Multi-Label Learning by Exploiting Label Correlations Locally*

Sheng-Jun Huang **Zhi-Hua ZHou**
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210046, China
{huangsj, zhouzh}@lamda.nju.edu.cn

Abstract

It is well known that exploiting label correlations is important for multi-label learning. Existing approaches typically exploit label correlations globally, by assuming that the label correlations are shared by all the instances. In real-world tasks, however, different instances may share different label correlations, and few correlations are globally applicable. In this paper, we propose the ML-LOC approach which allows label correlations to be exploited locally. To encode the local influence of label correlations, we derive a LOC code to enhance the feature representation of each instance. The global discrimination fitting and local correlation sensitivity are incorporated into a unified framework, and an alternating solution is developed for the optimization. Experimental results on a number of image, text and gene data sets validate the effectiveness of our approach.

Introduction

Traditional supervised learning deals with problems where one example is associated with a single class label. In many real-world tasks, however, one example may simultaneously have multiple class labels; for example, an image can be tagged with several keywords (Boutell *et al.* 2004), a document may belong to multiple topics (McCallum 1999; Ueda and Saito 2003), and a gene may be related to multiple functions (Elisseeff and Weston 2002). To handle such tasks, multi-label learning has attracted much attention during the past years (Ghamrawi and Mccallum 2005; Hsu *et al.* 2009; Hariharan *et al.* 2010; Bucak *et al.* 2011).

A straightforward solution to multi-label learning is to decompose the problem into a series of binary classification problems, each for one label. Such a solution, however, neglects the fact that information of one label may be helpful for the learning of another related label; especially when some labels have insufficient training examples, the label correlations may provide helpful extra information. Thus, it is not strange that the exploitation of label correlations has been widely accepted as a key component of current multi-label learning approaches (Tsoumakas *et al.* 2009; Dembczynski *et al.* 2010; Zhang and Zhang 2010).

To exploit label correlations, some approaches resort to external knowledge such as existing label hierarchies (Cai and Hofmann 2004; Rousu *et al.* 2005; Cesa-Bianchi *et al.* 2006; Jin *et al.* 2008) or label correlation matrices (Hariharan *et al.* 2010). Many other approaches try to exploit label correlations concealed in the training data. For example, the co-occurrence of labels in training data is utilized in (Tsoumakas *et al.* 2009; Petterson and Caetano 2011); the approach in (Ghamrawi and Mccallum 2005) tries to model the impact of an individual feature on the co-occurrence probability of label pairs; Sun et al. (2008) proposed to use a hypergraph to model the correlation information contained in different labels; in (Zhang and Zhang 2010), a Bayesian network structure is used to encode the conditional dependencies of both the labels and feature set.

Although different multi-label learning approaches have tried to exploit different orders (first-order, second-order and high-order) of label correlations (Zhang and Zhang 2010), they usually exploit label correlations in a global way by assuming that the correlations are shared by all instances. In real-world tasks, however, label correlations are naturally local, where a label correlation may be shared by only a subset of instances rather than all the instances. For example, as shown in Figure 1, we consider the strong correlation between *mountains* and *trees*. For the image (b), *trees* are less prominent and thus could be difficult to predict; in this case, the correlation between *mountains* and *trees* can be helpful to the learning task since the label *mountains* is relatively easier to predict in this image. For the image (c), with the *trees* clearly presents, this correlation turns to be misleading since it will suggest to include the label *mountains*, whereas this label is not proper for the image. Exploiting such correlations globally will enforce

*This research was supported by NSFC (61073097, 61021062), 973 Program (2010CB327903) and Baidu fund.

(a) *mountains, trees, sky, river* (b) *mountains, sky, river, trees* (c) *trees, desert, sky, camel*

Figure 1: Illustration of non-global label correlations.

unnecessary or even misleading constraints on instances that do not contain such correlations, and therefore may hurt the performance by predicting some irrelevant labels.

In this paper, we propose the ML-LOC (Multi-Label learning using LOcal Correlation) approach, which tries to exploit label correlations in the data locally. We do not assume that there are external knowledge sources specifying the locality of label correlations. Instead, we assume that the instances can be separated into different groups and each group share a subset of label correlations. To encode the local influence of label correlations, we construct a LOC (LOcal Correlation) code for each instance and use this code as additional features for the instance. We formulate the problem by incorporating global discrimination fitting and local correlation sensitivity into a unified framework, and develop an alternating solution for the optimization. The effectiveness of ML-LOC is validated in experiments.

