Multi-label learning with missing labels for image annotation and facial action unit recognition

Baoyuan Wu a, Siwei Lyub, Bao-Gang Hua, Qiang Jic,∗

a Department of Computer Science, University at Albany, SUNY, NY 12222, USA
b National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, PR China
c Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA

Article history:
Received 4 March 2014
Received in revised form 29 November 2014
Accepted 26 January 2015
Available online 12 February 2015

Keywords:
Multi-label learning
Missing labels
Image annotation
Facial action unit recognition

Abstract
Many problems in computer vision, such as image annotation, can be formulated as multi-label learning problems. It is typically assumed that the complete label assignment for each training image is available. However, this is often not the case in practice, as many training images may only be annotated with a partial set of labels, either due to the intensive effort to obtain the fully labeled training set or the intrinsic ambiguities among the classes. In this work, we propose a method for multi-label learning that explicitly handles missing labels. We train classifiers with the multi-label with missing labels (MLML) learning framework by enforcing the consistency between the predicted labels and the provided labels as well as the local smoothness among the label assignments. Experiments on three benchmark data sets in image annotation and one benchmark data set in facial action unit recognition demonstrate the improved performance of our method in comparison of several state-of-the-art methods.

1. Introduction
In multi-label learning, each example can be associated with many classes simultaneously. The typical examples include image annotation [1] and the recognition of facial action units (AUs) in facial images [2]. More formally, the complete labels of an example \( x \) over \( m \) classes, \( \{c_1, \ldots, c_m\} \) are represented as a vector \( z \in \{-1, 1\}^m \), where a positive value indicates that the example belongs to the corresponding class, and a negative value indicates the opposite. The prediction from an example \( x \) to its complete label vector, \( z \), is the main task of multi-label learning [3].

The majority of previous multi-label learning methods assume that each example in the training set is associated with a complete label assignment. However, it is difficult to obtain a complete label assignment for each training example. For example, when the size of the candidate classes is large, such as in image annotation (see Fig. 1), it is costly to acquire the complete label vector for even one training image. Another possible reason of this difficulty is the ambiguity among classes, such as in AU recognition. AU is typically labeled by trained experts, which could be a time-consuming process. Furthermore, due to the ambiguity among AUs such as cheek raiser (AU6) vs. eye lid tightener (AU7) (see Fig. 2) as well as the poor image quality, some AUs are difficult to label confidently (the detailed definitions of all AUs can be found in [2]). A more realistic scenario is that each image is only provided with a partial label assignment, while the assignments with other classes are missing (see Fig. 1). To explicitly accommodate the missing labels, for each input image we introduce the definition of an incomplete label vector \( y \in \{-1, 0, 1\}^m \); a nonzero element has the same meaning as in the complete label vector \( z \), while a zero element corresponds to a label with no assigned value for this image, i.e., missing label.

The focus of this work is multi-label learning with missing labels (MLML) of which the primary objective is to derive a parametric multi-label classifier from input data example \( x \) and its incomplete label vector \( y \). Many previous multi-label learning methods that do handle missing labels usually do not make a clear distinction between missing labels and negative labels, i.e., the missing labels are assigned to the negative value by default [4–6]. However, such a simple treatment is often ungrounded in the practical applications. Consider the image annotation example in Fig. 1, label “home” is not shown in the label list of the image on the left, but we cannot simply conclude that the image should have a negative value for that label, because it has a positive value in the available label “house” that has strong semantic correlation with “home”.

We formulate MLML as an inductive learning problem, which consists of three terms, including the consistency between the predicted labels and the provided labels, the smoothness of label
assignments measured with similarities between examples and between classes, as well as a $\ell_2^1$ norm over the model parameter to avoid over-fitting. Our algorithm is obtained from the iteratively re-weighted least squares (IRWLS) [7] method. We demonstrate the superior performance of our method on two computer vision problems: one is the annotation of general images, and the other is the recognition of facial action unit (AU) on facial images. The main contributions of our work can be summarized as follows:

(i) we describe a new method to handle missing labels in multi-label learning;
(ii) we present an efficient numerical algorithm to learn the inductive classifier using the MLML framework;
(iii) we formulate image annotation and AU classification as multi-label learning with missing label problems, and show that our method outperforms several state-of-the-art works on three benchmark data sets.

