADMM) can be used to exactly and efficiently solve this problem. To evaluate our proposed method, we consider two popular applications (image and video annotation), where the label hierarchy can be derived from Wordnet. Experimental results show that our method leads to a significant improvement in performance and robustness to missing labels over the state-of-the-art methods.

1. Introduction

In multi-label learning, we assume that an instance can be assigned to multiple classes simultaneously. For example, an image can be annotated using several tags, and a document can be associated with multiple topics. Although many multi-label learning methods have been proposed, many of such methods require completely labeled training instances. In practical applications, most training instances are only partially labeled, with some or all of the labels not provided/missing. Let us consider the task of large-scale image annotation, where the number of classes/tags is large (e.g. using labels of ImageNet [12]). In this case, a human annotator can only realistically annotate each training image with a subset of tags. Learning from such partially labeled instances is referred to as the multi-label learning with missing labels (MLML) problem [37, 43].

As labels are usually related by semantic meanings or co-occurrences, the key to filling and learning from missing labels is a good model to represent label dependency. One widely used model for label dependency is an undirected graph, through which the label information can be propagated among different instances and among different classes. For example, the label dependency between a pair of labels, such as instance similarity and class co-occurrence can be represented using such a graph (see green and blue edges in Figure 1). However, as stated in [37, 39], class co-occurrence that is derived from training labels can be inaccurate and even detrimental when many missing labels exist. Li et al. [23] propose to alleviate this limitation...
by using an auxiliary source (such as Wikipedia) to estimate co-occurrence relations.

Interestingly, the semantic dependency between two classes, such as “animal->horse” and “plant->grass” as shown in Figure 1, can foster further label dependency and improve label predictions in the test. In this case, it is intuitive to require that the label score (e.g. the presence probability) of the parent class cannot be lower than that of its child class. This is traditionally referred to as the semantic hierarchy constraint [2]. The undirected graph (with instance similarity and class co-occurrence edges) cannot guarantee that the final label predictions will satisfy all semantic hierarchy constraints. To address this problem, we add semantic dependencies into the graph as directed edges, thus, resulting in an overall mixed graph that encourages (or enforces) three types of label dependency (refer to Figure 1 for an example).

The goal of this work is to learn from partially labeled training instances and to correctly predict the labels of test instances which should satisfy the semantic hierarchy constraints. Firstly, motivated by [37, 39], a discrete objective function is formulated to simultaneously encourage consistency between predicted and ground truth labels and encode traditional label dependencies (instance similarity and class co-occurrence). Whereas, semantic hierarchy constraints are incorporated as hard linear constraints in the matrix optimization. The discrete problem is further relaxed to a continuous convex problem, which can be solved using ADMM [3, 44, 45, 46]. We summarize our contributions next.

Contributions: (1) We address the MLML problem by using a mixed graph to encode a network of label dependencies: instance similarity, class co-occurrence, and semantic hierarchy. (2) Learning on this mixed graph is formulated as a linearly constrained convex matrix optimization problem that is amenable to efficient solvers. (3) Our extensive experiments on the tasks of image and video annotation show the superiority of our method in comparison to the state-of-the-art. (4) We augment labeling of several widely used datasets, including Corel5k [13], ESP Game [32], IAPRTC-12 [20] and MediaMill [28], with a fully labeled semantic hierarchy drawn from Wordnet [14]. This ground truth augmentation will be made publicly available to enable further research on the MLML problem in computer vision.

2. Related work

In the literature of multi-label learning, the previous works that are designed to handle the missing labels can be generally partitioned into four categories. First, the missing labels are directly treated as negative labels, including [8, 29, 5, 9, 1, 34, 35, 10]. Common to these methods is that the label bias is brought into the objective function. As a result, their performance is greatly affected when massive ground-truth positive labels are initialized as negative labels. Second, filling in missing labels is treated as a matrix completion (MC) problem, including [19, 6, 41]. A recent work called LEML [43] is proposed in the empirical risk minimization (ERM) framework. Both MC models and LEML are based on the low rank assumption. Although the low rank assumption is widely used, it may not hold in practical multi-label problem. Third, missing labels can be treated as latent variables in probabilistic models, including the model based on Bayesian network [22, 31] and conditional restricted Boltzmann machines (CRBM) model. Last, Wu et al. [37] define three label states, including positive labels +1, negative labels −1 and missing labels 0, to avoid the label bias. However, the two solutions proposed in [37] involves matrix inversion, which limits the scalability to handle larger datasets. Wu et al. [39] propose an inductive model based on the framework of regularized logistic regression. It also adopts three label states and a hinge loss function to avoid the label bias. However, the classifier parameters corresponding to each class have to be learned sequentially. Furthermore, the computational cost of this method increases significantly with the number of classes and becomes prohibitive for very large datasets.

Hierarchical multi-label learning (HML) [18] has been applied to problems where the label hierarchy exists, such as image annotation [30], text classification [26, 27] and protein function prediction [2, 42]. Except for a few cases, most existing HML methods only consider the learning problem of complete hierarchical labels. However, in real problems, the incomplete hierarchical labels commonly occur, such as in image annotation. Yu et al. [42] recently propose a method to handle the incomplete hierarchical labels. However, the semantic hierarchy and the multi-label learning are used separately, such that the semantic hierarchy constraint cannot be fully satisfied. Deng et al. [11] develops a CRF model for object classification. The semantic hierarchy constraint and missing labels are also incorporated in this model. However, a significant difference is that [11] focuses on a single object in each instance, while there are multiple object in each instance in our problem.

In the applications of image annotation and video annotation, both missing labels and semantic hierarchy have been explored in many previous works, such as [29, 5, 9, 40, 24, 16] (missing labels) and [30] (semantic hierarchy). However, to the best of our knowledge, no previous work in image and video annotation has extensively studied missing labels and semantic hierarchy simultaneously.