We propose the ML-LOC approach in the next section, and then present our experiments, followed by the conclusion.

The ML-LOC Approach

The Framework

Let $\mathcal{X} = \mathcal{R}^d$ be the input space and $\mathcal{Y} = \{-1, +1\}^L$ be the label space with L possible labels. We denote by $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$ the training data that consists of n instances, where each instance $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]$ is a vector of d dimensions and $\mathbf{y}_i = [y_{i1}, y_{i2}, \cdots, y_{iL}]$ is the label vector of $\mathbf{x}_i$, y_{il} is $+1$ if $\mathbf{x}_i$ has the l-th label and -1 otherwise. The goal is to learn a function $\mathbf{f} : \mathcal{X} \to \mathcal{Y}$ which can predict the label vectors for unseen instances.

We introduce a code vector $\mathbf{c}_i$ for each instance $\mathbf{x}_i$ to encode the local influence of label correlations, and expand the original feature representation with the code. Since similar instances share the same subset of label correlations, the code is expected to be local, i.e., similar instances will have similar $\mathbf{c}$. Notice that we measure the similarity between instances in the label space rather than in the feature space because instances with similar label vectors usually share the same correlations. For simplicity, we assume $\mathbf{f}$ consists of L functions, one for a label, i.e., $\mathbf{f} = [f_1, f_2, \cdots, f_L]$, and each f_l is a linear model: $f_l(\mathbf{x}, \mathbf{c}) = \langle \mathbf{w}_l, [\phi(\mathbf{x}), \mathbf{c}] \rangle = \langle \mathbf{w}_l^x, \phi(\mathbf{x}) \rangle + \langle \mathbf{w}_l^c, \mathbf{c} \rangle$, where $\langle \cdot, \cdot \rangle$ is the inner production, $\phi(\cdot)$ is a feature mapping induced by a kernel κ, and $\mathbf{w}_l = [\mathbf{w}_l^x, \mathbf{w}_l^c]$ is the weight vector. Here $\mathbf{w}_l^x$ and $\mathbf{w}_l^c$ are the two parts of $\mathbf{w}_l$ corresponding to the original feature vector $\mathbf{x}$ and the code $\mathbf{c}$, respectively. Aiming to the global discrimination fitting and local correlation sensitivity, we optimize both $\mathbf{f}$ and C such that the following function is minimized:

$$\min_{\mathbf{f}, C} \sum\nolimits_{i=1}^{n} V(\mathbf{x}_i, \mathbf{c}_i, \mathbf{y}_i, \mathbf{f}) + \lambda_1 \Omega(\mathbf{f}) + \lambda_2 Z(C) \quad (1)$$

where V is the loss function defined on the training data, Ω is a regularizer to control the complexity of the model $\mathbf{f}$, Z is a regularizer to enforce the locality of the codes C, and λ_1 and λ_2 are parameters trading off the three terms.

Among the different multi-label losses, in this paper, we focus on the commonly used *hamming loss* and define the loss function V as the summarization of losses across all the labels:

$$V(\mathbf{x}_i, \mathbf{c}_i, \mathbf{y}_i, \mathbf{f}) = \sum\nolimits_{l=1}^{L} loss(\mathbf{x}_i, \mathbf{c}_i, y_{il}, f_l), \quad (2)$$

where $loss(\mathbf{x}, \mathbf{c}, y, f) = \max\{0, 1 - yf([\phi(\mathbf{x}), \mathbf{c}])\}$ is the Hinge loss in SVM. For the second term of Eq. 1, we simply implement it as:

$$\Omega(\mathbf{f}) = \sum\nolimits_{l=1}^{L} ||\mathbf{w}_l||^2. \quad (3)$$

The third term of Eq. 1 is utilized to enforce that similar instances have similar codes. With the previous discussion we know that for a specific instance, only a subset of label correlations are helpful whereas the others are less informative or even harmful, and similar instances share the same label correlations. We assume that the training data can be separated into m groups $\{G_1, G_2, \cdots, G_m\}$, where instances in the same group share the same subset of label correlations. Inspired by (Zhou and Zhang 2007; Zhou *et al.* 2012), the groups can be discovered via clustering. Correspondingly, we denote by S_j the subset of correlations that are shared by the instances in G_j. Since label correlations are usually unavailable and very difficult to obtain, we instead represent S_j with a prototype of all the instances in G_j, denoted by $\mathbf{p}_j$. As in Eq. 4, we generate the prototype $\mathbf{p}_j$ in the label space by averaging the label vectors of the instances in G_j:

$$\mathbf{p}_j = \frac{1}{|G_j|} \sum\nolimits_{\mathbf{x}_k \in G_j} \mathbf{y}_k, \quad (4)$$

where $|G_j|$ is the number of instances in G_j. Then, given an instance $\mathbf{x}_i$, we define c_{ij} to measure the local influence of S_j on $\mathbf{x}_i$. The more similar $\mathbf{y}_i$ and $\mathbf{p}_j$, the more likely that $\mathbf{x}_i$ shares the same correlations with instances in G_j, suggesting a larger value of c_{ij} and implying a higher probability that S_j is helpful to $\mathbf{x}_i$. We name the concatenated vector $\mathbf{c}_i = [c_{i1}, c_{i2}, \cdots, c_{im}]$ LOC code (LOcal Correlation code) for $\mathbf{x}_i$, and define the regularizer term on the LOC codes as:

$$Z(C) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} ||\mathbf{y}_i - \mathbf{p}_j||^2. \tag{5}$$

By substituting Eqs. 2, 3 and 5 into Eq. 1, the framework can be rewritten as:

$$\min_{W,C,P} \sum_{i=1}^{n} \sum_{l=1}^{L} \xi_{il} + \lambda_1 \sum_{l=1}^{L} ||\mathbf{w}_l||^2 + \lambda_2 \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} ||\mathbf{y}_i - \mathbf{p}_j||^2 \tag{6}$$

$$\begin{aligned} s.t. \quad & y_{il} \langle \mathbf{w}_l, [\phi(\mathbf{x}_i), \mathbf{c}_i] \rangle \geq 1 - \xi_{il} \\ & \xi_{il} \geq 0 \quad \forall i \in \{1, \cdots, n\},\ l \in \{1, \cdots, L\} \\ & \sum\nolimits_{j=1}^{m} c_{ij} = 1 \quad \forall i \in \{1, \cdots, n\} \\ & 0 \leq c_{ij} \leq 1 \quad \forall i \in \{1, \cdots, n\},\ j \in \{1, \cdots, m\} \end{aligned}$$

c_{ij} measures the probability that S_j is helpful to x_i, thus, it is constrained to be in the interval [0, 1], and the sum of each $\mathbf{c}_i$ is constrained to be 1.

The Alternating Solution

We solve the optimization problem in Eq. 6 in an alternating way, i.e., optimizing one of the three variables with the other two fixed. When we fix C and P to solve W, the third term of Eq. 6 is a constant and thus can be ignored, then Eq. 6 can be rewritten as:

$$\min_{W} \sum_{i=1}^{n} \sum_{l=1}^{L} \xi_{il} + \lambda_1 \sum_{l=1}^{L} ||\mathbf{w}_l||^2 \tag{7}$$

$$\begin{aligned} s.t. \quad & y_{il} \langle \mathbf{w}_l, [\phi(\mathbf{x}_i), \mathbf{c}_i] \rangle \geq 1 - \xi_{il} \\ & \xi_{il} \geq 0 \quad \forall i = \{1, \cdots, n\},\ l = \{1, \cdots, L\} \end{aligned}$$

Notice that Eq. 7 can be further decomposed into L optimization problems, where the l-th one is:

$$\min_{\mathbf{w}_l} \sum_{i=1}^{n} \xi_{il} + \lambda_1 ||\mathbf{w}_l||^2 \tag{8}$$

$$\begin{aligned} s.t. \quad & y_{il} \langle \mathbf{w}_l, [\phi(\mathbf{x}_i), \mathbf{c}_i] \rangle \geq 1 - \xi_{il} \\ & \xi_{il} \geq 0 \quad \forall i = \{1, \cdots, n\} \end{aligned}$$

which is a standard SVM model. So, W can be optimized by independently training L SVM models.