2. Related works

In the literature of multi-label learning, some recent works have studied the problem with missing labels. The semi-supervised multi-label learning (SMSE2) [6] addresses the special case with examples either fully labeled or completely unlabeled. The weak label learning (WELL) method [4] focuses on the case where examples only have a partial set of positive labels available with the rest of the labels unassigned. Both SMSE2 and WELL assign the missing label values to zero by default, which coincides with the numerical value assigned to negative labels. As such, these methods implicitly assume that missing labels are equivalent to negative labels. This assumption is made explicitly in the work of multi-label learning with incomplete class assignment (MLR/GL) [5], where available labels all take positive values, and the missing labels are assigned to negative values, and thus becomes a fully labeled multi-label learning problem. However, as we have argued in Introduction, it is questionable if such an assumption always holds in actual data sets, and whenever it does not, treating missing label indiscriminately as negative labels introduce undesirable bias to the learning problem. The recent work of multi-label learning based on Bayesian compressed sensing (BML-CS) [10] can be used to solve the MLML problem, but it assumes a continuous probability model over the binary labels, and the resulting solution is based on a more costly MCMC algorithm. Moreover, the labels are assumed to be independently distributed in BML-CS, while the proposed model naturally incorporates the correlations between examples and between classes. In another related literature, i.e., matrix completion (MC), some recent works [11–13] have been proposed to handle the MLML problem. Their basic idea is concatenating the label matrix and feature matrix into a unified matrix, based on which the standard matrix completion techniques can be applied to fill in the missing labels. These works also avoid the label bias. However, the low rank assumption in MC implicitly implies class correlations. In contrast, our smoothness assumption explicitly captures such correlations, and we can also replace the smoothness to embed other types of correlations. Besides, the proposed efficient solution only involves matrix multiplication, while the solution to MC exploits the expensive SVD decomposition.

In the literature of image annotation, in addition to the single label classification [34–36] and scene classification [37,38], both MIML and MLML consider multi-label problems, but there exist two significant differences between them. For clarity, we take the image classification as the example. Firstly, their tasks are different. MIML aims to predict both image-level and region-level labels, while MLML only focuses on image-level label prediction. Secondly, in terms of predicting the image-level labels, MIML can be seen as supervised learning, while MLML can handle the missing labels.

3. Multi-label learning with missing labels

We start with a general setting of the MLML problem. We assume that the data are represented as matrix $X = (x_1, \ldots, x_n)$, where each
example is a d-dimensional column vector. Each example can be associated with m different classes \( \{c_1, \ldots, c_m\} \), and we are also provided with an available label matrix \( Y = (y_1, \ldots, y_n) \) with \( y_i \in \{\pm 1, 0\} \). The definition of \( Y \) allows for the inclusion of completely labeled examples, i.e., \( y_i \in \{\pm 1\} \), or unlabeled examples, i.e., \( y_i \in \{0\} \), or examples with only positive or negative labels, i.e., \( y_i \in \{\pm 1\} \) or \( \{0\} \), respectively.

Our objective is learning a mapping function based on the data X and the provided labels Y, to predict the labels of unlabeled testing examples based on the label consistency and smoothness. Specifically, we want to learn the mapping function \( f_\theta(x_i) = \theta^T h(x_i) \), where \( \theta \in \mathbb{R}^d \) denotes the parameter corresponding to the class \( c_i \), \( h(x_i) \) is a general feature function that maps the original d-dimensional data to the \( d \)-dimensional feature. It will be specified in experiments. Given the mapping function, the predicted label \( Z_\theta \) can be determined through a sign function, i.e., \( Z_\theta = \text{sgn}(\theta^T h(x_i)) \): \( \text{sgn}(a) = 1 \) if \( a > 0 \); \( \text{sgn}(a) = -1 \) if \( a \leq 0 \).

To achieve this goal, we leverage two assumptions, the label consistency and the label smoothness. Firstly, we require \( Z_\theta = Y_\theta \) whenever \( Y_\theta \neq 0 \). In other words, predicted complete labels should also be consistent with the available label matrix whenever label \( l \) is provided for example \( x_i \). On the other hand, for \( Z_\theta \) where the label information is missing in the available label matrix, i.e., \( Y_\theta = 0 \), we will “fill” their values in using two smoothness assumptions:

(i) **Example-level smoothness**: If the features of two images \( x_i, x_j \) are similar, then their labels represented by the corresponding columns of matrix \( Z_\theta \) should also be close.

(ii) **Class-level smoothness**: If two classes \( c_k \) and \( c_l \) have close semantic meanings, then their instantiations in the overall data set \( X \), represented by the corresponding rows of matrix \( Z_\theta \) should also be similar.

We formulate the MLML problem as an optimization problem that maximizes the label smoothness and the label consistency simultaneously.

### 3.1. Label smoothness

**Example-level smoothness**: The example-level similarity is computed using a measure defined over each pair of examples and kept in a symmetric positive definite matrix \( V_c \), with \( V_c(i,j) = K_c(x_i, x_j) \). The similarity kernel function \( K_c(\cdot, \cdot) \) is defined as follows [6]:

\[
K_c(x_i, x_j) = \begin{cases} 
    e^{-\|x_i - x_j\|^2/\epsilon_i}, & i \neq j \\
    0, & i = j.
\end{cases}
\]

where \( \epsilon_i = \sqrt{1 - \|x_i\|^2} \) and \( x_i \) is the \( i \)-th nearest neighbor of \( x_i \). With the example similarity matrix, the example-level smoothness of the complete label matrix \( Z_\theta \) is measured by

\[
S_Y(Z) = \frac{1}{2} \sum_{k=1}^{m} \sum_{l=1}^{m} V_c(i,j) \left( \frac{Z_{lk} - Z_{lj}}{\sqrt{d_Y(i)}} - \frac{Z_{lk} - Z_{lj}}{\sqrt{d_Y(j)}} \right)^2,
\]

where the normalization term \( d_Y(i) = \sum_l V_c(i,l) \) makes \( S_Y(Z) \) invariant to the different scaling factors of the elements of \( V_c \) [40]. The above formula can be further simplified by the introduction of the normalized Laplacian matrix \( L_Y = I - D_Y^{-1/2} V_Y D_Y^{-1/2} \), with \( D_Y = \text{diag}(d_Y(1), \ldots, d_Y(m)) \), to

\[
S_Y(Z) = \text{tr}(Z_L Z_Y^T).
\]

