2014 22nd International Conference on Pattern Recognition

Multi-Label Learning with Missing Labels

Baoyuan Wu*, Zhilei Liu†, Shangfei Wang†, Bao-Gang Hu* and Qiang Ji‡
* National Laboratory of Pattern Recognition, CASIA, Beijing 100190, China
† University of Sciences and Technology of China, Anhui 230027, China
‡ Rensselaer Polytechnic Institute, NY 12180, USA
wubaoyuan1987@gmail.com, leivo@mail.ustc.edu.cn, sfwang@ustc.edu.cn, hubg@nlpr.ia.ac.cn, qji@ecse.rpi.edu

***Abstract*—In multi-label learning, each sample can be assigned to multiple class labels simultaneously. In this work, we focus on the problem of multi-label learning with missing labels (MLML), where instead of assuming a complete label assignment is provided for each sample, only partial labels are assigned with values, while the rest are *missing* or *not provided*. The positive (presence), negative (absence) and missing labels are explicitly distinguished in MLML. We formulate MLML as a transductive learning problem, where the goal is to recover the full label assignment for each sample by enforcing consistency with available label assignments and smoothness of label assignments. Along with an exact solution, we also provide an effective and efficient approximated solution. Our method shows much better performance than several state-of-the-art methods on several benchmark data sets.**

I. Introduction

In multi-label learning, one instance can be assigned to several categories simultaneously [1]. It has been successfully applied in many real problems, such as image annotation [2]–[4], semantic scene classification [5], and text categorization [6], [7]. However, in traditional multi-label learning, an important assumption is that the training instances are completely labeled. For example, three candidate categories c_1, c_2, c_3 are provided for all instances. One instance x_i is labeled as $(c_1, \neg c_2, c_3)$. It means that x_i is assigned to c_1 and c_3, not assigned to c_2. c_1 and c_3 are referred to as *positive labels* for x_i, while c_2 is referred to as *negative labels*. However, the completely labeled instances are not always available in real problems. There are two main reasons [2], including the large number of candidate categories and the ambiguity between categories. A typical sample is the annotation of facial action units (AUs) [8], which is an important problem in affective computing. AU should be labeled by trained experts [9], due to the ambiguity between AUs, such as cheek raiser (AU6) v.s lid tightener (AU7). In this case, it is difficult to provide the complete AU labels for one facial image (the detailed definitions of all 64 AUs can be found in [8]). More formally, if x_i is labeled as $(c_1, \neg c_2, ?c_3)$, it means there is no information about whether x_i is labeled as c_3 or not. c_3 is referred to as the *missing label* for x_i. x_i is called as *partially labeled sample*, in which missing labels exist. If x_i has only missing labels, then it is called as *completely unlabeled sample*. Our goal is to predict the complete label assignments of the unlabeled samples by exploiting the partially (including completely) labeled samples, referred to as *multi-label learning with missing labels* (MLML).

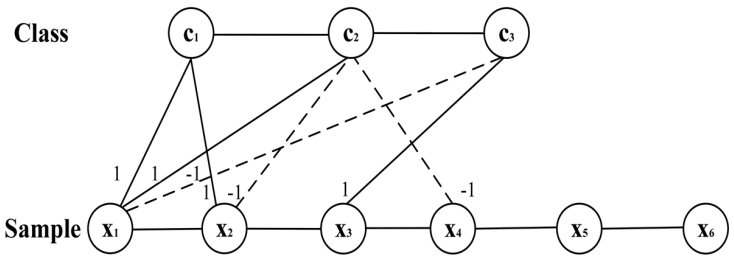
Many previous multi-label learning models that also handle the missing labels can be seen as the special case of MLML. The semi-supervised multi-label learning (SMSE2) [10] addresses the special case with samples either fully labeled or completely unlabeled. Both MLR-GL [2] and weak label learning (WELL) [3] consider the case that only a partial set of positive labels and missing labels are provided for each training sample. To handle the missing labels, a common solution adopted in above three models is treating the missing labels as negative labels, and then they become a fully labeled multi-label learning problem. This assumption is made based on the observation that most labels are negative labels for each sample. However, it is not always true, and will introduce undesirable bias to the original learning problem. A obvious bias is that some ground-truth positive labels are incorrectly set as negative labels. As we will show in the later experiments, such a bias may lead to poor and unstable performance.