When we fix W, P to solve C, the task becomes:

$$\min_{C} \sum_{i=1}^{n} \sum_{l=1}^{L} \xi_{il} + \lambda_2 \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} ||\mathbf{y}_i - \mathbf{p}_j||^2 \tag{9}$$

$$\begin{aligned} s.t. \quad & y_{il} \langle \mathbf{w}_l, [\phi(\mathbf{x}_i), \mathbf{c}_i] \rangle \geq 1 - \xi_{il} \\ & \xi_{il} \geq 0 \quad \forall i = \{1, \cdots, n\},\ l = \{1, \cdots, L\} \\ & \sum\nolimits_{j=1}^{m} c_{ij} = 1 \quad \forall i = \{1, \cdots, n\} \\ & 0 \leq c_{ij} \leq 1 \quad \forall i = \{1, \cdots, n\},\ j = \{1, \cdots, m\} \end{aligned}$$

Algorithm 1 The ML-LOC algorithm

```
INPUT:
   training set {X, Y}, parameters λ1, λ2 and m
TRAIN:
   initialize P and C with k-means
   repeat:
      optimize W according to Eq. 7
      optimize C according to Eq. 9
      update P according to Eq. 11
   until convergence
   for j=1:m
      train a regression model R_j
      for the j-th dimension of LOC codes
   end for
TEST:
   predict the LOC codes: c_tj ← R_j(φ(x_t))
   predict the labels: y_tl ← ⟨w_l, [φ(x_t), c_t]⟩
```

Again, Eq. 9 can be decomposed into n optimization problems, and the i-th one is:

$$\min_{\mathbf{c}_i} \sum_{l=1}^{L} \xi_{il} + \lambda_2 \sum_{j=1}^{m} c_{ij} ||\mathbf{y}_i - \mathbf{p}_j||^2 \tag{10}$$

$$\begin{aligned} s.t. \quad & y_{il} \langle \mathbf{w}_l^c, \mathbf{c}_i \rangle \geq 1 - \xi_{il} - y_{il} \langle \mathbf{w}_l^x, \phi(\mathbf{x}_i) \rangle \\ & \xi_{il} \geq 0 \quad \forall l = \{1, \cdots, L\} \\ & \sum\nolimits_{j=1}^{m} c_{ij} = 1 \\ & 0 \leq c_{ij} \leq 1 \quad \forall j = \{1, \cdots, m\} \end{aligned}$$

which is a linear programming and can be solved efficiently. After the training, the c's reflect the similarities between the labels and the prototypes; the larger the similarity, the larger the corresponding element of c. Due to page limit, we will show some examples in a longer version.

Finally, as W and C are updated, the belongingness of each instance x_i to each group G_j may be changed, so the prototype P is correspondingly updated according to:

$$\mathbf{p}_j = \sum\nolimits_{i=1}^{n} c_{ij} \mathbf{y}_i. \tag{11}$$

Notice that the classification models are trained based on both the original feature vector and the LOC codes. Given a test instance $\mathbf{x}_t$, its LOC code $\mathbf{c}_t$ is unknown. We thus train m regression models on the training instances and their LOC codes, one for a dimension of LOC codes. Then, in testing phase, the LOC codes of the test instance can be obtained from the outputs of regression models. Notice that the regression models can be trained with a low computational cost because usually m is not large.

The pseudo code of ML-LOC is presented in Algorithm 1. P and C are initialized with the result of k-means clustering on the label vectors. In detail, the la-

Table 1: Results (mean±std.) on image data sets. •(∘) indicates that ML-LOC is significantly better(worse) than the corresponding method on the criterion based on paired t-tests at 95% significance level. ↑(↓) implies the larger(smaller), the better.