**Class-level smoothness**: The similarities between classes are intrinsic to their semantic meanings. When such similarities are available from other sources, we can use them to construct the class similarity matrix \( V_C \), which embeds the semantic similarities between the classes \( C \). However, if such information is absent, we can exploit the co-occurrence relationships among classes, which can be computed based on the available label matrix \( Y \), as did in [6]. Specifically, we define an \( m \times m \) nonnegative matrix \( V_C \), with \( V_C(i,j) = K_C(\eta(i), \eta(j)) \), in which the kernel function \( K_C \) is defined as follows [6]:

\[
K_C(\eta(i), \eta(j)) = \begin{cases} 
    e^{-||Y_{ij} - Y_{lj}||/\epsilon}, & i \neq j \\
    0, & i = j.
\end{cases}
\]

where the kernel size \( \epsilon = \sqrt{1 - ||Y_{ij}||} / (||Y_{ij}|| \cdot ||Y_{lj}||) \). \( Y_{ij} \) is a sub-vector of \( Y_{ij} \), i.e., \( Y_{ij} = (Y_{ij}, Y_{i2}, \ldots, Y_{ii}) \), where \( l \) denotes the number of training examples, and \( Y_{ij} \) is the \( i \)-th row vector of \( Y \). Using the class similarity matrix, we measure the class-level smoothness of \( Z \) with

\[
S_C(Z) = \frac{1}{2} \sum_{k=1}^{m} \sum_{l=1}^{m} V_C(i,j) \left( \frac{Z_{ik} - Z_{lk}}{\sqrt{d_C(i)}} - \frac{Z_{ik} - Z_{lk}}{\sqrt{d_C(j)}} \right)^2,
\]

where \( d_C(i) = \sum_l V_C(i,l) \) normalizes the factor so that \( S_C(Z) \) is not affected by the different scaling factors of the elements of \( V_C \). Similar to the previous case, we simplify \( S_C(Z) \) by first introducing the normalized Laplacian matrix, i.e., \( L_C = I - D_C^{-1/2} V_C D_C^{-1/2} \) with diagonal matrix \( D_C = \text{diag}(d_C(1), \ldots, d_C(m)) \), and rewriting the above formula as [40]

\[
S_C(Z) = \text{tr}(Z_L Z_C).
\]

### 3.2. MLML objective function

Combining the two smoothness measures as well as the label consistency term, we formulate the MLML problem as the solution to the following optimization problem:

\[
\arg\max_{\theta} \sum_{i,j} \epsilon \theta(Y_{ij}, h(x_i)) + \beta \text{tr}(Z_L Z_Y^T) + \gamma \text{tr}(L_C Z).
\]

where the two pre-set parameters \( \beta \) and \( \gamma \) control the trade-off between the loss function and two smoothness terms, and can be tuned by cross validation.

We use the hinge loss function \( \epsilon \theta(Y_{ij}, h(x_i)) = \max(0, 1 - Y_{ij} \theta^T h(x_i)) \) to enforce the consistency between the predicted label and the provided label \( Y_\theta \). However, this loss function and the sign function in the definition of \( Z_\theta \) in the smoothness terms are non-differentiable, which makes the numerical optimization of \( \theta \) difficult. To alleviate this, we introduce two relaxations to the original problem (7). First the hinge loss function \( \epsilon \theta(Y_{ij}, h(x_i)) \) is approximated by a convex differentiable loss function, as follows:

\[
\epsilon \theta(Y_{ij}, h(x_i)) \approx -\log \left( \frac{1}{1 + \exp(-Y_{ij} \theta^T h(x_i))} \right).
\]

Second, the sign function \( \text{sgn}(\theta^T h(x_i)) \) is approximated as

\[
\text{sgn}(\theta^T h(x_i)) \approx 2(\theta^T h(x_i)) - 1 \in [-1, 1],
\]

where \( \sigma(a) = 1/(1 + \exp(-a)) \) denotes the sigmoid function. These two relaxations are illustrated in Fig. 3. Moreover, the \( L_2 \) norm with respect to \( \theta = (\theta_1, \ldots, \theta_m) \in \mathbb{R}^d \) is used to find sparse representations based on the features. Note that we do not require each individual feature coefficient to be sparse. Each \( \theta_i \) may still be dense, but for each data, only a small fraction of \( \theta \) will be relevant [41]. As a
result, our problem becomes learning $\Theta$ by minimizing
\[
\sum_{i} \sum_{j} \log (1 + e^{-(y_{ij}^{0} h(x_{ij}))}) + \beta \text{tr}(\hat{Z} L_{1} \hat{Z}^T) + \gamma \text{tr}(\hat{Z} L_{2} \hat{Z}) + \eta \| \Theta \|_{2,1},
\]
(10)
where $\hat{Z} = [2\sigma(\Theta^{T} h(X)) - 1] \in [-1, 1]^{m \times n}$ and $h(X) = (h(x_1), \ldots, h(x_n)) 
\in \mathbb{R}^{d \times n}$. The $\ell_{2,1}$ norm is defined as $\| \Theta \|_{2,1} = \sum_{j=1}^{d} \sqrt{\sum_{i=1}^{m} \theta_{ij}^{2}}$, where $\theta_{ij}^{2}$ denotes the $j$-th entry of $\Theta_i$.