Note that the semantic hierarchy constraint used in our model is similar to the ranking constraint [15, 5] that is widely used in multi-label ranking models, but there are significant differences. Firstly, the ranking constraint used in these models means the predicted value of the provided positive label should be larger than the one of the provided negative label, while the semantic hierarchy constraint involves the ranking between the parent and the child class.
Besides, the ranking constraint is always incorporated as the loss function, while the semantic hierarchy constraint is formulated as the linear constraint in our model.

3. Problem and proposed model

3.1. Problem definition

Similar to traditional MLML problems, our method takes as input two matrices: a data matrix \( X = [x_1, \cdots, x_n] \in \mathbb{R}^{d \times n} \), which aggregates the \( d \)-dimensional feature vectors of all \( n \) (training and test) instances, and a label matrix \( Y = [y_1, \cdots, y_n] \in \{0, \frac{1}{2}, +1\}^{m \times n} \), which aggregates the \( m \)-dimensional label vectors of the instances. Therefore, each instance \( x_i \) can take one or more labels from the \( m \) different classes \( \{c_1, \ldots, c_m\} \). Its corresponding label vector \( y_i = Y_{i\cdot} \) determines its membership to each of these classes. For example, if \( Y_{ij} = +1 \), then \( x_i \) is a member of \( c_j \) and if \( Y_{ij} = 0 \), then \( x_i \) is not a member of this class. However, if \( Y_{ij} = \frac{1}{2} \), then the membership of \( x_i \) to \( c_j \) is considered missing (i.e. it has a missing label).

With this notation, all \( m \) labels of each testing instance \( x_k \) are missing, i.e. \( y_k = \frac{1}{2} \). The semantic hierarchy is encoded as another matrix: \( \Phi = [\phi_1, \cdots, \phi_n] \in \mathbb{R}^{m \times n} \), with \( n_c \) being the number of directed edges. \( \phi_j = [0, \ldots, 1, \ldots, 0]^{\top} \) denotes the index vector of the \( j \)-th directed edge (see Figure 1), with \( \phi_i(\text{parent}) = 1 \) and \( \phi_i(\text{child}) = -1 \), while all other entries are 0.

Our goal is to obtain a complete label matrix \( Z \in \{+1, 0\}^{m \times n} \) that satisfies the following three properties simultaneously.

(i) \( Z \) is sufficiently consistent with the provided (not missing) labels in \( Y \), i.e. \( Z_{ij} = Y_{ij} \) if \( Y_{ij} \neq \frac{1}{2} \).

(ii) \( Z \) benefits from label similarity among similar instances and class co-occurrence, i.e. if \( x_i \) and \( x_j \) have similar features or the classes they belong to co-occur, then their corresponding predicted labels should be similar. (iii) \( Z \) is consistent with the semantic hierarchy \( \Phi \). To enforce this, we ensure that if \( c_a \) is the parent of \( c_b \), a hard constraint is applied, which guarantees that the score (the presence probability) of \( c_a \) should not be smaller than the score of \( c_b \). This constraint ensures that the final predicted labels are consistent with the semantic hierarchy. By incorporating all three criteria simultaneously, label information is propagated from provided labels to the missing labels.

3.2. Label consistency

Label consistency of \( Z \) with \( Y \) is formulated as

\[
\ell(Y, Z) = \sum_{i,j} Y_{ij}(Y_{ij} - Z_{ij}) = \text{const} - \text{tr}(Y^\top Z),
\]

where \( \text{const} = \text{tr}(Y^\top Y) \), and \( Y \) is defined as \( Y_{ij} = (2Y_{ij} - 1)\tau_{ij} \), where \( \tau_{ij} \) is a penalty function mismatches between \( Y_{ij} \) and \( Z_{ij} \). We set \( \tau_{ij} \) in the following manner. If \( Y_{ij} = 0 \), then \( \tau_{ij} = r_- > 0 \), if \( Y_{ij} = +1 \), then \( \tau_{ij} = r_+ > r_- \), and if \( Y_{ij} = \frac{1}{2} \), then \( \tau_{ij} = 0 \). In doing so and unlike the work in [37, 39], a higher penalty is incurred if a ground truth label is \( +1 \) and predicted as 0, as compared to the reverse case. This idea embeds the observation that most entries of \( Y \) in many multi-label datasets (with a relatively large number of classes) are 0 and that \( +1 \) labels are rare. Of course, missing labels are not penalized. Eq (1) satisfies the label consistency. When \( Y_{ij} = +1 \), \( Z_{ij} \) is encouraged to be \( +1 \); when \( Y_{ij} = 0 \), \( Z_{ij} \) is encouraged to be 0; when \( Y_{ij} = \frac{1}{2} \), there is no constraint on \( Z_{ij} \).

3.3. Instance-level label dependency

Unlike the work in [37, 39], we incorporate instance-level label similarity, i.e. property (ii), using the regularization term in Eq (2).

\[
\text{tr}(ZL_XZ^\top) = \sum_{k,i,j} W_X(i,j) \left[ \frac{Z_{ki}}{\sqrt{d_X(i)}} - \frac{Z_{kj}}{\sqrt{d_X(j)}} \right]^2,
\]

where instance similarity matrix \( W_X \) is: \( W_X(i,j) = \exp(-||x_i - x_j||^2 \epsilon) \), \( \forall i \neq j \) and \( W_X(i,i) = 0 \). The kernel size \( \epsilon \) is \( \|x_i - x_h\|_2 \) and \( x_h \) is the \( h \)-th nearest neighbor of \( x_i \). Similar to [37], we set \( h = 7 \). The normalization term \( d_X(i) = \sum_j W_X(i,j) \) makes the regularization term invariant to the different scaling factors of the elements of \( W_X \) [33]. The normalized Laplacian matrix is \( L_X = I - D_X^{-\frac{1}{2}}W_XD_X^{-\frac{1}{2}} \) with \( D_X = \text{diag}(d_X(1), \cdots, d_X(n)) \).