We formulate the MLML problem through extending the SMSE2 [10], based on two assumptions of label consistency and label smoothness. Label consistency encourages the predicted label matrix to be consistent with the provided label matrix. Label smoothness is implemented on two levels: sample-level smoothness means two samples with similar features should have similar labels; class-level smoothness indicates that two semantically dependent classes should have similar instantiations. Although exploiting the same assumptions with SMSE2, we explicitly distinguish the negative and missing labels. As we will show in the later experiments, such a change will lead to significant performance improvement, because the bias of the initial label matrix vanishes in our formulation. Moreover, we not only use the exact solution based on solving the Sylvester equation, which is also used in SMSE2 [10], but also propose an efficient and effective approximated solution to the MLML problem.

The following three points highlight our contributions. (1) We present a general definition of the multi-label learning with missing labels, which can generalize several previous models. The significant change is that the positive, negative and missing labels are explicitly distinguished in MLML. (2) We present both the exact solution and an efficient and effective approximated solution to the MLML problem. (3) The efficacy of the proposed method is verified on three benchmark data sets compared with several related works.

The rest of this paper is organized as follows: Section II presents the formulation and solution of the multi-label learning with missing labels; Section III conducts numerical experiments on three benchmark data sets; Section IV concludes this paper.

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.343



The initial label matrix Y

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 1. A graphical illustration of multi-label learning with missing labels. The figure on the left consists of two layers: the bottom layer includes the sample nodes; the top layer contains the class nodes. The initial labels of samples are represented by the links between two layers: the solid link indicates a positive label; the dashed link denotes a negative label; no link means a missing label. The initial label matrix is presented on the right, where one column vector corresponds to one sample node and one row vector corresponds to one class node. The links between nodes in the bottom layer represent the similarities between different samples; the links between nodes in the top layer denote the semantic dependencies between the classes. Note that both layers are 2-D fields, and we just plot a chain for clarity.

II. Multi-label Learning with Incomplete Labels

A. Problem Formulation

In the data set $X = (x_1, \ldots, x_n)$, each sample $x_i \in \mathcal{R}^{d\times 1}$ can be associated to m classes $C = \{c_1, c_2, \ldots, c_m\}$ simultaneously. Specifically, the labels of x_i can be represented as a column vector $y_i \in \{1, 0, -1\}^{m\times 1}$. We define an initial label matrix $Y = (y_1, y_2, \ldots, y_n) \in \{1, 0, -1\}^{m\times n}$: the positive label $Y_{ij} = 1$ indicates x_j is labeled as c_i; the negative label $Y_{ij} = -1$ means x_j is not labeled as c_i; the missing label $Y_{ij} = 0$ denotes no information about whether x_j is labeled as c_i or not. If all entries of y_i are non-zero, then x_i is an fully labeled instance. If y_i contains zero and non-zero entries together, then x_i is a partially labeled sample. If all entries of y_i are zero, then x_i is a unlabeled sample. Our goal is to predict a complete label matrix $Z \in \{1, -1\}^{m\times n}$ by utilizing the initial label matrix Y, called as *multi-label learning with missing labels* (MLML). A brief illustration of MLML is presented in Figure 1. To achieve this goal, we exploit two assumptions [10] as follows:

- **label consistency**. The predicted label matrix Z should be consistent with the initial label matrix Y;
- **label smoothness**. The smoothness assumption is implemented on two levels: the sample-level smoothness means if the two samples x_i and x_j are similar, then their labels, i.e., the corresponding column vectors of Z should be similar; the class-level means if two classes c_i and c_j are semantically similar, then their instantiations, i.e., the corresponding row vectors of Z should be similar.