3.3. Learning

We solve (10) with the iteratively re-weighted least squares (IRWLS) method to handle the non-differentiable $\ell_{2,1}$ term [7,41]. Specifically, we use the fact $|a| = \min_{u} \sigma_{j}(u^{2}/u + u)$ to replace the $\ell_{2,1}$ term as
\[
\arg \min_{\Theta, u \geq 0} J_{\Theta}(X) + \eta \frac{d}{2} \sum_{j=1}^{d} \left( \sum_{i=1}^{m} \theta_{ij}^{2}(u_{j} - u) \right),
\]
(11)
where $\Theta = (u_{1}, \ldots, u_{d}) \in \mathbb{R}^{d \times 1}$, and $J_{\Theta}(X)$ denotes the first three terms in (10). Eq. (11) is solved with coordinate descent, by iteratively minimizing each of the unknown variables, i.e., $\Theta$ and $u$, until convergence.

The optimal solution to $u$ is thus $u^{*} = \sqrt{\sum_{i=1}^{m} \theta_{i}^{2}}(j) = 1, \ldots, d$. Given the fixed $u$, the parameters $\theta_{1}, \ldots, \theta_{m}$ can be learned sequentially. Specifically, to optimize $\theta_{i}$, we seek the solution to the following problem:
\[
\arg \min_{\theta_{i}} J_{\theta_{i}}(X) + \frac{u_{i}}{2} \text{diag}(u)^{-1} \theta_{i},
\]
(12)
which is solved by gradient descent. Specifically, the gradient of the above objective function w.r.t. $\theta_{i}$ is given by
\[
\nabla \theta_{i} = \eta \cdot \text{diag}(u)^{-1} \theta_{i} + 2 \sum_{j=1}^{n} h(x_{ij}) \cdot \sigma_{j}(1 - \sigma_{j}) \cdot \left( \frac{\partial g_{ij}}{\partial d_{ij}} \cdot \left( \frac{2r_{j} - 1}{\sqrt{d_{ij}}^{3}} \sum_{r=1}^{m} V_{c}(i, r) - \sum_{r=1}^{m} V_{c}(i, r)(2\sigma_{j} - 1) \right) \right) + 2 \gamma \left( \frac{\partial g_{ij}}{\partial d_{ij}} \cdot \left( \frac{2r_{j} - 1}{\sqrt{d_{ij}}^{3}} \sum_{r=1}^{m} V_{c}(i, r)(2\sigma_{j} - 1) \right) \right) - \sum_{j=1}^{n} \left( 1 - \sigma_{j} \right) \left( V_{c}(i, r) h(x_{ij}) \right) V_{c}(i, r) h(x_{ij}),
\]
where $\sigma_{j}$ is the shorthand notation for the sigmoid function $\sigma(\theta_{ij}^{T} h(x_{ij}))$. Then at the $t$-th iteration, $\theta_{i}$ is updated with gradient descent as follows:
\[
\theta_{i}^{t+1} = \theta_{i}^{t} - \alpha_{t} \nabla \theta_{i},
\]
(14)
where the step size $\alpha_{t}$ is determined by an optimal step size search based on the Armijo rule [42]. The detailed derivations of the above learning process can be found in 5.

4. Experiments

4.1. Data sets

We evaluate the proposed method on three benchmark data sets in image annotation and AU recognition. Specifically, the used data sets include

(i) Image annotation: The three widely used data sets, ESP Game [8], MIR Flickr [43] and NUS-WIDE-Lite [44], are adopted in our experiments. With a similar setting as [5], we delete the images of few positive labels and some rare classes. Specifically, in ESP Game, we delete the images with less than 5 positive labels and the classes that exist in less than 300 images. Then ESP Game consists of 10,457 images with 54 classes, including 9418 training and 1039 testing images. In MIR Flickr, images with less than 3 positive labels and classes that exist in less than 300 images are deleted. Then 10,199 images with 47 classes are remained, including 5137 training images and 5062 testing images. In NUS-WIDE-Lite images with less than 7 positive labels and classes that exist in less than 150 images are deleted, then 2456 images with 23 classes are remained. The positive label proportions in the whole ground-truth label matrix is 11.88%, 5.30% and 29.4% in ESP Game, MIR Flickr and NUS-WIDE-Lite respectively. In ESP Game and MIR Flickr, each image is represented by the RGB features in a 4096-dimensional vector [45]. In NUS-WIDE-Lite, each image is represented by a 265-dimensional vector, which consists of three types of features: 64-dimensional color histogram; 73-dimensional edge direction histogram; 128-dimensional wavelet texture.\(^3\) For the above three data sets, the Euclidean distance is used as the distance between images. The original dimension is reduced by PCA, in order to reduce the computational cost.

