LIFT: Multi-Label Learning with Label-Specific Features

Min-Ling Zhang$^{1,2,*}$

$^1$School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
$^2$National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
zhangml@seu.edu.cn

Abstract

Multi-label learning deals with the problem where each training example is represented by a single instance while associated with a set of class labels. For an unseen example, existing approaches choose to determine the membership of each possible class label to it based on identical feature set, i.e. the very instance representation of the unseen example is employed in the discrimination processes of all labels. However, this commonly-used strategy might be suboptimal as different class labels usually carry specific characteristics of their own, and it could be beneficial to exploit different feature sets for the discrimination of different labels. Based on the above reflection, we propose a new strategy to multi-label learning by leveraging label-specific features, where a simple yet effective algorithm named LIFT is presented. Briefly, LIFT constructs features specific to each label by conducting clustering analysis on its positive and negative instances, and then performs training and testing by querying the clustering results. Extensive experiments across sixteen diversified data sets clearly validate the superiority of LIFT against other well-established multi-label learning algorithms.

1 Introduction

Multi-label learning deals with objects having multiple labels simultaneously, which widely exist in real-world applications [Boutell et al., 2004; Elisseeff and Weston, 2002; Schapire and Singer, 2000]. Formally, let $\mathcal{X} = \mathbb{R}^d$ be the $d$-dimensional input space and $\mathcal{Y} = \{l_1, l_2, \ldots, l_q\}$ be the finite set of $q$ possible labels. Then, the task of multi-label learning (or multi-label classification) is to learn a function $h : \mathcal{X} \rightarrow 2^\mathcal{Y}$ which maps each instance $x \in \mathcal{X}$ into a set of proper labels $h(x) \subseteq \mathcal{Y}$.

During the past decade, considerable number of approaches have been proposed to learn from multi-label data [Tsoumakas et al., 2009]. The common strategy adopted by existing approaches is that identical feature representation of the instance, i.e. $x$, is utilized to discriminate all the class labels in $\mathcal{Y}$. This amounts to induce a family of $q$ functions $\{f_1, f_2, \ldots, f_q\}$, where $f_k : \mathcal{X} \rightarrow \mathbb{R}$ determines the membership of $l_k$ to each instance such that $h(x) = \{l_k | f_k(x) > 0, 1 \leq k \leq q\}$. Here all functions in the family operate on the same feature set $x$. Although this strategy has gained much success in algorithm design, it might be too straightforward and away from being optimal for multi-label learning.

For example, in automatic image annotation, suppose $sky$ and $desert$ are two possible labels in the label space. Intuitively, $color$-based features would be preferred in discriminating $sky$ and non-$sky$ images, $texture$-based features would be preferred in discriminating desert and non-desert images, while both $color$- and $texture$-based features might be useful in discriminating other labels in the label space. For another example, in text categorization, features related to terms such as $government$, $reform$ and $president$ might be important in discriminating political and non-political documents, while features related to terms such as $stadium$, $matches$ and $champions$ might be important in discriminating sport and non-sport documents.

The above observations reflect the fact that, in multi-label learning different class labels in the label space may carry specific characteristics of their own. Therefore, we hypothesize that if label-specific features, i.e. the most pertinent and discriminative features for each class label, could be used in the learning process, a more effective solution to the problem of multi-label learning can be expected.

To justify this hypothesis, a new algorithm named LIFT, i.e. multi-label learning with Label specific Features, is proposed in this paper. LIFT tackles multi-label learning problem with two simple steps. Firstly, for each class label $l_k \in \mathcal{Y}$, clustering analysis is conducted on its positive instances as well as negative instances in the training set, and then features specific to $l_k$ are constructed by querying the clustering results. Secondly, a family of $q$ classifiers are induced where each of them is trained with the generated label-specific features other than the original features under $\mathcal{X}$. To thoroughly evaluate the effectiveness of LIFT, extensive experiments over sixteen regular-scale and large-scale data sets clearly verify that our approach compares favorably against other state-of-the-art multi-label learning algorithms.