Based on above assumptions, we formulate the MLML problem as follows [10]:

$$\arg\min_{Z} \parallel Z - Y \parallel_{\mathcal{F}}^2 + \frac{\lambda_X}{2} tr(ZL_XZ^T) + \frac{\lambda_C}{2} tr(Z^T L_C Z), \tag{1}$$

where λ_X and λ_C are user-defined positive constants, which can be tuned by cross validation. Actually, in Section III-D, we will show that the proposed method is not very sensitive to these two values. $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm, while tr indicates the trace of matrix. The first term denotes the label consistency, while the last two terms represent the label smoothness. Specifically, we have

$$tr(ZL_XZ^T) = \sum_{k=1}^{m}\sum_{i,j}^{n} V_X(i,j)\left(\frac{Z_{ki}}{\sqrt{d_X(i)}} - \frac{Z_{kj}}{\sqrt{d_X(j)}}\right)^2,$$

where $L_X = I - D_X^{-\frac{1}{2}} V_X D_X^{-\frac{1}{2}}$ with the diagonal matrix $D_X = \text{diag}(d_X(1), \cdots, d_X(n))$. I denotes the identity matrix of the adaptive size, and hereafter we use the same symbol I to represent different sized identity matrices for clarity. The normalization term $d_X(i) = \sum_j^n V_X(i,j)$ makes the above smoothness term invariant to the different scaling factors of the elements of V_X [11]. V_X denotes the similarity matrix among samples, which will be specified later. Similarly, we also have

$$tr(Z^T L_C Z) = \sum_{k=1}^{n}\sum_{i,j}^{m} V_C(i,j)\left(\frac{Z_{ik}}{\sqrt{d_C(i)}} - \frac{Z_{jk}}{\sqrt{d_C(j)}}\right)^2,$$

where $L_C = I - D_C^{-\frac{1}{2}} V_C D_C^{-\frac{1}{2}}$ with the diagonal matrix $D_C = \text{diag}(d_C(1), \cdots, d_C(m))$. And $d_C(i) = \sum_j^m V_C(i,j)$ normalizes the factor so that the above smoothness term is not affected by the different scaling factors of the elements of V_C. V_C denotes the similarity matrix among classes, which will be specified later.

B. Similarity matrices

1) The sample similarity V_X: V_X includes all pairwise correlations among X. Here we utilize the affinity matrix, as follows:

$$V_X(i,j) = \exp\big(-d^2(x_i, x_j)/\sigma_i\sigma_j\big), \tag{2}$$

which is computed based on a k-nn graph, i.e., if x_j is not within the k-nearest neighbors of x_i, then $A_{ij} = 0$. We set $k = 20$ in experiments. Note that $V_X(i,i) = 0$. $d(x_i, x_j)$ denotes the distance between x_i and x_j (here the Euclidean distance is used). $\sigma_i = d(x_i, x_h)$, where x_h is the h-th nearest neighbor of x_i. Following the suggestion in [12], we set $h = 7$.

2) The class similarity V_C: V_C embeds the semantic correlations among the classes C. Some works have focused on developing the semantic correlations, such as subset constraints and exclusion constraints [4], [13]. Here we simply define a m-square weight matrix, as follows:

$$V_C(i,j) = \exp\big(-\eta[1 - \frac{\langle \overline{Y}_{\cdot i}, \overline{Y}_{\cdot j}\rangle}{\parallel \overline{Y}_{\cdot i} \parallel \parallel \overline{Y}_{\cdot j} \parallel}]\big), \tag{3}$$

where $\overline{Y}_{\cdot i} = (Y_{1i}, Y_{2i}, \ldots, Y_{li})$ is a sub-vector of $Y_{\cdot i}$, and l denotes the number of partially labeled samples[1]. The parameter η is set as 10 in our experiments.

C. Solutions

1) An exact solution: For clarity, we denote the objective function (1) as $\mathcal{J}(Z)$. Obviously it is a convex optimization problem, such that we can easily gain the global optima by setting the derivative of $\mathcal{J}(Z)$ with respect to Z as 0, as follows:

$$\frac{\partial \mathcal{J}(Z)}{\partial Z} = 2(Z - Y) + \lambda_X Z L_X + \lambda_C L_C Z = 0. \quad (4)$$

It equals to solve a Sylvester matrix equation, as follows:

$$Z(I + \lambda_X L_X) + (I + \lambda_C L_C)Z = 2Y. \quad (5)$$

There have been many works about solving this equation [14], [15]. It has a unique solution. However, the computational complexity is $O(n^3)$. For a large data set, it means a high cost. Note that the entries of Z^* will be continuous values. We can do label ranking in each row to recover the integral values. We denote this solution as **MLML-exact**.