3.4. Class-level label dependency

Here, we consider two types of class-level label dependency, namely class co-occurrence and semantic hierarchy.

Class co-occurrence: This dependency is encoded using the regularization term in Eq (3).

\[
\text{tr}(Z^\top L_C Z) = \sum_{k,i,j} W_C(i,j) \left[ \frac{Z_{ik}}{\sqrt{d_C(i)}} - \frac{Z_{jk}}{\sqrt{d_C(j)}} \right]^2,
\]

Here, we define the class similarity matrix \( W_C \) as: \( W_C(i,j) = \frac{x_i^\top x_j^\top}{\|x_i^\top\|_1 \|x_j^\top\|_1} \), \( \forall i \neq j \) and \( W_C(i,i) = 0 \). The normalized Laplacian matrix is defined as \( L_C = I - D_C^{-\frac{1}{2}}W_C D_C^{-\frac{1}{2}} \) with \( D_C = \text{diag}(d_C(1), \cdots, d_C(m)) \).

Semantic hierarchy: To enforce the semantic hierarchy constraint, i.e. property (iii), we apply the following constraint: \( Z(i_{\text{parent}}, j) \geq Z(i_{\text{child}}, j) \), \( \forall i = 1, \ldots, n_c, \forall j = 1, \ldots, n \). The resulting constraints can be aggregated in matrix form, as

\[
\Phi^\top Z \geq 0.
\]
3. ADMM framework [3], we can minimize Problem (6) by the Frobenius matrix norm. Following the conventional method), \( \rho > 0 \), and \( t = 1 \)

2: while not converge do
3: Update \( Z_{t+1} \) according to Eq (8);  
4: Update \( Q_{t+1} \) according to Eq (9); 
5: Update \( \Lambda_{t+1} \) according to Eq (10); 
6: Set \( t = t + 1; \rho = 5\rho \) 
7: end while 
8: return \( Z^* = Z_{t+1} \).

3.5. MLML with mixed graph (ML-MG)

We formulate the MLML problem using the constructed mixed graph as a binary matrix optimization problem by linearly combining Eqs (1,2,3) to form the objective and Eq (4) to enforce the semantic hierarchy constraints.

\[
\text{arg min}_{Z} \ - \text{tr}(Y^T Z) + \beta \text{tr}(ZL_x Z^\top) + \gamma \text{tr}(Z^\top L_C Z), \\
\text{s.t.} \quad Z \in \{0, 1\}^{m \times n}, \quad \Phi^T Z \geq 0.
\]  
(5)

Due to the binary constraint on \( Z \), it is difficult to efficiently solve this discrete problem. Thus, we use a conventional box relaxation, which relaxes \( Z \) to take on values in \([0, 1]^{m \times n}\). The relaxed ML-MG problem in Eq (5) is a convex quadratic problem (QP) with linear matrix constraints (refer to the supplementary material for more details).

\[
\text{arg min}_{Z} \ - \text{tr}(Y^T Z) + \beta \text{tr}(ZL_x Z^\top) + \gamma \text{tr}(Z^\top L_C Z), \\
\text{s.t.} \quad Z \in [0, 1]^{m \times n}, \quad \Phi^T Z \geq 0.
\]  
(6)

Due to its convexity and smoothness, the ML-MG problem can be efficiently solved by many solvers. In this work, we adopt the alternative direction of method of multipliers (ADMM) [3] for this purpose.

4. ADMM for ML-MG

After adding a non-negative slack variable \( Q \in \mathbb{R}_n^{n \times n} \), we can formulate the augmented Lagrange function of Problem (6) as in Eq (7), where \( Z \in [0, 1]^{m \times n} \) and \( Q \geq 0 \).

\[
L_{\rho}(Z, Q, \Lambda) = -\text{tr}(Y^T Z) + \beta \text{tr}(ZL_x Z^\top) + \gamma \text{tr}(Z^\top L_C Z) + \text{tr}(\Lambda^T (Z - Q)) + \frac{\rho}{2} ||(Z - Q)||_F^2
\]  
(7)

Here, \( \Lambda \in \mathbb{R}_n^{n \times n} \) is the Lagrange multiplier (dual variable), \( \rho > 0 \) is a tradeoff parameter, and \( || \cdot ||_F \) denotes the Frobenius matrix norm. Following the conventional ADMM framework [3], we can minimize Problem (6) by alternating index. When updating \( Z_t \) in Eq (8), we set \( \overline{X}_t = -Y + \Phi \Lambda_q - \rho \Phi Q_t, \overline{B}_t = \beta L_x \) and \( \overline{C}_t = \gamma L_C + \frac{\rho}{2} \Phi \Phi^\top \).

The resulting problem is a convex QP with box constraints that can be efficiently solved using projected gradient descent (PGD) with exact line search [4]. Moreover, the updates for \( Q_t \) and \( \Lambda_t \) are closed form. More details of the optimization are provided in the supplementary material.

\[
Z_{t+1} = \text{arg min}_{Z \in [0, 1]^{m \times n}} L_{\rho}(Z, Q_t, \Lambda_t)
\]  
(8)

\[
= \text{arg min}_{Z \in [0, 1]^{m \times n}} \text{tr}(\overline{X}_t^\top Z) + \text{tr}(\overline{B}_t Z^\top) + \text{tr}(Z^\top \overline{C}_t Z)
\]  
(9)

\[
Q_{t+1} = \text{arg min}_{Q \geq 0} L_{\rho}(Z_{t+1}, Q, \Lambda_t)
\]  

\[
= \max(0, \Phi^\top Z_{t+1} + \frac{1}{\rho} \Lambda_{t}^\top)
\]  
(10)

\[
\Lambda_{t+1} = \Lambda_t + \rho(\Phi^\top Z_{t+1} - Q_{t+1}).
\]  

The ADMM solution to the ML-MG problem is summarized in Algorithm (1). Based on the proof in [17, 25], our ADMM algorithm is guaranteed to converge to the global minimum of Problem (6).