$^*$This work was supported by the National Science Foundation of China (60805022), Ph.D. Programs Foundation of Ministry of Education of China for Young Faculties (200802941009), and the Startup Foundation of Southeast University.
2 Related Work

Generally, the task of multi-label learning is rather challenging due to the exponential number of possible label sets (i.e., \(2^m\)) to be predicted for an unseen example. To cope with this issue, existing approaches focus on exploiting the correlations between labels to facilitate the learning process, which can be roughly grouped into three categories based on the order of label correlations [Zhang and Zhou, 2010], namely first-order approaches, second-order approaches and high-order approaches.

First-order approaches tackle multi-label learning problem by decomposing it into a number of independent binary classification problems. Specifically, a family of \(q\) functions \(\{f_1, f_2, \ldots, f_q\}\) are learned in a label-by-label style, one for each possible class label [Boutell et al., 2004; Zhang and Zhou, 2007]. First-order approaches are conceptually simple, efficient and easy to implement, while may be less effective due to their ignorance of label correlations.

Second-order approaches tackle multi-label learning problem by considering pairwise relations between labels. Specifically, a family of \(q\) functions \(\{f_1, f_2, \ldots, f_q\}\) are learned by exploring the interactions between a pair of functions \(f_k\) and \(f_{k'}\) such as their co-occurrence patterns [Ghamrawi and McCallum, 2005; Zhu et al., 2005] or their ranking constraints [Elisseeff and Weston, 2002; Zhang and Zhou, 2006; Fürnkranz et al., 2008]. Second-order approaches address label correlations to some extent and are relatively effective, while may suffer from the fact that label correlations could go beyond second-order under real-world scenarios.

High-order approaches tackle multi-label learning problem by considering high-order relations between labels. Specifically, a family of \(q\) functions \(\{f_1, f_2, \ldots, f_q\}\) are learned by exploring the influences among a subset of functions \(\{f_{k_1}, f_{k_2}, \ldots, f_{k_q}\}\) (\(q' \leq q\)) [Toumoukas and Vlahavas, 2007; Read et al., 2009; Zhang and Zhang, 2010] or among all the functions [Yan et al., 2007; Ji et al., 2008; Cheng and Hültermeier, 2009]. High-order approaches could address more realistic label correlations, while may exhibit high model complexities.

A commonness of these existing approaches is that they deal with multi-label learning problem mainly from the perspective of output space, i.e. manipulating correlations between labels, and identical features inherited from the original input space are directly used in discriminating all the labels. In the next section, we will present the LIFT approach which works mainly from the perspective of input space, i.e. manipulating label-specific features.

3 The LIFT Approach

Let \(D = \{ (x_i, Y_i) | 1 \leq i \leq m \} \) be the training set with \(m\) multi-label training examples, where \(x_i \in \mathcal{X}\) is a \(d\)-dimensional feature vector and \(Y_i \subseteq \mathcal{Y}\) is the set of labels associated with \(x_i\). Then, LIFT learns from \(D\) following two elemental steps, i.e. label-specific features construction and classification models induction.

In the first step, LIFT aims to generate distinguishing features which capture the specific characteristics of each label to facilitate its discrimination process. To achieve this, LIFT employs clustering techniques which have been widely used as stand-alone tools to gain insights into the properties of data [Jain et al., 1999]. Specifically, with respect to each class label \(l_k \in \mathcal{Y}\), the set of positive training instances \(\mathcal{P}_k\) as well as the set of negative training instances \(\mathcal{N}_k\) are denoted as:

\[
\mathcal{P}_k = \{ x_i | (x_i, Y_i) \in D, l_k \in Y_i \} \\
\mathcal{N}_k = \{ x_i | (x_i, Y_i) \in D, l_k \notin Y_i \}
\]

In other words, \(\mathcal{P}_k\) and \(\mathcal{N}_k\) consist of the training instances in \(D\) with and without label \(l_k\) respectively.