(ii) AU recognition: The benchmark data set in AU recognition, the Extended Cohn- Kanade (CK+) database [9] is adopted. 327 facial images with 16 most frequent AU classes (including AU 1, 2, 4, 5, 6, 7, 9, 12, 14, 15, 17, 20, 23, 24, 25, and 27) are chosen from the whole database. The positive label proportion in the whole ground-truth label matrix is 24.94%. Each facial image is described by a 201-dimensional column vector, which is

\(^{3}\)The features are downloaded from ‘http://lear.inrialpes.fr/people/guillaumin/data.php’.

\(^{4}\)The features are download from ‘http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm’.
cascaded by four types of features: 102-dimensional vector of the locations of 51 facial feature points; 40-dimensional texture features; 30-dimensional appearance features; 29-dimensional shape features.

The basic statistics of above data sets are summarized in Table 1. For NUS-WIDE-Lite and AU data, we randomly partition the whole data set to 5 uniform folds, and 4 folds are used as the training set while the remaining one is used as the testing set. Consequently we obtain 5 results. We further repeat this process 5 runs to obtain different partitions. Then we will obtain 25 results in total. Finally the mean and the standard deviation of the evaluation scores (the evaluation metrics will be specified later) of the 25 results are computed as the outputs. For all data sets, to simulate different scenarios with missing labels, we create training sets with varying portions of available labels, from 20% (i.e., 80% missing labels) to 100% (i.e., no missing labels). In each case, the missing labels are randomly chosen and removed from the ground-truth label matrix of the training data. We perform the experiments 5 runs to obtain different missing labels. In all cases, the experimental results of the testing sets are reported, which are summarized as the mean and standard deviation.

4.2. Experiment settings

Initializations: The initialization of the parameters Θ in the proposed method is performed as follows: for each single class, a logistic regression with the ℓ1 norm is trained (implemented by the built-in function “lassoglm” in Matlab software), then the learned Θi is used as the initialization of our model parameter. The feature function hj(xi) is simply specified as 1:xi in our experiments, then d′ = d + 1. Parameters β, γ and η in Eq. (10) are tuned by cross validation with the range [10−2, 100], [10−2, 102] and [10−2, 102] respectively.

Evaluation metrics: As the predicted continuous labels can be seen as a ranking list of each sample, we adopt two widely used performance metrics for multi-label ranking, namely, average precision (AP) [46] and area under ROC curve (AUC) [5]. Specifically, AP is calculated as follows [46]:

\[ \text{AP} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|S_i|} \sum_{s_i \in S_i} \left[ \delta \left( \text{rank}(x_i, s_i) < \text{rank}(x_i, s_j) \right) \right] \]

where \( S_i \) denotes the set of ground-truth positive classes of sample \( x_i \), and \( \text{rank}(x_i, s_i) \) represents the rank order of class \( s_i \) in the label ranking list of \( x_i \). AUC [5] measures the total area under ROC curve (true positive rate vs. false positive rate) of all classes. Specifically, given the label ranking list for each sample, the first \( k \) classes are treated as positive, while others are treated as negative. Through varying \( k \) from 1 to \( m \) (the number of classes), the ROC curve of all classes is gained. Higher values of both metrics represent better performances. Note that both metrics are the overall evaluations of the label ranking list, hence the score corresponding to each class is not calculated.

Comparisons: We compare the proposed method with several previous works on multi-label learning with missing labels and image annotation, including SMSE2 [6], MLR-GL [5] and MC-Pos [12]. We implement SMSE2 in Matlab and adopt the publicly available code MLR-GL. The code of MC-Pos is provided by its authors. We make our best effort to adjust the parameters in these methods as suggested in the original papers. We also include two baselines: one is learning the logistic regression with \( \ell_1 \) norm (it is also the initialization method of our algorithm, as mentioned above), and the other is learning a binary SVM classifier, for each single class independently. They both only use the labeled examples as the training set in each individual class, while the images with missing labels are ignored. Besides, we evaluate the influences of the example-level and class-level smoothness in the objective function (7), by setting \( \beta = 0 \) and \( \gamma = 0 \) respectively. They are denoted as \( \text{MLML}_{\beta = 0} \) and \( \text{MLML}_{\gamma = 0} \).