data	criteria	ML-LOC	BSVM	ML-kNN	RankSVM	ECC	ECC-LOC
image	$hloss$ ↓	.156±.007	.179±.006•	.175±.007•	.339±.021•	.180±.010•	.165±.008•
	$one\text{-}error$ ↓	.277±.022	.291±.016•	.325±.024•	.708±.052•	.300±.022•	.249±.020∘
	$coverage$ ↓	.175±.015	.181±.010•	.194±.012•	.420±.013•	.200±.011•	.201±.011•
	$rloss$ ↓	.152±.017	.161±.009•	.177±.013•	.463±.018•	.247±.016•	.297±.022•
	$aveprec$ ↑	.819±.014	.808±.010•	.788±.014•	.516±.011•	.789±.014•	.805±.011•
scene	$hloss$ ↓	.076±.003	.098±.002•	.090±.003•	.251±.017•	.095±.004•	.075±.004
	$one\text{-}error$ ↓	.179±.010	.209±.014•	.238±.012•	.457±.065•	.232±.011•	.152±.009∘
	$coverage$ ↓	.069±.008	.073±.005•	.084±.005•	.192±.032•	.095±.004•	.098±.006•
	$rloss$ ↓	.065±.009	.070±.005•	.083±.006•	.214±.039•	.139±.008•	.201±.011•
	$aveprec$ ↑	.891±.008	.876±.008•	.857±.007•	.698±.047•	.846±.007•	.871±.006•
corel5K	$hloss$ ↓	.009±.000	.009±.000•	.009±.000•	.012±.001•	.014±.000•	.015±.000•
	$one\text{-}error$ ↓	.638±.010	.768±.009•	.740±.011•	.977±.018•	.666±.008•	.674±.012•
	$coverage$ ↓	.474±.013	.316±.003∘	.307±.003∘	.735±.036•	.746±.007•	.809±.007•
	$rloss$ ↓	.218±.008	.141±.002∘	.135±.002∘	.408±.035•	.601±.006•	.702±.007•
	$aveprec$ ↑	.294±.005	.214±.003•	.242±.005•	.067±.007•	.227±.004•	.194±.005•

bel vectors are clustered into m clusters and the prototype $\mathbf{p}_k$ is initialized with the center of the k-th cluster. c_{ik} is assigned 1 if $\mathbf{y}_i$ is in the k-th cluster and 0 otherwise. Here k-means is used simply because its popularity; ML-LOC can be facilitated with other clustering approaches. Then W, C and P are updated alternatingly. Since Eq. 6 is lower bounded, it will converge to a local minimum. After that, m regression models are trained with the original features as inputs and the LOC codes as outputs. Given a test instance $\mathbf{x}_t$, the LOC code $\mathbf{c}_t = [c_{t1}, c_{t2}, \cdots, c_{tm}]$ is firstly obtained with the regression models. Then the final label vector $\mathbf{y}_t$ is obtained by $y_{tl} = \langle \mathbf{w}_l, [\phi(\mathbf{x}_t),\ \mathbf{c}_t] \rangle$.

Experiments

Comparison with State-of-the-Art Approaches

To examine the effectiveness of LOC codes, ML-LOC is firstly compared with BSVM (Boutell *et al.* 2004), which learns a binary SVM for each label, and can be considered as a degenerated version of ML-LOC without LOC codes. ML-LOC is also compared with three other state-of-the-art multi-label methods: ML-kNN (Zhang and Zhou 2007) which considers first-order correlations, RankSVM (Elisseeff and Weston 2002) which considers second-order correlations and ECC (Read *et al.* 2011) which considers higher-order correlations. Finally, we construct another baseline ECC-LOC, by feeding the new feature vectors expanded with LOC codes into ECC as input. For the compared methods, the parameters recommended in the corresponding literatures are used. For ML-LOC, parameters are fixed for all the data sets as: $\lambda_1 = 1$, $\lambda_2 = 100$ and $m = 15$. LibSVM (Chang and Lin 2011) is used to implement the SVM models for both ML-LOC and BSVM.

We evaluate the performances of the compared approaches with five commonly used multi-label criteria: *hamming loss*, *one error*, *coverage*, *ranking loss* and *average precision*. These criteria measure the performance from different aspects and the detailed definitions can be found in (Schapire and Singer 2000; Zhou *et al.* 2012). Notice that we normalize the *coverage* by the number of possible labels such that all the evaluation criteria vary between $[0,\ 1]$.