5. Experiments

In this section, we evaluate the proposed method and the state-of-the-art methods on four benchmark datasets in image annotation and video annotation.

5.1. Experimental setup

Datasets. Four benchmark multi-label datasets are used in our experiments, including Corel5k [13], ESP Game [32], IAPRTC-12 [20], and MediaMill [28]. These datasets are chosen because they are representative and popular benchmarks for comparative analysis among MLML methods. Since the scope of this paper is not feature design, we obtain the data and label matrices (\( X \) and \( Y \)) of the first three image datasets from the seminal work [21] 1. Each image in these datasets is described by the dense SIFT features and is represented by a 1000-dimensional vector. The features and labels of the video dataset MediaMill are downloaded from the ‘Mulan’ website 2.

Semantic hierarchy. We build a semantic hierarchy for each dataset based on Wordnet [14]. Specifically, for each dataset, we search for each class in Wordnet and extract one or more directed paths (i.e., a long sequence of directed edges from parent class to child class). In each path, we identify the nearest class that is also in the label vocabulary (i.e., the set \( \{c_1, \cdots, c_m\} \) of all classes of this dataset) as the parent class. This procedure is repeated for all \( m \) classes in the dataset to form the semantic hierarchy matrix \( \Phi \). In the

---

1http://lear.inrialpes.fr/people/guillaumin/data.php
2http://mulan.sourceforge.net/datasets-mlc.html
same manner, we build the hierarchy for each of the four datasets. Similar to [30], we also consider two types of semantic dependency: “is a” and “is a part of”. For example, a part of the semantic hierarchy of Corel 5k is shown in Figure 2. Due to the space limit, we provide the complete semantic hierarchies and the complete label matrices for all four datasets in the supplementary material. A summary of these hierarchies are given in Table 1.

**Figure 2:** A part of the semantic hierarchy of Corel 5k

Table 1: Details of the semantic hierarchies for the different datasets that we augmented.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># nodes</th>
<th># edges</th>
<th># root</th>
<th># leaf</th>
<th># singleton</th>
<th>depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP Game [12]</td>
<td>268</td>
<td>129</td>
<td>41</td>
<td>92</td>
<td>120</td>
<td>4</td>
</tr>
<tr>
<td>IAPRTC-12 [20]</td>
<td>291</td>
<td>179</td>
<td>36</td>
<td>132</td>
<td>98</td>
<td>4</td>
</tr>
<tr>
<td>MediaMill [28]</td>
<td>101</td>
<td>63</td>
<td>14</td>
<td>52</td>
<td>30</td>
<td>3</td>
</tr>
</tbody>
</table>

Note that in all the datasets, the provided ground-truth label matrices do not fully satisfy the semantic hierarchy constraints. In other words, some images or videos are labeled with a child class but not with the corresponding parent class. Therefore, we augment the label matrix according to the semantic hierarchy for each dataset. The semantically enhanced comprehensive ground-truth label matrix is referred to as “complete”, while the original label matrix as “original”. The basic statistics of both the complete and original label matrix are summarized in Table 2.

**Methods for comparison.** Several state-of-the-art and recent multi-label methods that can handle missing labels are used for comparison, including MLR-GL [5], MC-Pos [6], FastTag [9], MLML-exact and MLML-approx [37] and LEML [43]. MLR-GL and FastTag are specially developed for image annotation, while other methods are general machine learning methods. Also, a state-of-the-art method in hierarchical multi-label learning, CSSAG [2], is also evaluated. CSSAG is a decoding method based on the predicted label matrix of one another algorithm, i.e., the kernel dependency estimation (KDE) algorithm [36]. However, the KDE algorithm doesn’t work in the case of missing labels. To make a fair comparison between CSSAG and ML-MG, the results of ML-PGD are used as the input of CSSAG. The results are obtained with publicly available MATLAB source code of these methods provided by the authors. As a baseline, we also compare with a binary SVM classifier, which is trained on only labeled instances of each class.

**Evaluation metrics.** Average precision (AP) [47] is adopted to measure the ranking performance of the continuous label matrix $Z$ generated by each of the 8 methods. Top-5 accuracy is also used to measure the predicted discrete labels (the top 5 ranked labels of each instance are set as positive, while others are negative labels). To quantify the degree to which the semantic hierarchy constraints are violated, we adopt a simplified hierarchical Hamming loss, similar to [26],

$$\ell_H(Z_k, Y_C) = \frac{1}{nm} \sum_{i,j} \mathbb{I}[Z_k(i,j) = 1] \land (\hat{Z}_k(pa(i), j) = 0) \land (Y_C(pa(i), j) = 0),$$

where $\hat{Z}_k$ denotes the discrete label matrix generated by setting the top-$k$ labels in the continuous label vector of each instance as +1, while all others as 0. $Y_C$ denotes the average hierarchical loss (AHL) as $\hat{\ell}_H = \frac{1}{|S|} \sum_{k \in S} \ell_H^k$. We set $S = \{5, 10, 20, 50, 100, 150\}$ in our experiments.