4.3. Results

Image annotation: The results of the image annotation task are shown in Tables 2–4. On ESP Game data, the proposed method MLML gives better results than other methods in most cases. On MIR Flickr data, the performance of MLML is better results than other methods in the cases of high label proportions, while is worse when the label proportion is low. Then possible reasons include when the label proportion is low, the computed class correlations based on concurrency are likely to be far from the ground-truth class correlations. Consequently the performance of MLML may be harmed by such a poor class correlation. Besides, compared with MC-Pos, the proposed inductive model may be more sensitive to the unbalanced label distribution problem. On NUS-WIDE-Lite data, the proposed method MLML also performs better than other methods in most cases. The comparisons between MLML and MLML_{\beta = 0} and MLML_{\gamma = 0} are also presented. On ESP Game, MLML_{\beta = 0} performs better than MLML in some cases. It demonstrates that the example-level smoothness does not provide the positive information to help MLML. The possible reason is that the Euclidean distance in \( K_{\gamma} \) is not the best choice to reveal the example-level correlations on this data. The performance is expected to be further improved by exploiting more sophisticated distances. On MIR Flickr, MLML does not show consistent superiority to MLML_{\beta = 0} and MLML_{\gamma = 0}. This demonstrates that the influences of example-level and class-level smoothness depend on label distribution and the provided label proportion. Moreover, the overall performance of all methods on these two data sets is not very high. It may be due to the simple image features and the image distance used in our experiments. However, it is fair enough to compare the performance of different methods. The comparisons on NUS-WIDE-Lite show that MLML_{\gamma = 0} performs better than MLML in the cases of low provided label proportions. The possible reason is when many missing labels exist, the computed kernel \( K_{\gamma} \) is far from the ground-truth semantic dependencies among classes. As such, it may harm the model performance. Furthermore, the symbol “−” in Tables 2, 3 and 5 indicates that the method does not work or is inapplicable in the corresponding case. Specifically, SMSE is developed for semi-supervised learning. Some annotation results are shown in Table 6.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Example</th>
<th>Class</th>
<th>Feature</th>
<th>Avg. posi.-class/example</th>
<th>Avg. posi.-example/class</th>
<th>Posi. label proportion (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP Game [8]</td>
<td>10,457</td>
<td>54</td>
<td>4096</td>
<td>6.41</td>
<td>1242.2</td>
<td>11.88</td>
</tr>
<tr>
<td>MIR Flicker [43]</td>
<td>10,199</td>
<td>47</td>
<td>4096</td>
<td>1.22</td>
<td>540.39</td>
<td>5.30</td>
</tr>
<tr>
<td>AU [9]</td>
<td>327</td>
<td>16</td>
<td>201</td>
<td>3.99</td>
<td>81.5</td>
<td>24.94</td>
</tr>
</tbody>
</table>
AU recognition: The results on the AU data are presented in Table 5. The proposed method MLML outperforms other methods in most cases. When the label proportion is 20%, the result of MLML is slightly lower than the result of SVM. The possible reason is the class-level smoothness (see Eq. (4)). When the label proportion is low, the computed similarities between classes may be inaccurate. This point is also proved by the comparison between MLML and MLML_γ = 0. It suggests that exploiting a more robust class-
level smoothness independent of the provided labels may further boost the performance of MLML. This will be explored in our future work. The predicted labels on some facial images are shown in Table 7.

### 4.4. Top-5 accuracy of each class

In above evaluations, the results are calculated based on the label ranking list of all classes. To evaluate the performance of each class, in the following we show the results evaluated by the top-5 accuracy, in the case of 100% label proportion. Specifically, for the label ranking list of each testing example, we assign the first five classes as positive classes, while all other classes as negative. Hence we obtain a discrete label matrix. Based on this matrix, we compute the average top-5 accuracy, which equals to the fraction of the correct prediction labels in ground-truth label matrix. The corresponding results of different algorithms are summarized in Table 8.

The top-5 accuracy of each class is also shown in Fig. 4. Note that all compared algorithms, including the proposed MLML algorithm, show poor performance on ESP Game and MIR Flickr data. We believe that the main reason is the extremely unbalanced label distribution, i.e., the positive labels are significantly less than the negative labels. As shown in Table 1, the positive label percentage are only 11.88% and 5.30% in ESP Game and MIR Flickr respectively. Actually, the unbalanced label distribution is a typical problem in multi-label learning. This motivates us to design more robust model, such as the weighted model. This will be explored in our future work.

### 4.5. Computational complexity and convergence

The computational complexity of the proposed algorithm and other compared algorithms are summarized in Table 9. The main computational cost of our algorithm is the computation of $\nabla \theta_i$ (see Eq. (13)), which only involves matrix multiplication. Its computational complexity is $O(mn(m+n+d))$. Moreover, by setting $V_C$ and $V_{\hat{C}}$ as sparse matrix, the complexity can be further reduced significantly. As $\theta_1, \ldots, \theta_m$ are learned sequentially, the overall complexity of the proposed algorithm is $O(mn(m+n+d))$. It is comparable with MLR-GL, and is much faster than other algorithms. However, when the number of candidate classes $m$ is large, the complexity of MLML will be high. This is why we only choose a part of candidate classes in above experiments. To alleviate this problem, the label space reduction can be exploited to be combined with our current model. This will be explored in our future work.

The convergence curves of the proposed algorithm on all data sets are shown in Fig. 5. Considering the number of classes in each data set, the proposed algorithm often converges in a small number of iterations. It guarantees the computational efficiency of our algorithm.