2) An approximated solution: We find an interesting issue that the objective function (1) is completely same with the one used in [16], which wants to propagate the pairwise constraints among two sources. A a result, we can utilize the efficient method designed for constraint propagation to solve (1). The main idea is to solve (1) through an alternative optimization procedure. Specifically, $\mathcal{J}(Z)$ is divided into two sub-problems:

$$Z_X^* = \arg\min_{Z_X} \frac{1}{2} \parallel Z_X - Y \parallel_{\mathcal{F}}^2 + \frac{\lambda_X}{2} tr(Z_X L_X Z_X^T), \quad (6)$$

$$Z^* = \arg\min_{Z} \frac{1}{2} \parallel Z - Z_X^* \parallel_{\mathcal{F}}^2 + \frac{\lambda_C}{2} tr(Z^T L_C Z), \quad (7)$$

where $Z_X \in \mathcal{R}^{m \times n}$. Denote the objective functions in above two equations as $\mathcal{J}_1(Z_X)$ and $\mathcal{J}_2(Z)$ respectively. Set their derivatives with respect to Z_X and Z as 0, then we gain:

$$\frac{\partial \mathcal{J}_1(Z_X)}{\partial Z_X} = Z_X - Y + \lambda_Y Z_X L_X = 0, \quad (8)$$

$$\frac{\partial \mathcal{J}_2(Z)}{\partial Z} = Z - Z_X^* + \lambda_C L_C Z = 0. \quad (9)$$

Combining the above two equations, one can easily obtain a closed-form solution as follows:

$$Z^* = (1-\alpha_X)(1-\alpha_C)(I - \alpha_C \overline{L}_C)^{-1} Y (I - \alpha_X \overline{L}_X)^{-1}, \quad (10)$$

where $\alpha_X = \frac{\lambda_X}{\lambda_X + 1} \in (0, 1)$, $\alpha_C = \frac{\lambda_c}{\lambda_c + 1} \in (0, 1)$. $\overline{L}_X = D_X^{-\frac{1}{2}} V_X D_X^{-\frac{1}{2}}$, $\overline{L}_C = D_C^{-\frac{1}{2}} V_C D_C^{-\frac{1}{2}}$. This solution is denoted as **MLML-appro**.

III. Experiments

In this section, we test the proposed method in three benchmark data sets in multi-label learning, including Emotions [17], AU [18] and Yeast [19], as shown in Table I. The predicted labels of the unlabeled testing samples are evaluated by the metric of the area under the ROC curve (AUC) [20].

[1]Since the label values of the completely unlabeled samples are all 0, they can not provide useful information for the semantic correlations. This is why we ignore the label entries corresponding to the completely unlabeled samples in Eq. (3).

TABLE I. Data statistics

data set	domain	# exam.	# categ.	# fea.	avg. positive-class/sample	avg. positive-sample /class
Emotions [17]	music	593	6	72	1.87	184.7
AU [18]	image	327	16	201	3.99	81.5
Yeast [19]	biology	2417	14	103	4.23	731.5

A. Experiments Setting

1) Data processing: In all experiments, each feature is normalized into $[-1, 1]$ for all data sets. The whole data set is randomly partitioned to 5 uniform folds. In each time one fold is used as the testing data, while the other four folds are training data. We repeat this process 10 times, then $5 \times 10 = 50$ results are gained. The mean value and standard deviation (std) are computed as the final outputs. To present the influence of missing labels, we vary the given label proportion in training data, from 20% (80% missing labels) to 100% (no missing labels). In each proportion, the missing labels are randomly chosen and removed from the ground-truth complete label matrix of the training data, then the initial label matrix Y is gained.

2) Comparisons: We compare the proposed method with several previous works on multi-label learning with missing labels, including SMSE2 [10], WELL [3] and MLR-GL [2]. We implement SMSE2 in matlab and adopt the publicly available matlab codes for WELL and MLR-GL. Note that SMSE2 is originally designed for semi-supervised multi-label learning, but it can also solve the MLML problem by setting the missing labels as negative labels. So we also run SMSE2 in the cases that missing labels exist in training data. In both MLML and SMSE2, the two trade-off parameters α_X and α_C are determined by the 5-fold cross-validation within the training data. We make our best effort to adjust the parameters in other methods as suggested in the original papers. Besides, as a baseline, a binary logistic regression classifier is trained based on only labeled samples for each class independently, and is implemented by the built-in functions *glmfit* and *glmval* in matlab.