**Other settings.** To simulate different scenarios with missing labels, we create training datasets with varying portions of provided labels, ranging from 5% (i.e., 95% of the whole training label matrix is missing) to 100% (i.e., no missing labels). In each case, the missing labels are randomly chosen among leaf and singleton classes in the semantic hierarchy, and are set to $\frac{1}{2}$ in the original label matrix of the training data. We repeat this process 5 times to obtain different data splits. In all cases, the experimental results of test data are computed based on the complete label matrix. The reported results are summarized as the mean and standard deviation over all the runs. The average runtime of each method on the different datasets is also reported. Because the authors of [43] only provide the MEX file for LEML under Ubuntu system, we run it on a workstation with Ubuntu 12.04.2 and Intel Xeon X5650 2.67 GHz CPU. All other methods are run on the same machine with Windows 7. The trade-off parameters $\beta$ and $\gamma$ are tuned by cross-validation. The tuning ranges are set as $\beta \in \{0.1, 1, 5, 10, 50\}$ and $\gamma \in \{0, 0.01, 0.1, 1, 10\}$. Both $W_X$ and $W_C$ are defined

3Trained with the LIBSVM package [7].

4If missing labels are generated on root and intermediate classes, many of them can be directly inferred as positive labels if their children classes are positive. Consequently the true missing label proportion may be inconsistent with the original designed proportion. To avoid this variation, we only generate missing labels on leaf and singleton classes.
Table 2: Data statistics of features and label matrices of four benchmark datasets.

<table>
<thead>
<tr>
<th>dataset</th>
<th># instances (training, test)</th>
<th># class</th>
<th># feature</th>
<th>(k_X, k_C)</th>
<th>label matrix</th>
<th>avg pos-class/inst</th>
<th>avg inst/pos-class</th>
<th>pos-class rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corel 5k [13]</td>
<td>4999 = 4500 + 499</td>
<td>260</td>
<td>1000</td>
<td>20, 10, 100</td>
<td>original</td>
<td>3.40</td>
<td>65.30</td>
<td>1.31%</td>
</tr>
<tr>
<td>ESP Game [32]</td>
<td>20770 = 18689 + 2081</td>
<td>268</td>
<td>1000</td>
<td>20, 10, 100</td>
<td>complete</td>
<td>4.84</td>
<td>93.06</td>
<td>1.86%</td>
</tr>
<tr>
<td>IAPRTC-12 [20]</td>
<td>19627 = 17665 + 1962</td>
<td>291</td>
<td>1000</td>
<td>20, 10, 100</td>
<td>original</td>
<td>4.69</td>
<td>363.2</td>
<td>1.75%</td>
</tr>
<tr>
<td>MediaMill [28]</td>
<td>43907 = 30993 + 12914</td>
<td>101</td>
<td>120</td>
<td>20, 10, 100</td>
<td>complete</td>
<td>9.88</td>
<td>666.3</td>
<td>3.39%</td>
</tr>
</tbody>
</table>

Figure 3: Average precision (top) and top-5 accuracy (bottom) results of four benchmark datasets for methods without semantic hierarchy. The bar on each point indicates the corresponding standard deviation. Figure better viewed on screen.

5.2. Results without semantic hierarchy

Figure 3 shows AP and top-5 accuracy results of the eight methods when the semantic hierarchy is not used, i.e., when \(\Phi = 0\). In this case, the inequality constraints in ML-MG are degenerate. Then the proposed model is a convex QP with box constraints that is solvable using the PGD method. We denote the hierarchy-free version of ML-MG as ML-PGD. ML-PGD consistently outperforms the other methods, thus, showing its superiority over other MLML methods even without hierarchy information. The improvement over the most competitive method on the four datasets is usually 5% (AP) or 3% (accuracy). Compared with MLML-exact and MLML-approx, ML-PGD shows significant improvement, especially when large proportions of the labels are missing. This is due to two main reasons. Firstly, there are many noisy negative labels in the original training label matrix, i.e., some positive labels 1 are set to 0. Since a larger penalty is incurred when misclassifying a positive label in ML-PGD, the influence of noisy negative labels can be alleviated. However, this is not the case for both MLML-exact and MLML-approx. Secondly, ML-PGD does not give any bias to missing labels. In contrast, missing labels are encouraged to be intermediate values between negative and positive labels in MLML-exact and MLML-approx, which brings in label bias. This is why their performance decreases significantly as the missing proportion increases. Interestingly, most of the multi-label methods are outperformed by the binary SVM in most cases. This suggests that they are sensitive to noisy and missing labels. Also, we note that some of the baseline methods fail to produce results in some test cases, especially when the missing label proportion is high. For example, SVM fails when there are no positive instances for some classes. Because of its extremely slow runtime (> 10^4 seconds per iteration), it was infeasible to run MLR-GL on the last three datasets. Similarly, the high memory requirements of MLML-exact and MLML-approx preclude running them on MediaMill data.

5.3. Results with semantic hierarchy

The results of utilizing the semantic hierarchy are shown in Figure 4, where the AP results of ML-PGD are repeated to facilitate the comparison.
ML-PGD vs. ML-MG. In the case of 0% missing labels, the improvement of ML-MG over ML-PGD ranges from 10% − 19% across the four datasets. This suggests that the semantic hierarchy constraints provide very useful information to correct the noisy negative labels. Also, ML-MG significantly outperforms ML-PGD at high missing label proportions. This lends evidence to the fact that instance-level similarity and class co-occurrence are not enough to regularize the MLML problem and that semantic hierarchy provides essential information to fill in missing labels.

CSSAG vs. ML-MG. We take the continuous labels generated by ML-PGD as the input to CSSAG, which will one of two processing choices: one is to change the input continuous label score to 0 according to the semantic hierarchy and the predefined number of positive labels, while the other is to keep it unchanged. Thus, the AP results of its continuous outputs are similar to those of its input label matrix, i.e., ML-PGD. Although CSSAG can ensure that there are no inconsistent labels in its discrete label matrix (setting all positive continuous labels as discrete positive labels), it cannot provide a consistent continuous label ranking. In contrast, ML-MG can satisfy these two conditions simultaneously.

Qualitative Results. Figure 5 shows labels predicted by our proposed ML-MG method when it is applied to the task of image annotation. It is worthwhile to note that parent classes are always ranked ahead of their children classes, thus, visualizing the direct effect of semantic hierarchy. Due to the space limit, more qualitative examples are provided in the supplementary material.