### 4.6. Discussions

According to the above results, find that in different proportions of missing labels, the performances of MLML seem to be inconsistent. Indeed in Section 1 we have demonstrated that
MLML method can avoid the label bias (i.e., treating missing labels as negative labels directly), which exists in some compared methods, such as SMSE2, WELL and MLR-GL. However, this advantage cannot always guarantee that when the amount of missing labels increases, the performance of MLML becomes better than other methods. We believe that the reason is the performance of MLML is not only influenced by the amount of missing labels, but is also influenced by other factors, including the percentage of positive labels in the ground-truth label matrix, as well as the initial parameters from binary logistic regression. Specifically, firstly when many missing labels exist, the computed class-level dependency based on the provided label vector (i.e., $K_2$, see Eq. (4)) may be far from the true semantic dependencies among classes. Secondly, when many missing labels exist, the binary logistic regression, which only exploits the labeled examples in each class, may not provide good initial parameters for MLML. These two points may harm the performance of MLML. Thirdly, if the positive label percentage is very low, such as the 8.83% in MIR Flickr, the label bias may not significantly harm the performance of other methods. Because treating missing as negative labels will bring in many correct negative labels, while the number of wrong labels (treating the ground-truth positive labels as negative labels) is small. As a result, the benefit from the correct labels may compensate the harm from the wrong labels.

Moreover, we also observe that in many cases the performance superiority of MLML is not very significant over the compared methods. We believe that there are three possible reasons. Firstly, MLML can be seen as an extension of linear logistic regression model. The linear model may limit the performance of MLML. It inspires us to make kernelization of MLML, which is expected to further enhance the performance. Secondly, the class-level dependency $K_2$ depends on the provided labels of the specific data. It inspires us to explore and exploit some other types of dependencies independent of the provided labels, which is expected to make the performance of MLML more robust. Last, the example-level dependency $K_1$ depends on the hand-crafted image features. In terms of image-based problems, visually similar images may not always lead to the closeness of labels. This semantic gap may harm the model performance. It inspires us to extract or learn more informative features to describe the semantic meaning of the image, based on saliency detection [48] or deep learning [49].

5. Conclusions and future work

In this paper, we introduce a new formulation and solution to multi-label learning with missing labels, and demonstrate its effectiveness on two computer vision applications, image annotation and AU recognition. Different from the existing methods, our model explicitly takes account of missing labels and systematically incorporates the label consistency and smoothness into the solution. Experiments on three benchmark data sets have verified the efficacy of the proposed method.

As future work, there are several directions we would like to further study. First, in our current work we only adopt the simple concurrency in the class-level smoothness and the simple Euclidean distance in example-level smoothness. Other types of class correlations and image distances have been explored in many existing works in the literature of multi-label learning and image

Table 9

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>$O(n^3)$</td>
<td>$O(mn^2)$</td>
<td>$O(mn^2 + n^3)$</td>
<td>$O(mn(m + n + d))$</td>
</tr>
</tbody>
</table>

**Fig. 4.** Top-5 accuracy of each class on the proposed algorithm on different data sets.
annotation respectively [50,51]. Combined with these works, the performance of our model is expected to be further improved. Second, we only use the linear logistic regression in our model. However, our model can be kernelized by setting different feature functions \( h(x_i) \). The kernelized model is expected to handle the data with more complex distributions. Third, as discussed in Section 4, the proposed MLML model still suffers from two problems: one is the scalability to the large scale class space, and the other is the unbalanced label distribution. They could be alleviated by label space reduction and the weighted model respectively. Fourth, sometimes the soft labels (labels with uncertainty) occur in particular scenarios, such as the AU intensity. Handling and predicting such soft labels can be seen as regression problems, such as [52]. As the discrete label \( Z_d \) is approximated by a continuous curve (see Eq. (9) and Fig. 3) in our learning process, our method can directly handle the provided soft labels and also output soft labels. As such it is a potential direction to extend our method to the multi-label regression problems. Last, we will also explore other applications of MLML, such as image denoising, image segmentation and scene classification.

**Conflict of interest**

None declared.

**Acknowledgments**

The work was mostly completed when the first author was a visiting student at Rensselaer Polytechnic Institute (RPI), supported by a scholarship from China Scholarship Council (CSC). We thank CSC and RPI for their support. Qiang Ji is supported in part by a grant from the US National Science Foundation (NSF, No. 1145152). Siwei Lyu is supported in part by the US National Science Foundation Research Grant (CCF-1319800) and the National Science Foundation Early Faculty Career Development (CAREER) Award (IIS-0953373). Bao-Gang Hu and Baoyuan Wu are supported in part by the National Natural Science Foundation of China (NSFC, No. 61273196).