We study the effectiveness of ML-LOC on seven multi-label data sets from three kinds of tasks: image classification, text analysis and gene function prediction. In the following experiments, on each data set, we randomly partition the data into the training and test sets for 30 times, and report the average results as well as standard deviations over the 30 repetitions.

Image Data We first perform the experiments on three image classification data sets: *image*, *scene* and *corel5k*. They have 2000, 2407, 5000 images, and 5, 6, 374 possible labels, respectively. The performances on five evaluation criteria are summarized in Table 1. Notice that for *average precision*, a larger value implies a better performance, whereas for the other four criteria, the smaller, the better. First, we exclude ECC-LOC, and compare ML-LOC with the baselines without considering local correlations. On *hamming loss*, for which ML-LOC is designed to optimize, our approach outperforms the other approaches significantly on all the three data sets. Although ML-LOC does not

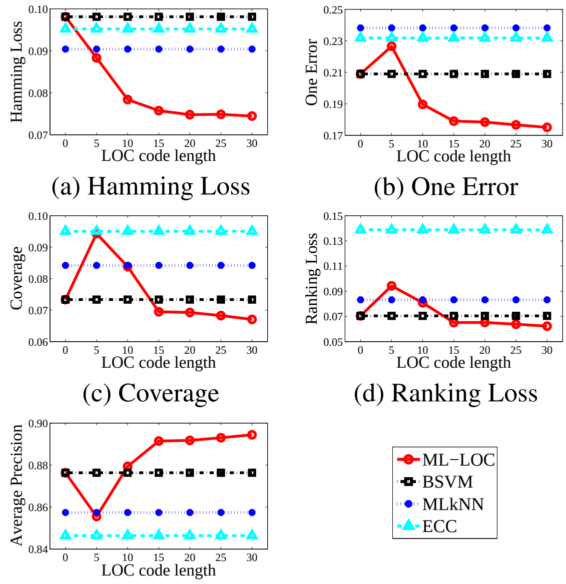
Table 2: Results (mean±std.) on text data sets. •(◦) indicates that ML-LOC is significantly better(worse) than the corresponding method on the criterion based on paired t-tests at 95% significance level. ↑(↓) implies the larger(smaller), the better.

data	criteria	ML-LOC	BSVM	ML-kNN	RankSVM	ECC	ECC-LOC
medical	$hloss$ ↓	.010±.001	.020±.001•	.016±.001•	.038±.001•	.010±.001	.010±.001
	$one\text{-}error$ ↓	.135±.015	.290±.026•	.275±.023•	.735±.020•	.110±.016◦	.104±.014◦
	$coverage$ ↓	.032±.006	.055±.006•	.060±.007•	.156±.007•	.071±.008•	.094±.012•
	$rloss$ ↓	.021±.004	.041±.005•	.043±.006•	.141±.006•	.109±.014•	.162±.022•
	$aveprec$ ↑	.898±.009	.776±.017•	.788±.017•	.394±.016•	.866±.014•	.831±.018•
enron	$hloss$ ↓	.048±.002	.056±.002•	.051±.002•	.311±.367•	.055±.002•	.055±.003•
	$one\text{-}error$ ↓	.257±.034	.359±.033•	.299±.031•	.855±.020•	.228±.036◦	.212±.035◦
	$coverage$ ↓	.314±.017	.294±.017◦	.246±.016◦	.500±.025•	.391±.021•	.442±.024•
	$rloss$ ↓	.113±.009	.115±.008	.091±.008◦	.267±.019•	.246±.018•	.294±.019•
	$aveprec$ ↑	.662±.019	.578±.019•	.636±.015•	.262±.017•	.637±.021•	.621±.023•

aim to optimize the other four evaluation criteria, it still achieves the best performance in most cases. Comparing with BSVM, which is a degenerated version of ML-LOC without LOC codes, ML-LOC is always superior to BSVM except for the *coverage* and *ranking loss* on *corel5K*, verifying the effectiveness of the LOC codes. It is noteworthy that BSVM outperforms ECC on the *scene* data set on all the five criteria. This interesting observation will be further studied later in this paper. Finally, when looking at ECC-LOC, its performances are worse than or comparable with ML-LOC. This is reasonable because most helpful local correlations have already been captured in the LOC codes, and thus little additional gains or even misleading information could be obtained by exploiting global correlations.

