Partial Label Learning via Feature-Aware Disambiguation

Min-Ling Zhang\textsuperscript{1,2}, Bin-Bin Zhou\textsuperscript{1,2}, Xu-Ying Liu\textsuperscript{1,2}
\textsuperscript{1} School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
\textsuperscript{2}Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China
\{zhangml, zhoubinbin, liuxy\}@seu.edu.cn

ABSTRACT
Partial label learning deals with the problem where each training example is represented by a feature vector while associated with a set of candidate labels, among which only one label is valid. To learn from such ambiguous labeling information, the key is to try to disambiguate the candidate label sets of partial label training examples. Existing disambiguation strategies work by either identifying the ground-truth label iteratively or treating each candidate label equally. Nonetheless, the disambiguation process is generally conducted by focusing on manipulating the label space, and thus ignores making full use of potentially useful information from the feature space. In this paper, a novel two-stage approach is proposed to learning from partial label examples based on feature-aware disambiguation. In the first stage, the manifold structure of feature space is utilized to generate normalized labeling confidences over candidate label set. In the second stage, the predictive model is learned by performing regularized multi-output regression over the feature space and then utilized to generate normalized labeling confidences over candidate label set. The key for successful partial label learning is therefore trying to disambiguate the set of candidate labels, where existing strategies include disambiguation by identification or disambiguation by averaging. For identification-based disambiguation, the ground-truth label is regarded as latent variable and identified through iterative refining procedure such as EM\textsuperscript{[5, 15, 16, 18, 24]}, for averaging-based disambiguation, all the candidate labels are treated equally and the prediction is made by averaging their modeling outputs\textsuperscript{[8, 13, 27]}. By taking specific views on the candidate labels, both of the existing strategies conduct disambiguation by only focusing on the manipulation of label space. Nonetheless, it is natural to postulate that the potentially useful information from feature space should also be exploited to facilitating the disambiguation process. Specifically, to help disambiguate the candidate label set, one might make use of the smoothness assumption that examples close to each other in the feature space will tend to share identical label in the label space. For instance, suppose we have three PL training examples \((x_1, \{y_2, y_3\}), (x', \{y_1, y_2\})\) and \((x'', \{y_3, y_4\})\) where \(x\) is shown to be close to \(x'\) while far from \(x''\) in the feature space. Then, it is reasonable to assign higher labeling confidence on \(y_2\) than \(y_3\) for \(x\) (i.e. \(P(y_2 | x) \geq P(y_3 | x)\)), as \(y_2\) is a shared candidate label between close instances \((x, x')\) while \(y_3\) is a shared candidate label between distant instances \((x, x'')\).

In light of the above observation, a novel two-stage approach named PL.-LEAF, i.e. Partial Label LEArning via Feature-aware disambiguation, is proposed in this paper. In the first stage, the manifold structure among training examples is analyzed in the feature space and then utilized to generate normalized labeling confidences over candidate label set. In the second stage, a multi-class predictive model is learned by fitting a regularized multi-output regressor with the generated labeling confidences. Extensive experiments on controlled UCI data sets as well as real-world PL data sets clearly show the effectiveness of feature-aware disambiguation for partial label learning.

The rest of this paper is organized as follows. Section 2 briefly
reviews related work in partial label learning. Section 3 introduces the proposed PL-LEAF approach. Section 4 reports experimental results of comparative studies. Finally, Section 5 concludes the paper and discusses future research issues.

2. RELATED WORK

Due to the ambiguous labeling information conveyed by PL training examples, partial label learning can be regarded as a weakly-supervised learning framework. It situates between two ends of the supervision spectrum, i.e. standard supervised learning with explicit supervision and unsupervised learning with blind supervision. Furthermore, partial label learning is related to other well-studied weakly-supervised learning frameworks, including semi-supervised learning, multi-instance learning and multi-label learning, while the weak supervision scenario for partial label learning is different to those counterpart frameworks.

Semi-supervised learning [4, 29] aims to learn a predictive model \( f: \mathcal{X} \rightarrow \mathcal{Y} \) from few labeled data together with abundant unlabeled data. For unlabeled data the ground-truth label assumes the entire label space, while for PL data the ground-truth label is confined within its candidate label set. Multi-label learning [11, 28] aims to learn a predictive model \( f: \mathcal{X} \rightarrow 2^{\mathcal{Y}} \) from training examples each associated with multiple labels. For multi-label data the associated labels are all valid ones, while for PL data the associated labels are only candidate ones.

Existing approaches learn from PL training examples by trying to disambiguate their candidate label sets. One disambiguation strategy is to assume certain parametric model \( F(\mathbf{x}, y; \theta) \) where the ground-truth label is regarded as latent variable and identified as \( \hat{y}_i = \arg \max_{y \in S_i} F(\mathbf{x}_i, y; \theta). \) Generally, the latent variable is refined iteratively via EM procedure which optimizes objective function defined according to the maximum likelihood criterion:

\[
\sum_{i=1}^{m} \log \left( \sum_{y \in S_i} F(\mathbf{x}_i, y; \theta) \right) + \| \mathbf{x}_i - \mathbf{y} \|_2^2 \tag{1}
\]

One potential drawback of the identification-based disambiguation strategy lies in that, rather than recovering the ground-truth label \( y_i, \) the identified label \( \hat{y}_i \) might turn out to be false positive label in the candidate label set (i.e. \( S_i \setminus \{ y_i \} \)).