**Appendix A: optimization of the MLML objective function**

For convenience, we firstly repeat the objective function (see Eq. (11) in Section 3.3 of the main paper) as follows:

\[
\min_{\theta \in \Theta} J(\theta) = \min_{\theta \in \Theta} \left[ \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} Y_{ij} \theta_{ij}^2 + \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{n}{r} V_c(i, r) \right]
\]

In more detail, the above objective function can be expanded as follows (in the following, we will use \( \sigma_{ij} \) as the shorthand notation for \( \sigma(\theta_{ij}^T h(x_i)) \)):

\[
\min_{\theta \in \Theta} \beta \sum_{i=1}^{m} \sum_{j=1}^{n} Y_{ij} \theta_{ij}^2 \left( \frac{2\sigma_{ij} - 1}{\sqrt{d_i(r)}} - \frac{2\sigma_{ij} - 1}{\sqrt{d_i(r)}} \right)^2 + \gamma \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{r=1}^{n} V_c(i, r) \times \left( \frac{2\sigma_{ij} - 1}{\sqrt{d_i(r)}} - \frac{2\sigma_{ij} - 1}{\sqrt{d_i(r)}} \right)^2 \sum_{i=1}^{m} \sum_{j=1}^{n} \log \sigma(Y_{ij} \theta_{ij}^T h(x_i)) \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{n}{r} \sigma_{ij}^2 + u_j \right],
\]

where \( \theta_{ij} \) denotes the \( j \)-th entry of \( \theta \). This problem is solved with coordinate descent, with iteratively updating \( \theta \) and \( u \), until convergence.
1. Given $\Theta$, learning $\mathbf{u}$

The optimal solution to $\mathbf{u}$ can be easily gained by setting the gradient of (17) w.r.t. $\mathbf{u}$ to be zero, as follows:

$$
-\frac{\sum_{i=1}^m \theta_i^2(j)}{u_j^*} + 0 = 0,
$$
with $u_j > 0 \implies u_j^* = \sqrt{\sum_{i=1}^m \theta_i^2(j)}.

(18)

2. Given $\mathbf{u}$, learning $\Theta$

Given the fixed $\mathbf{u}$, the last term of Eq. (17) becomes:

$$
\sum_{j=1}^d \left( \frac{\sum_{i=1}^m \theta_i^2(j)}{u_j^*} + u_j \right) = \sum_{i=1}^m \theta_i^2(j) - \sum_{j=1}^d u_j = \sum_{i=1}^m \theta_i^2(j) \text{diag}(\mathbf{u})^{-1} \theta_i + \sum_{j=1}^d u_j.
$$

Obviously this part becomes a weighted least squares. Furthermore, when all other parameters $\Theta, \theta$, and $\mathbf{u}$ are fixed, the objective function with respect to $\Theta$ becomes:

$$
\arg \min_\Theta \beta \sum_{j=1}^d V_j(i, r) \left( \frac{2\sigma_i - 1}{\sqrt{\text{det}(\mathbf{u})}} \right)^2 + \gamma \sum_{i=1}^m \sum_{j=1}^d V_i(i, r) \frac{1}{\sqrt{\text{det}(\mathbf{u})}} \theta_i^2 \text{diag}(\mathbf{u})^{-1} \theta_i.
$$

(19)

Next, denote the four terms in (19) as $A, B, C$, and $D$. Then their respective gradient w.r.t. $\Theta$ are as follows:

$$\frac{\partial A}{\partial \Theta} = 2\beta \sum_{j=1}^d V_j(i, r) \left( \frac{2\sigma_i - 1}{\sqrt{\text{det}(\mathbf{u})}} \right)^2 + \gamma \sum_{i=1}^m \sum_{j=1}^d V_i(i, r) \frac{1}{\sqrt{\text{det}(\mathbf{u})}} \theta_i^2 \text{diag}(\mathbf{u})^{-1} \theta_i.$$

$$\frac{\partial B}{\partial \Theta} = 2\beta \sum_{j=1}^d V_j(i, r) \left( \frac{2\sigma_i - 1}{\sqrt{\text{det}(\mathbf{u})}} \right)^2 + \gamma \sum_{i=1}^m \sum_{j=1}^d V_i(i, r) \frac{1}{\sqrt{\text{det}(\mathbf{u})}} \theta_i^2 \text{diag}(\mathbf{u})^{-1} \theta_i.$$

$$\frac{\partial C}{\partial \Theta} = - \sum_{j=1}^d \left( -\frac{1}{\sqrt{\text{det}(\mathbf{u})}} \text{diag}(\mathbf{u})^{-1} \theta_i \right).$$

Combining the above four terms, i.e., $\nabla \Theta = \frac{\partial A}{\partial \Theta} + \frac{\partial B}{\partial \Theta} + \frac{\partial C}{\partial \Theta} + \frac{\partial D}{\partial \Theta}$, we obtain

$$\nabla \Theta = \eta \cdot \text{diag}(\mathbf{u})^{-1} \theta_i + 2 \sum_{j=1}^d v_j(i, r) \cdot \left( \frac{2\sigma_i - 1}{\sqrt{\text{det}(\mathbf{u})}} \right)^2 + \gamma \sum_{i=1}^m \sum_{j=1}^d V_i(i, r) \frac{1}{\sqrt{\text{det}(\mathbf{u})}} \theta_i^2 \text{diag}(\mathbf{u})^{-1} \theta_i.$$

(20)

Using $\nabla \theta_i$ can be updated by gradient descent with a line search based on Armijo rule. Then $\theta_1, \ldots, \theta_n$ are learned sequentially.

References