Text Data *Medical* is a data set of clinical texts for medical classification. There are 978 instances and 45 possible labels. *Enron* is a subset of the Enron email corpus (Klimt and Yang 2004), including 1702 emails with 53 possible labels. Results on these text data sets are summarized in Table 2. It can be seen that the comparison results are similar as that on the image data sets. ML-LOC outperforms the baselines in most cases, especially on *hamming loss* and *average precision*. ECC has advantages on *one error*, but achieves suboptimal performances on other criteria.

Gene Data *Genebase* is a data set for protein classification; it has 662 instances and 27 possible labels. *Yeast* is a data set for predicting the gene functional classes of the Yeast Saccha-romyces cerevisiae, containing 2417 genes and 14 possible labels. Table 3 shows the results. Again, ML-LOC achieves decent performance and is significantly superior to the compared approaches in most cases. ECC achieves the best performance on *one error*, but is less effective on the other criteria. One possible reason is that ECC utilize label correlations globally on all the instances and may predict some irrelevant labels for some instances that do not share the correlations. The incorrect predictions may be ranked after some very confident correct predictions, and thus do not affect the *one error* which cares only the top-ranked label, whereas leading to worse performances on the other criteria.



(a) Hamming Loss (b) One Error (c) Coverage (d) Ranking Loss (e) Average Precision (f) Legend

Figure 2: Influence of LOC code length on *scene*

Influence of LOC Code Length

To examine the influence of the length of LOC codes, i.e., the parameter m, we run ML-LOC with m varying from 0 to 30 with step size of 5. Due to the page limit, we only report the results on the *scene* data set in Figure 2, whereas experiments on other data sets get similar

Table 3: Results (mean±std.) on gene data sets. •(○) indicates that ML-LOC is significantly better(worse) than the corresponding method on the criterion based on paired t-tests at 95% significance level. ↑(↓) implies the larger(smaller), the better.

data	criteria	ML-LOC	BSVM	ML-kNN	RankSVM	ECC	ECC-LOC
genebase	*hloss* ↓	.001±.000	.005±.001•	.005±.001•	.008±.003•	.001±.001•	.001±.001•
	one-error ↓	.004±.003	.004±.003	.012±.010•	.047±.015•	.000±.001○	.000±.001○
	coverage ↓	.010±.002	.013±.003•	.015±.003•	.031±.011•	.013±.004•	.014±.004•
	rloss ↓	.001±.001	.002±.001•	.004±.002•	.016±.007•	.005±.003•	.007±.004•
	aveprec ↑	.997±.002	.995±.002•	.989±.005•	.953±.014•	.994±.003•	.993±.004•
yeast	*hloss* ↓	.187±.002	.189±.003•	.196±.003•	.196±.003•	.208±.005•	.200±.005•
	one-error ↓	.216±.010	.217±.011	.235±.012•	.224±.009•	.180±.012○	.169±.008○
	coverage ↓	.451±.006	.461±.005•	.449±.006	.473±.010•	.512±.009•	.517±.008•
	rloss ↓	.162±.004	.169±.005•	.168±.006•	.172±.006•	.279±.011•	.290±.012•
	aveprec ↑	.777±.005	.771±.007•	.762±.007•	.767±.007•	.731±.007•	.739±.007•

results. Notice that, for *average precision*, the larger the value, the better the performance; but for the other four criteria, the smaller, the better. The results of BSVM, ML-kNN and ECC are also plotted in the figures; the result of RankSVM is not plotted because it cannot be properly shown in the figures. As can be seen from the figure, a short LOC code leads to a poor performance, whereas after m grows sufficiently large, ML-LOC outperforms the other approaches and achieves stable performances.