The popular \(k\)-means algorithm [Jain et al., 1999] is adopted to partition \(\mathcal{P}_k\) into \(m^+\) disjoint clusters whose centers are denoted as \(\{p^+_1, p^+_2, \ldots, p^+_m\}\). Similarly, \(\mathcal{N}_k\) is also partitioned into \(m^-\) disjoint clusters whose centers are denoted as \(\{n^-_1, n^-_2, \ldots, n^-_m\}\). Here, we choose to set equivalent number of clusters for \(\mathcal{P}_k\) and \(\mathcal{N}_k\), i.e. \(m^+ = m^- = m_k\). In this way, clustering information gained from positive instances as well as negative instances are treated with equal importance. Specifically, the number of clusters retained for both positive and negative instances is set to be:

\[
m_k = r \cdot \min (|\mathcal{P}_k|, |\mathcal{N}_k|)
\]

where \(|\cdot|\) represents the set cardinality and \(r \in [0, 1]\) is the ratio parameter controlling the number of clusters thus retained.

Intuitively, the retained cluster centers characterize the underlying structure of input space, and can be used as the bases for label-specific features construction with regard to \(l_k\). In

---

**Table 1: Pseudo-code of LIFT.**

<table>
<thead>
<tr>
<th>Y</th>
<th>Lift(D, r, L, u)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs:</strong></td>
<td><strong>Outputs:</strong></td>
</tr>
<tr>
<td>D: the multi-label training set ({ (x_i, Y_i)</td>
<td>1 \leq i \leq m } ), (x_i \in \mathcal{X}), (Y_i \subseteq \mathcal{Y}), (\mathcal{X} = \mathbb{R}^d), (\mathcal{Y} = {l_1, l_2, \ldots, l_q})</td>
</tr>
<tr>
<td>r: the ratio parameter as used in Eq.(2)</td>
<td></td>
</tr>
<tr>
<td>L: the binary learner for classifier induction</td>
<td></td>
</tr>
<tr>
<td>u: the unseen example ((u \in \mathcal{X}))</td>
<td></td>
</tr>
</tbody>
</table>

**Process:**

1. for \(k = 1 \to q\) do
2. Form \(\mathcal{P}_k\) and \(\mathcal{N}_k\) based on \(D\) according to Eq.(1);
3. Perform \(k\)-means clustering on \(\mathcal{P}_k\) and \(\mathcal{N}_k\), each with \(m_k\) clusters as defined in Eq.(2);
4. Create the mapping \(\phi_k\) for \(l_k\) according to Eq.(3);
5. endfor
6. for \(k = 1 \to q\) do
7. Form \(D_k^+\) according to Eq.(4);
8. Induce \(f_k^+\) by invoking \(L\) on \(D_k^+\); i.e. \(f_k^+ \leftrightarrow L(D_k^+)\);
9. endfor
10. \(Y = \{ l_k | f_k^+(\phi_k(u)) > 0, 1 \leq k \leq q \}\).
LIFT creates a mapping \( \phi_k: \mathcal{X} \rightarrow \mathcal{Z}_k \) from the original \( d \)-dimensional space \( \mathcal{X} \) to the \( 2m_k \)-dimensional label-specific feature space \( \mathcal{Z}_k \) as follows:

\[
\phi_k(x) = \left[ d(x, p_{1k}^k), \ldots, d(x, p_{mk}^k), d(x, n_{1k}^k), \ldots, d(x, n_{mk}^k) \right]
\]

(3)

Here, \( d(\cdot, \cdot) \) returns the distance between two instances and is set to the Euclidean metric in this paper.

Note that we do not claim that the above process is the best possible practice for label-specific features construction. Actually, the mapping \( \phi_k \) can be implemented in alternative ways, such as setting different number of clusters for positive and negative instances (i.e. \( m_k^+ \neq m_k^- \)), utilizing more sophisticated distance for \( d(\cdot, \cdot) \) other than the Euclidean metric, or even employing \( k \)NN rule [Wang et al., 2010] to identify the bases for feature mapping other than invoking the \( k \)-means procedure, etc. Nevertheless, LIFT’s simple construction process of label-specific features suffices to yield competitive performance as shown in Section 4.