5.4. Convergence and sensitivity to initialization

Here, we evaluate the convergence of ML-MG using different label initializations. We only present the curve in the case of 0% missing labels, as shown in Figure 6. In the top row, we initialize the label matrix Z by setting all missing labels as 0. In this case, ML-MG converges to its best AP in less than 30 iterations on Corel 5k and in less than 10 on the other datasets. In the bottom row, missing labels are initialized as random values in [0, 1]. We repeat the random initializations 10 times and report mean and std values. The extremely small std values of both objective function and AP values suggest ML-MG is insensitive to initialization.

5.5. Computational analysis

The computational complexity of ML-PGD is $O_{PGD} = O(T_1((2k_X + k_C)mn + 3k_C m^2 + 8m^2 n))$, while that of ML-MG is $O_{MG} = O(T_2(O_{PGD} + 2n mn)).$ $T_1$ and $T_2$ denote the number of PGD and ADMM iterations respectively. As shown in Figure 6, $T_2$ is always very small. The detailed derivation of the complexity is provided in the supplement.
To emphasize the computational efficiency of our method, we report the average runtime of all methods on the four datasets in Table 3. In the case of 0% missing labels, each method is run 10 times and the average runtime is recorded. For MLR-GL, LEML and ML-MG, the number of maximum iterations is set as 20, while 50 for ML-PGD. The ranks of the mapping matrix in LEML are set as 50, 50, 50, and 20 for the four datasets, respectively. Clearly, our proposed ML-MG method (and its hierarchy-free variant ML-PGD) are significantly more computationally attractive than the other methods.

Table 3: Runtime in seconds of all 9 methods

6. Conclusions and discussions

This work proposes a novel model to handle the problem of multi-label learning with missing labels. A unified network of label dependency is built based on a mixed graph, which jointly incorporates instance-level label similarity and class co-occurrence as undirected edges, as well as, semantic hierarchy as directed edges. A convex problem is formulated by encoding the undirected edges as regularization terms, while embedding the directed edges as linear constraints. Thus multi-label learning and enforcing semantic hierarchy constraints can be performed simultaneously. A computationally attractive algorithm based on ADMM is used to exactly optimize this problem. Applying our method on image and video annotation tasks have demonstrated its superior performance against state-of-the-art methods. Moreover, we contribute manually generated semantic hierarchies for four popular benchmark datasets, which will be beneficial to the research community at large.

We realize that semantic hierarchy and missing labels exist in many other real-world problems, including text classification [26, 27], protein function prediction [2, 42], tracking [38, 46] and action recognition [48, 49]. In the future, we aim to apply ML-MG to these applications.

Acknowledgments: This work is supported by competitive research funding from King Abdullah University of Science and Technology (KAUST). The participation of Siwei Lyu in this work is partly supported by US National Science Foundation Research Grant (CCF-1319800) and National Science Foundation Early Faculty Career Development (CAREER) Award (IIS-0953373). We thank Fabian Caba Heilbron for his help on figure plotting, and Rafal Protasiuk for his help on data collection. We thank the reviewers for their constructive comments.

References

ML-MG: Multi-label Learning with Missing Labels Using a Mixed Graph (Supplementary Material)

Baoyuan Wu  
KAUST  
Saudi Arabia  
baoyuan.wu@kaust.edu.sa

Siwei Lyu  
University at Albany, SUNY  
Albany, NY, USA  
slyu@albany.edu

Bernard Ghanem  
KAUST  
Saudi Arabia  
bernard.ghanem@kaust.edu.sa

1. Convexity proof

For clarity and continuity, we rewrite the continuous optimization problem here,

$$\min_{Z} f_1(Z) = -\text{tr}(Y^TZ) + \beta \text{tr}(ZLXZ^\top) + \gamma \text{tr}(Z^\top LCZ),$$

s.t. $Z \in [0, 1]^{m \times n}, \quad \Phi^\top Z \geq 0. \tag{1}$

Firstly we introduce the following vector variables:

$$z = \text{vec}(Z) = [Z_{11}, \ldots, Z_{m_1}, \ldots, Z_{mn}]^\top \in \{-1, +1\}^{mn \times 1},$$

$$y = \text{vec}(Y) = [Y_{11}, \ldots, Y_{m_1}, \ldots, Y_{mn}]^\top \in \{-1, 0, +1\}^{mn \times 1},$$

$$W = \beta \cdot W_X \otimes I_m + \gamma \cdot I_n \otimes W_C \in \mathbb{R}^{mn \times mn},$$

$$L = \beta \cdot L_X \otimes I_m + \gamma \cdot I_n \otimes L_C \in \mathbb{R}^{mn \times mn},$$

$$\Phi = \Phi \otimes I_n,$$

where $\otimes$ indicates the Kronecker product [6]. Then Problem (1) can be transformed to its equivalent vector based formulation, as follows:

$$\begin{align*}
\arg \min_{z} & \quad f_2(z) = -y^\top z + z^\top Lz, \\
\text{s.t.} & \quad z \in [0, 1]^{mn \times 1}, \quad \Phi^\top z \geq 0. \quad \tag{2}
\end{align*}$$

**Lemma 1.** $L$ is positive semi-definite (PSD).

**Proof.** Given two square matrix $A \in \mathbb{R}^{n_1 \times n_1}$ and $B \in \mathbb{R}^{n_2 \times n_2}$, their eigenvalues are denoted as $\lambda_1, \ldots, \lambda_{n_1}$ and $\mu_1, \ldots, \mu_{n_2}$. According to the property of Kronecker product, the eigenvalues of $A \otimes B$ are $\lambda_i \mu_j$, $i = 1, \ldots, n_1; j = 1, \ldots, n_2$. $L_X$ is PSD and $I_m$ is positive definite (PD). Obviously all eigenvalues of $L_X \otimes I_m$ are non-zero values, so $L_X \otimes I_m$ is a PSD matrix. Similarly we can obtain that $I_n \otimes L_C$ is also PSD. Finally, as $L$ is the positive weighted linear combination of two PSD matrices, it is easy to conclude that $L$ is a PSD matrix. 