Locally vs. Globally Correlations Exploiting

We have noted that on the *image* data set in Table 1, BSVM, which does not exploit label correlations, outperforms ECC which globally exploits correlations on all the five evaluation criteria, whereas ML-LOC is significantly better than both of them. This observation implies that globally exploiting the naturally local label correlations may lead to negative effects in some cases. To further understand the difference between approaches locally and globally exploiting correlations, we examine the predictions of the three approaches on *image*. First, we calculate the *precision* and *recall*, which measure, respectively, the percentage of the predicted labels that are correct, and the percentage of true labels that are predicted. The *precision/recall* of BSVM, ECC and ML-LOC are 0.86/0.33, 0.63/0.63 and 0.80/0.50, respectively. The low precision and high recall of ECC suggest that the globally exploiting approach may assign many irrelevant labels to the images. As the examples shown in Figure 3, for the first image, *tree* is relatively difficult to be identified; by exploiting label correlation, ECC and ML-LOC are able to predict it, but BSVM fails as it does not exploit label correlation. For the second image, ECC predicts more labels as needed, possibly because it overly generalizes through globally exploiting label correlations.

image		
ground-truth	*mountain, trees*	*trees, sunset*
BSVM	*mountain*	*trees, sunset*
ECC	*mountain, trees*	*trees, sunset, mountain*
ML-LOC	*mountain, trees*	*trees, sunset*

Figure 3: Typical prediction examples of BSVM, ECC and ML-LOC

Conclusion

In real-world multi-label learning tasks, label correlations are usually shared by subsets of instances rather than all the instances. Contrasting to existing approaches that exploit label correlations globally, in this paper we propose the ML-LOC approach which exploits label correlations locally, through encoding the local influences of label correlations in a LOC code, and incorporating the global discrimination fitting and local correlation sensitivity into a unified framework. Experiments show that ML-LOC is superior to many state-of-the-art multi-label learning approaches. It is possible to develop variant approaches by using other loss functions for the global discrimination fitting and other schemes for measuring the local correlation sensitivity. The effectiveness of these variants will be studied in future work. There may be better schemes than regression models to predict the LOC codes; in particular, the labels and local correlations may be predicted simultaneously with a unified framework. It is also interesting to incorporate external knowledge on label correlations into our framework.

Acknowledgements: We thank Yu-Feng Li for helpful discussions.

References

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

S. S. Bucak, R. Jin, and A. K. Jain. Multi-label learning with incomplete class assignments. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2801–2808, Colorado Springs, CO, 2011.

L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, pages 78–87, Washington, DC, 2004.

N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Hierarchical classification: Combining bayes with SVM. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 177–184, Pittsburgh, PA, 2006.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011.

K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning*, pages 279–286, Haifa, Israel, 2010.

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

N. Ghamrawi and A. Mccallum. Collective multilabel classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 195–200, Bremen, Germany, 2005.

B. Hariharan, L. Zelnik-Manor, S. V. N. Vishwanathan, and M. Varma. Large scale max-margin multi-label classification with priors. In *Proceedings of the 27th International Conference on Machine Learning*, pages 423–430, Haifa, Israel, 2010.

D. Hsu, S. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 772–780. MIT Press, Cambridge, MA, 2009.

B. Jin, B. Muller, C. Zhai, and X. Lu. Multi-label literature classification based on the gene ontology graph. *BMC bioinformatics*, 9(1):525, 2008.

B. Klimt and Y. Yang. Introducing the enron corpus. In *Proceedinds of the 1st International Conference on Email and Anti-Spam*, Mountain View, CA, 2004.

A. McCallum. Multi-label text classification with a mixture model trained by EM. In *Working Notes of the AAAI'99 Workshop on Text Learning*, Orlando, FL, 1999.

James Petterson and Tiberio S. Caetano. Submodular multi-label learning. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1512–1520. MIT Press, Cambridge, MA, 2011.

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

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

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

L. Sun, S. Ji, and J. Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 668–676, Las Vegas, NV, 2008.

G. Tsoumakas, A. Dimou, E. Spyromitros, and V. Mezaris. Correlation-based pruning of stacked binary relevance models for multi-label learning. In *Proceedings of the 1st International Workshop on Learning from Multi-Label Data*, pages 101–116, Bled, Slovenia, 2009.

N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 721–728, Cambridge, MA, 2003. MIT Press.

M.-L. Zhang and K. Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 999–1007, Washington, DC, 2010.

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

Z.-H. Zhou and M.-L. Zhang. Solving multi-instance problems with classifier ensemble based on constructive clustering. *Knowledge and Information Systems*, 11(2):155–170, 2007.

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