B. Classification Results

The AUC results on different data sets are shown in Table II, III and IV respectively. The proposed methods show the best performance in most cases. MLML-appro gives the similar results with MLML-exact, and the effectiveness of the proposed approximated solution is verified. Note that MLML-appro performs even better than MLML-exact in some cases. This is possible. Because we do not optimize the AUC value directly, so the global optima of the objective function (1) may not correspond to the highest AUC value. But the consistency between (1) and the AUC value still holds in most cases. Compared with SMSE2, MLML-exact always give better results, especially in the case of low label proportion, which demonstrates the benefit of explicitly distinguishing the negative and missing labels. However, the larger improvement of MLML is reflected in robustness, which will be verified later. WELL shows poor performances in most cases, especially on the case of low label proportion, suffering from its confusion between negative and missing labels. The binary logistic regression also performs poor when the label proportion is low. We believe the reason is over-fitting, i.e., the small data size and the large number of parameters. The over-fitting is most obvious

on the AU data, where the total data size is 327, but the parameter is up to 202. MLR-GL shows the best performance among the compared methods. However, 4 parameters need to be manually tuned, and its performance is sensitive to the parameters in our experiments. In contrast, only 2 trade-off parameters should be manually determined in MLML. As we will show later, MLML is robust to the parameters. Moreover, MLR-GL gives an iterative solution, while we present the closed form solution to MLML. The running time between them will also be compared later. Moreover, due to the label bias, the positive label proportion in the ground-truth label matrix should be an important factor which can influence the specific results of all methods and the performance gap between other methods and MLML method. However, the positive proportions of three data sets used in our experiments are nearly same (see Table I). We will evaluate the relationship between this proportion and the performance on more data sets with different positive proportions in future work.

TABLE II. AUC RESULTS (%) ON EMOTIONS DATA

Algorithms	Label Proportion			
	20%	40%	80%	100%
Logistic	70.42±1.44	75.49±1.07	83.51±0.81	84.89±0.15
WELL [3]	69.26±1.36	76.76±1.00	81.23±0.26	82.33±0.36
MLR-GL [2]	84.64±0.30	85.92±0.35	**87.57±0.25**	87.86±0.17
SMSE2 [10]	83.91±0.67	85.72±0.41	87.23±0.14	87.19±0.24
MLML-appro	**84.88±0.62**	86.77±0.32	87.20±0.21	87.81±0.19
MLML-exact	84.73±0.67	**86.84±0.51**	87.46±0.22	**87.89±0.22**

TABLE III. AUC RESULTS (%) ON AU DATA

Algorithms	Label Proportion			
	20%	40%	80%	100%
Logistic	71.03±0.56	68.15±0.86	65.87±2.19	81.83±0.73
WELL [3]	78.15±0.71	83.90±0.52	82.29±0.17	82.22±0.05
MLR-GL [2]	**87.46±0.45**	89.44±0.38	90.68±0.25	91.39±0.09
SMSE2 [10]	85.04±0.47	88.36±0.56	90.97±0.21	91.88±0.27
MLML-appro	85.17±0.47	89.69±0.29	91.90±0.20	92.40±0.21
MLML-exact	87.27±0.42	**89.71±0.19**	**91.91±0.22**	**92.46±0.23**

TABLE IV. AUC RESULTS (%) ON YEAST DATA

Algorithms	Label Proportion			
	20%	40%	80%	100%
Logistic	77.67±0.35	80.80±0.30	82.47±0.10	83.00±0.07
WELL [3]	76.92±0.66	78.23±0.28	78.81±0.21	79.20±0.04
MLR-GL [2]	82.58±0.04	83.35±0.04	83.90±0.04	84.01±0.05
SMSE2 [10]	78.33±0.11	79.19±0.10	79.50±0.03	79.76±0.02
MLML-appro	82.74±0.13	83.02±0.07	84.51±0.11	84.70±0.06
MLML-exact	**83.05±0.19**	**83.69±0.08**	**84.41±0.08**	**84.70±0.12**

C. Empirical running time

The empirical running time (the average CPU time of 50 trials) of different methods are shown in Table V. All values are recorded based on the same machine with the Windows 7 system and Intel Core i5 2.30GHz CPU. For MLR-GL, the number of iterations T is about 10 in our experiments. Note that the cost of computing the distance matrix between instances is not included, since it is a shared step in all the methods except the binary logistic regression. Obviously the proposed approximated solution is much faster than all other methods.