**Proposition 1.** Problem (1) is convex.

**Proof.** The Hessian of the objective function in (2) w.r.t. $z$ is $L$, which has been proven to be a PSD matrix in Lemma 1. Thus the objective function (2) is a convex function in $z$. The box and linear inequality constraints lead to a convex feasible solution space (that satisfies Slater’s condition), so it is easy to conclude that Problem (2) is a convex optimization problem. Finally, as Problem (1) and (2) are equivalent, thus Problem (1) is a convex problem. 

2. Optimization details of ADMM for ML-MG

The augmented Lagrange function of Problem (1) is defined as follows:

$$L_\rho(Z, Q, \Lambda) = f_1(Z) + \text{tr}[\Lambda^\top (\Phi^\top Z - Q)] + \frac{\rho}{2} \|\Phi^\top Z - Q\|^2_F, \quad \tag{3}$$

where $Z \in [0, 1]^{mn \times n}$ and $Q \succeq 0$, and $\Lambda \in \mathbb{R}^{n \times n}$ represents the Lagrange multiplier (dual variable), $\rho > 0$ is a tradeoff parameter. $\| \cdot \|_F$ denotes the Frobenius norm. Following the conventional ADMM framework [1], we minimize Problem (1) by the following alternating updates.

$$Z_{t+1} = \arg \min_{Z \in [0, 1]} L_\rho(Z, Q_t, \Lambda_t) = \arg \min_{Z \in [0, 1]} \left[ f_1(Z) + \text{tr}[\Lambda_t^\top (\Phi^\top Z - Q_t)] + \frac{\rho}{2} \|\Phi^\top Z - Q_t\|^2_F \right] \quad \tag{4}$$

where $\Lambda_t = -\nabla Z + \Phi A_t - \rho \Phi Q_t, B_t = \beta L_X$ and $C_t = \gamma L_C + \frac{\rho}{2} \Phi \Phi^\top$. Clearly, $\Phi \Phi^\top$ is PSD, so $C_t$ is also positive semi-definite. According to Proposition 1, Problem (4) is a convex quadratic programming (QP) problem with
Experiments, the computational cost can be significantly re-
coldest the number of neighbors at the instance-level, while

\( O \) denotes the number of neighbors at the class-level

Consequently, the computational complexity of ML-MG is 

Thus we have

Projected gradient descent. The gradient of the objective function (4) w.r.t \( Z \), and the step size are computed as

\[
\nabla Z_t = \Lambda_t + 2Z_tB + 2C_tZ_t, \\
\alpha_t = \arg \min_{\alpha \geq 0} L_\rho(Z - \alpha \nabla Z_t, Q, \Lambda_t)
\]

\[
= \frac{1}{2} \text{tr}(\Lambda_t^\top \nabla Z_t) + \text{tr}(Z_tB_t \nabla Z_t^\top) + \text{tr}(\nabla Z_t^\top C_tZ_t)
\]

where \( t \) denotes the iteration index of PGD. Then \( Z \) is updated as follows:

\[
Z_{t+1} = \min(1, \max(0, Z_t - \alpha_t \nabla Z_t)).
\]

As Problem (4) is convex, PGD is guaranteed to converge to the global optimal solution. However, to reduce computational cost, we step this update step only after a few PGD iterations. This heuristic makes the overall convergence of the original optimization problem much faster, without any considerable effect on performance.

3. Complexity and implementation details

Complexity. Firstly, we analyze the computational complexity of the ML-PGD algorithm. When implementing ML-PGD, the following observations can help to accelerate the code. First, both \( L_X \) and \( L_C \) are sparse, and there are only \( nkX \) and \( mkC \) non-zero entries respectively. \( kX \ll n \) denotes the number of neighbors at the instance-level, while \( kC \ll m \) is the number of neighbors at the class-level (their specific values on different datasets are shown in Table 2 of the main manuscript). Second, there are some shared terms between different steps, such as \( ZB_k \) and \( C_kZ \). Third, it is known that \( \text{tr}(abc) = \text{tr}(bca) = \text{tr}(cab) \). Thus we have \( \text{tr}(A_k^\top Z) = \text{tr}(ZA_k^\top) \) or \( \text{tr}(Z^\top C_kZ) = \text{tr}(ZZ^\top C_k) \). Considering \( m \ll n \) always holds in our experiments, the computational cost can be significantly reduced from \( O(mn^2) \) to \( O(mn^2) \), or from \( O(mn^2 + m^2n) \) to \( O(m^2n + m^3) \). Utilizing the above three points, the actual computational complexity of ML-PGD is \( O_{MG} = O(T_2(O_1 + 2n_mmn)) \), with \( T_2 \) being the number of iterations of the ADMM algorithm. As shown in Section 5.4 of the main manuscript, \( T_2 \) is small in the majority of cases (\( T_2 \leq 20 \)). Besides, when calling the PGD algorithm in ADMM, \( T_3 \) is set to a small value (5) and in which case the ADMM algorithm always achieves very good performance in a few iterations. In terms of space complexity, since \( \mathbf{L}_X \) and \( \mathbf{L}_C \) are sparse, the largest size of all matrices is only \( mn \). So, the overall space complexity is \( O(nkX + mkC + 11mn + 5m^2 + n_d(2n + n_e)) \).

Graph construction. In our experiments, \( W_C \) represents the co-occurrence between all pairs of classes, while \( W_X \) is efficiently computed using the kd-tree technique [4].