In the second step, LIFT aims to induce a family of \( q \) classification models \( \{f_{1k}, f_{2k}, \ldots, f_{qk}\} \), a binary training set \( D_k^* \) with \( m \) examples is created from \( D \) and \( \phi_k \) as follows:

\[
D_k^* = \{ (\phi_k(x_i), Y_i) \mid (x_i, Y_i) \in D \}
\]

where

\[
Y_i(k) = +1 \text{ if } l_k \in Y_i; \text{ Otherwise, } Y_i(k) = -1
\]

(4)

Based on \( D_k^* \), any binary learner can be applied to induce a classification model \( f_{ik}^* : \mathcal{Z}_k \rightarrow \mathbb{R} \) for \( l_k \). Given an unseen example \( u \in \mathcal{X} \), its associated label set is predicted as:

\[
Y = \{ l_k \mid f_{ik}^*(\phi_k(u)) > 0.1 \leq k \leq q \}
\]

In principle, the classifiers induction process of LIFT is similar to those of the first-order approaches discussed in Section 2. The major difference lies that LIFT induces the classifier on each label with label-specific features instead of the original ones.

In summary, Table 1 presents the complete description of LIFT. Based on the multi-label training examples, LIFT firstly constructs label-specific features for each possible class label (steps 1 to 5); After that, a family of \( q \) binary classification models are induced based on the constructed features in a round-robin style (steps 6 to 9); Finally, the unseen example is fed to the learned system for prediction (step 10).

Note that the idea of exploiting label-specific features should be regarded a meta-strategy to multi-label learning, and LIFT serves as one possible instantiation of this strategy. There are learning algorithms such as decision trees which can implicitly do feature selection during their learning process. However, this is only a by-product of these algorithms and should not be regarded a stand-alone tool for feature construction. Employing algorithms such as decision trees in first-order approaches could yield similar methods to LIFT. However, the major difference lies in that the former one wraps around the construction of label-specific features and the induction of learning models, while LIFT constructs label-specific features independent of the model induction.

### 4 Experiments

#### 4.1 Configuration

To thoroughly evaluate the performance of our approach, a total of sixteen real-world multi-label data sets are studied in this paper. For each data set \( \mathcal{S} = \{ (x_i, Y_i) \mid 1 \leq i \leq \rho \} \), we use \( |\mathcal{S}|, \dim(S), L(S) \) and \( F(S) \) to denote the properties of number of examples, number of features, number of possible class labels, and feature type respectively. In addition, several other multi-label properties [Read et al., 2009; Tsoumakas et al., 2009] are denoted as:
multi-label learning algorithms, including B SVM [Boutell et al., 2004], ML-KNN [Zhang and Zhou, 2007], BP-MLL [Zhang and Zhou, 2006] and ECC [Read et al., 2009]. As discussed in Section 2, B SVM and ML-KNN are first-order approaches which work in a similar way as LIFT by generating classifiers in a label-wise style, though under the original feature space $X$. In addition, BP-MLL is a second-order approach and ECC is a high-order approach both of which consider exploiting label correlations in their learning processes. For fair comparison, LIBsvm (with linear kernel) [Chang and Lin, 2001] is employed as the binary learner for classifier induction to instantiate LIFT, B SVM and ECC.

As shown in Table 1, in addition to set $\mathcal{L}$ as linear kernel LIBsvm, the other factor needed to be specified for LIFT is $r$, i.e. the ratio parameter as used in Eq.(2). In this paper, $r$ is set to be 0.1 for all data sets. Furthermore, parameter configurations suggested in respective literatures are used for the other comparing algorithms. For B SVM, models are learned via the cross-training strategy [Boutell et al., 2004]; For ML-KNN, the number of nearest neighbors considered is set to be 10 [Zhang and Zhou, 2007]; For BP-MLL, the number of hidden neurons is set to be 20% of the input dimensionality and the maximum number of training epochs is set to be 100 [Zhang and Zhou, 2006]; For ECC, the ensemble size is set to be 10 and the sampling ratio is set to be 67% [Read et al., 2009].