TABLE V. EMPIRICAL RUNNING TIME (SEC.) OF DIFFERENT METHODS (T DENOTES THE NUMBER OF ITERATIONS IN MLR-GL).

Data	Logistic	SMSE2	WELL	MLR-GL	MLML-appro	MLML-exact
Emotions	4.68	0.055	2.31	$0.064 * T$	**0.015**	0.055
AU	7.86	0.014	3.57	$0.047 * T$	**0.0036**	0.014
Yeast	4.36	7.60	239.3	$0.65 * T$	**1.88**	7.60

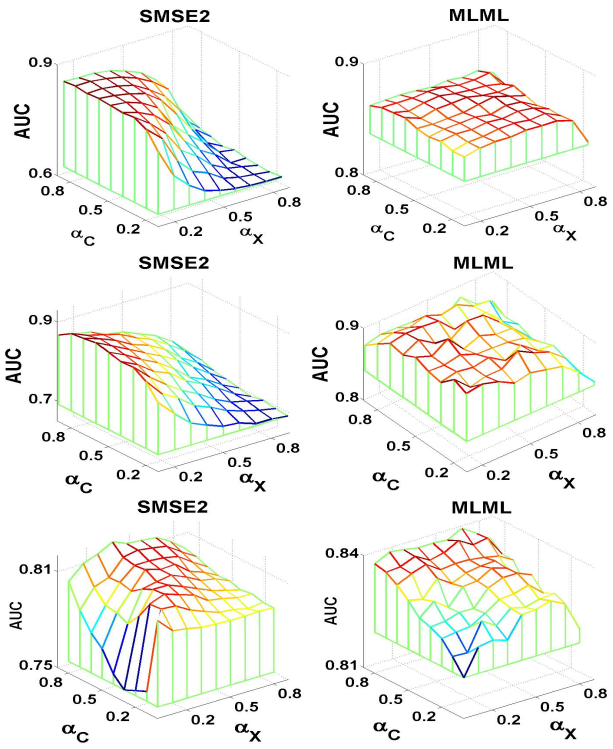


Fig. 2. AUC results with different parameters, label proportion = 50%: **top row** on the Emotions data, **Middle row** on the AU data, **Bottom row** on the Yeast data.

D. Parameter Tuning

The two trade-off parameters α_X and α_C in both MLML[2] and SMSE2 [10] control the influences between the label consistency and label smoothness. We vary α_X and α_C to study (1) the influences of different assumptions and (2) the robustness of MLML and SMSE2. For clarity, we fix the label proportion as 50%. We vary two parameters in the range $\{0.01 : 0.1 : 0.81\}$, then 81 results are gained. Results on Emotions are shown in the top row of Figure 2. For SMSE2, the results range from 63.31% to 86.73%, and the mean and std are 75.47±8.84%. In contrast, the results of MLML range from 84.01% to 87.43%, and the mean and std are $86.63 \pm 0.67\%$. Results on AU are shown in the intermediate row of Figure 2. The results of SMSE2 range from 69.35% to 89.72%, and the mean and std are $79.01 \pm 6.21\%$. The results of MLML range from 84.23% to 91.71%, and the mean and std are $88.93 \pm 1.9\%$. Results on Yeast are shown in the bottom row of Figure 2. For SMSE2, the results range from 75.68% to 82.34%, and the mean and std are $80.38 \pm 1.32\%$. The results of MLML range from 82.07% to 84.15%, and the mean and std are $83.41 \pm 0.45\%$. Above results demonstrate that MLML not only gives the higher AUC values than SMSE2, but also shows the much better robustness. Note that $\alpha_Y = \alpha_C = 0.01$ can be considered as ignoring the label smoothness. Compared with the results in this case, we conclude that the label smoothness can help to improve the performance of MLML, and the sample-level smoothness and class-level smoothness show different influences on different data sets.

[2]MLML in the section only denotes MLML-exact.

IV. Conclusions

This paper has solved the general problem of multi-label learning with missing labels (MLML), which generalizes several previous models that also handle the missing labels in multi-label learning. In MLML, the positive, negative and missing labels are explicitly distinguished, such that the ungrounded bias on labels in existing models is eliminated. We have demonstrated that such a change leads to significant performance improvement. Moreover, we present both the exact and an efficient approximated solution to the MLML problem. Experiments on three benchmark data sets have verified the efficacy of the proposed solutions.