An acceleration heuristic. The computation of the step size \( \alpha_t \) (see Equation (8)) takes about 50% of the running time in each iteration. However, we observe that the step size in consecutive iterations tends to be very close. Thus, we only compute the step size \( \alpha_t \) once in every 5 iterations, during which the step size is multiplied by a damping factor (0.9 in our experiments) with the last computed step size. Compared to the case where the step size is computed exactly in each iteration, the runtime is significantly reduced to about 40% (this value depends on \( (m, n) \)) with a negligible change in prediction performance.

4. Additional evaluation results

In Section 5 of the main manuscript, the presented experimental results are obtained with the noisy (or inconsistent) initial label matrix. Specifically, in the initial label matrix, if the child class is positive (+1), but its parent class is negative (0) or missing (\( \frac{1}{2} \)), then such parent labels are denoted as noisy labels. The results presented in Section 5 have demonstrated ML-MG is robust to noisy labels, while the compared methods are significantly affected by them.

To further explore the performance of different methods, we present additional results in the case when these noisy labels are removed a priori. Specifically, the semantic hierarchy is used to fill noisy labels (i.e., if the child class is positive but any of its parent classes (first and higher order parents) are missing or negative, then fill positive labels to parents). Considering \( m \ll n \) always holds in our experiments, the computational cost can be significantly reduced from \( O(mn^2 + m^2n) \) to \( O(m^2n + m^3) \). Utilizing the above three points, the actual computational complexity of ML-PGD is \( O_{MG} = O(T_2(O_1 + 2n_mmn)) \), with \( T_2 \) being the number of iterations of the ADMM algorithm. As shown in Section 5.4 of the main manuscript, \( T_2 \) is small in the majority of cases (\( T_2 \leq 20 \)). Besides, when calling the PGD algorithm in ADMM, \( T_3 \) is set to a small value (5) and in which case the ADMM algorithm always achieves very good performance in a few iterations. In terms of space complexity, since \( \mathbf{L}_X \) and \( \mathbf{L}_C \) are sparse, the largest size of all matrices is only \( mn \). So, the overall space complexity is \( O(nkX + mkC + 11mn + 5m^2 + n_d(2n + n_e)) \).

Different usages of semantic hierarchy in our methods are also researched. For example, if the initial label matrix
Figure 1: Average precision (AP) results of four benchmark datasets for different methods with filled-in labels by semantic hierarchy. The bar on each point indicates the corresponding standard deviation (std). Figure better viewed on screen.

Figure 2: Average precision (AP) and average hierarchical loss (AHL) results of our methods with semantic hierarchy.

is filled using this semantic hierarchy, then the corresponding method is denoted as “Filling + xxx”. Evaluation results of different variants of our method are shown in Fig 2.

**ML-PGD vs. Filling + ML-PGD.** Filling + ML-PGD shows significant improvement over ML-PGD for both the AP and AHL criteria. This is primarily due to the fact that the semantic hierarchy provides additional positive labels for Filling + ML-PGD and that the AP measure focuses on these positive labels. As shown in Table 2, the positive class rate is very low in all datasets, which means positive labels are rare and valuable. Thus, the additional positive labels have a large impact to the overall AP performance of the method. As the positive labels are added on parent classes, there are more positive parent labels than positive child labels in the training label matrix. Thus the possibility of inconsistent labels will decrease in the final predicted label matrix, leading to the decrease of AHL values. When higher positive class rates exist, we expect that this improvement in performance to be less considerable.

**ML-PGD vs. ML-MG.** The difference between these two methods is that the semantic hierarchy is used in the optimization process of ML-MG. Consequently, there are significant improvements in both AP and AHL. Here, we note that the AHL results of ML-MG are always 0, which means semantic constraints are fully satisfied by ML-MG. This demonstrates the effectiveness of our method in combining semantic hierarchy and multi-label learning together.

**Filling + ML-PGD vs. ML-MG.** They represent two different usages of semantic hierarchy. The former uses semantic hierarchy and multi-label learning separately, while the latter exploits them together jointly. Although AP results of both methods are close to each other in most cases, AHL results reveal the significant difference in performance between them. ML-MG can not only remedy the information insufficiency of the noisy initial label matrix, but it also
ensures that the final labels are consistent with the semantic hierarchy. In contrast, although Filling + ML-PGD is trained with the filled-in label matrix, the final labels are not guaranteed to be consistent. Thus, we conclude that ML-MG is a better choice to utilize semantic hierarchy.

**Filling+ML-MG vs. others.** Filling + ML-MG thoroughly utilizes semantic hierarchy in different states, including preprocessing and optimization. It is not surprising that it produces the best performance in all cases, with an improvement of 1% to 4% over ML-MG and Filling + ML-PGD. Interestingly, this improvement increases as the missing label proportion in the data increases. This is primarily due to the fact that positive labels become more rare and valuable when the missing labels are significantly more. Thus, the additional positive labels filled by the semantic hierarchy have a larger impact.

**Summary.** All the above comparisons demonstrate the following two points.

1. Semantic hierarchy provides useful information for multi-label prediction.
2. Our proposed ML-MG method takes full advantage of this semantic hierarchy by outputting hierarchy consistent labels, which are good predictions of the ground truth labels.

**5. Additional prediction examples**

Additional prediction examples from the ESP Game and IAPRTC-12 datasets using ML-MG are shown in Fig 3 and 4 respectively. As we do not have the images of Corel 5k and MediaMill, their prediction examples are not presented. Here, we note that semantic hierarchy is fully satisfied in all examples. Specifically, the parent class is always ranked before its child classes.

**References**

Figure 3: Some prediction results of our ML-MG method. The images are extracted from ESP Game [5]. The predicted labels are ranked in descending order according to their label scores.

Figure 4: Some prediction results of our ML-MG method. The images are extracted from IAPRTC-12 [3]. The predicted labels are ranked in descending order according to their label scores.