### 4.2 Results

Performance evaluation in multi-label learning is more complicated than traditional single-label learning, as each example could be associated with multiple labels simultaneously. Firstly, five evaluation criteria popularly used in multi-label learning are employed, i.e. hamming loss, one-error, coverage, ranking loss and average precision [Schapire and Singer, 2000; Tsoumakas et al., 2009]. Briefly, these criteria evaluate the quality of the predicted label set for each test example and then return the averaged value across all the test examples. In addition, we also employ the AUC criterion (area under the ROC curve) to measure the performance of the methods.

In preliminary experiments, a number of values have been tested for $r$ by increasing it from 0.02 to 0.2 (stepsize 0.02). Results show that the performance of LIFT becomes stable as $r$ approaches 0.1.

Due to page limit, formal definitions on these criteria can be found in the references therein. In this paper, the coverage criterion is further normalized by $|\mathcal{Y}|$ so that all reported performance measures vary between [0,1].
under the ROC curve) to evaluate the quality of the predictions for each class label and then return the averaged value across all the class labels.

On each data set, ten-fold cross validation is conducted and the mean value and standard deviation of each evaluation criterion is recorded. For AUC and average precision, the larger the values the better the performance. While for the other four criteria, the smaller the values the better the performance. All the above criteria serve as good indicators for comprehensive comparative studies as they evaluate the performance of an algorithm from various aspects.

Tables 3 and 4 report the detailed experimental results on the regular-scale and large-scale data sets respectively. For each evaluation criterion, “↑” indicates “the smaller the better” while “↓” indicates “the larger the better”. Furthermore, the best performance among the five comparing algorithms is highlighted in boldface.

Across all the 96 configurations (i.e. 16 data sets × 6 criteria as shown in the two tables), Lift ranks in 1st place among the five comparing algorithms at 47.9% cases, in 2nd place at 45.8% cases, in 3rd place at only 6.3% cases, and never ranks in 4th and 5th places. To perform comparative analysis in more well-founded ways, Friedman test is further examined which is a favorable statistical test for comparisons of more than two algorithms over multiple data sets [Demšar, 2006].

For each evaluation criterion, Friedman test at 0.05 significance level (five algorithms, sixteen data sets) rejects the null hypothesis of “equal” performance, which leads to the use of post-hoc tests to find out which algorithms actually differ. Specifically, Nemenyi test is utilized where the performance of two algorithms is significantly different if their average ranks over all data sets differ by at least one critical difference (CD). Figure 1 gives the CD diagrams [Demšar, 2006] for each evaluation criterion, where the average rank of each comparing algorithm is marked along the axis (lower ranks to the right). Groups of algorithms that are not significantly different according to Nemenyi test are connected with a thick line. The critical difference (CD=1.525 at 0.05 significance level) is also shown above the axis in each subfigure.

To summarize, out of all the 24 comparisons (4 algorithms to compare × 6 criteria), Lift achieves statistically comparable performance in only 33% cases, i.e. the 8 comparisons against ML-KNN on hamming loss (Fig.1 a), against ECC on one-error (Fig.1 b), against BVM and BP-MLL on coverage (Fig.1 c), on ranking loss (Fig.1 d), and on AUC (Fig.1 f). Rather impressively, Lift achieves statistically superior performance in all the other 67% cases and no algorithms have once outperformed Lift. These results clearly validate the superior effectiveness of our approach to the other well-established multi-label learning algorithms.

5 Conclusion

The major contributions of our work are two-fold: 1) A new strategy to multi-label learning via label-specific features is proposed, which suggests a new direction for learning from multi-label data; 2) A simple algorithm named Lift is designed to justify the proposed strategy, whose effectiveness is thoroughly validated based on extensive comparative studies.

In the future, it is interesting to see whether Lift can be further improved by incorporating label correlations into its
label-specific features construction step as well as classification models induction step. Furthermore, designing other ways to fulfill the strategy of label-specific features is a direction very worth studying.

Acknowledgments

The author wishes to thank the anonymous reviewers for their helpful comments and suggestions.

References