Some future directions of MLML can be explored. First, the work only involves learning with hard labels, i.e., the given labels have 100% confidence. However, the hard labels are not always available in real applications. For example, the annotation of AU labels is often along with an intensity, i.e., the confidence of the annotation. These intensities can be easily transformed into soft labels, ranging from -1 to 1. To the best of our knowledge, no existing works have focused on multi-label learning with soft labels. If soft labels are simply discretized to hard labels, the information distortion will be introduced. However, we find that the proposed method can be directly extended to handle the soft labels. So we will explore more applications with soft labels in future. Second, we simply use the co-occurrence to compute the semantic dependencies among classes. Actually some previous works have focused on exploring the more comprehensive and complex semantic dependencies, such as [4], [21]. These dependencies can be exploited in our model, and the performance is expected to be further improved.

Acknowledgment

The authors would like to thank all reviewers for their constructive and helpful comments. The work was 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 supports. Qiang Ji and Shangfei Wang's involvement in this work is supported in part by the Natural Science Foundation of China (NSFC) under grant 61228304. Bao-Gang Hu and Baoyuan Wu are supported in part by NSFC 61273196.

References

[1] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 3, no. 3, pp. 1–13, 2007.

[2] S. Bucak, R. Jin, and A. Jain, "Multi-label learning with incomplete class assignments," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 2801–2808.

[3] Y. Sun, Y. Zhang, and Z. Zhou, "Multi-label learning with weak label," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[4] X. Chen, X. Yuan, Q. Chen, S. Yan, and T. Chua, "Multi-label visual classification with label exclusive context," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 834–841.

[5] M. Boutell, J. Luo, X. Shen, and C. Brown, "Learning multi-label scene classification," *Pattern recognition*, vol. 37, no. 9, pp. 1757–1771, 2004.

[6] S. Gao, W. Wu, C. Lee, and T. Chua, "A mfom learning approach to robust multiclass multi-label text categorization," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 42.

[7] S. Yang, H. Zha, and B. Hu, "Dirichlet-bernoulli alignment: A generative model for multi-class multi-label multi-instance corpora," in *Proceedings of Neural Information Processing Systems*, 2009, pp. 2143–2150.

[8] P. Ekamn and W. Friesen, "Facial action coding system (facs): manual," 1978.

[9] M. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, "Automatic recognition of facial actions in spontaneous expressions," *Journal of Multimedia*, vol. 1, no. 6, pp. 22–35, 2006.

[10] G. Chen, Y. Song, F. Wang, and C. Zhang, "Semi-supervised multi-label learning by solving a sylvester equation," in *SIAM international conference on data mining*, 2008, pp. 410–419.

[11] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[12] Z. Lihi and P. Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems*, 2004, pp. 1601–1608.

[13] S. Park and J. Fürnkranz, "Multi-label classification with label constraints," in *Proceedings of the Proceedings of the ECML/PKDD-08 Workshop on Preference Learning (PL-08)*, 2008, pp. 157–171.

[14] R. Bartels and G. Stewart, "Solution of the matrix equation ax+ xb= c [f4]," *Communications of the ACM*, vol. 15, no. 9, pp. 820–826, 1972.

[15] D. Hu and L. Reichel, "Krylov-subspace methods for the sylvester equation," *Linear Algebra and its Applications*, vol. 172, pp. 283–313, 1992.

[16] Z. Lu, H. Ip, and Y. Peng, "Exhaustive and efficient constraint propagation: A semi-supervised learning perspective and its applications," *arXiv preprint arXiv:1109.4684*, 2011.

[17] K. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multi-label classification of music into emotions," in *ISMIR 2008: Proceedings of the 9th International Conference of Music Information Retrieval*. Lulu. com, 2008, p. 325.

[18] P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 94–101.

[19] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," *NIPS*, vol. 14, pp. 681–687, 2001.

[20] S. Bucak, P. Kumar Mallapragada, R. Jin, and A. Jain, "Efficient multi-label ranking for multi-class learning: application to object recognition," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 2098–2105.

[21] S.-J. Huang and Z.-H. Zhou, "Multi-label learning by exploiting label correlations locally," in *AAAI*, 2012, pp. 949–955.