Another disambiguation strategy is to assume equal importance of each candidate label and then make prediction by averaging their modeling outputs. Under discriminative learning setting, the averaged output from all candidate labels, i.e. \( \frac{1}{|S_i|} \sum_{y \in S_i} F(\mathbf{x}_i, y; \theta), \) is distinguished from the outputs from non-candidate labels, i.e. \( F(\mathbf{x}_i, y; \theta) \ (y \notin S_i). \) Under instance-based learning setting, the predicted label for unseen instance is determined by averaging the candidate labeling information from its neighboring examples in the PL training set. One potential drawback of the averaging-based disambiguation strategy lies in that the essential modeling output from ground-truth label \( y_i, \) might be overwhelmed by the distractive outputs from false positive labels.

In the next section, a novel partial label learning approach named PL-LEAF will be introduced. Other than disambiguation by identification or averaging, PL-LEAF facilitates the disambiguation process by making use of local topological information from the feature space. The candidate label set is disambiguated in the form of normalized labeling confidences, which differs from the crisp-style disambiguation of identifying a single candidate label or the uniform-style disambiguation of averaging all candidate labels.

3. THE PROPOSED APPROACH

As shown in Section 1, the task of partial label learning is to learn a multi-class classifier \( f: \mathcal{X} \rightarrow \mathcal{Y} \) from the PL training set \( D = \{(\mathbf{x}_i, S_i) \mid 1 \leq i \leq m\}. \) For the proposed PL-LEAF approach, its key novelty lies in how the disambiguation procedure is conducted.

Specifically, for each PL training example \( (\mathbf{x}_i, S_i), \) PL-LEAF aims to disambiguate its candidate label set \( S_i \) via a normalized real-valued vector \( \lambda_i = (\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{ik})^\top. \) Here, each component \( \lambda_{ik} \) represents the labeling confidence of \( y_k \) being the ground-truth label for \( \mathbf{x}_i, \) which satisfies the following constraints:

- \( \lambda_{ik} = 0 \ (\forall y_k \notin S_i) \)
- \( \lambda_{ik} \geq 0 \ (\forall y_k \in S_i) \) and \( \sum_{y_k \in S_i} \lambda_{ik} = 1 \)

Once the normalized labeling confidence vectors have been generated, the predictive model will be induced by utilizing the disambiguation results.

In the next subsections, the two basic stages of PL-LEAF, i.e. feature-aware disambiguation and predictive model induction, will be scrutinized respectively.

3.1 Feature-Aware Disambiguation

In the first stage, PL-LEAF aims to disambiguate the candidate label set by exploiting useful information from the feature space. To fulfill this task, the popular smoothness assumption is adopted to enable information exploitation from the feature space to the label space. Given the PL training set \( D = \{(\mathbf{x}_i, S_i) \mid 1 \leq i \leq m\}, \) a weighted graph \( G = (V, E, \mathbf{W}) \) is constructed over the training examples to characterize the manifold structure of feature space. Here, \( V = \{\mathbf{x}_i \mid 1 \leq i \leq m\} \) corresponds to the set of vertices and \( E = \{(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i, \mathbf{x}_j \in \text{KNN}(\mathbf{x}_i), \ i \neq j \} \) corresponds to the set of directed edges from \( \mathbf{x}_i \) to \( \mathbf{x}_j \) if \( \mathbf{x}_i \) is among the \( k \)-nearest neighbors of \( \mathbf{x}_j. \) Furthermore, \( \mathbf{W} = [W_{ij}]_{m \times m} \) corresponds to the non-negative weight matrix where \( W_{ij} = 0 \) if \( (\mathbf{x}_i, \mathbf{x}_j) \notin E. \) For the weight matrix, its \( j \)-th column \( \mathbf{w}_j = (W_{1j}, W_{2j}, \ldots, W_{mj})^\top \) is determined by solving the following linear least square problem:

\[
\min_{\mathbf{w}_j} \left\| \mathbf{x}_j - \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in E} W_{ij} \mathbf{w}_j \right\|_2^2 \tag{1}
\]

subject to:

- \( \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in E} W_{ij} = 1 \)
- \( W_{ij} \geq 0 \ (\forall (\mathbf{x}_i, \mathbf{x}_j) \in E) \)

Conceptually, the weight value \( W_{ij} ((\mathbf{x}_i, \mathbf{x}_j) \in E) \) characterizes the relative importance of neighboring example \( \mathbf{x}_i \) in reconstructing \( \mathbf{x}_j. \) According to the constraint \( \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in E} W_{ij} = 1, \) the above optimization problem (OP) can be re-written as:

\[
\min_{\mathbf{w}_j} \mathbf{w}_j^\top \mathbf{G}_j \mathbf{w}_j \tag{2}
\]

subject to:

- \( \mathbf{1}^\top \mathbf{w}_j = 1 \)
- \( W_{ij} \geq 0 \ (\forall (\mathbf{x}_i, \mathbf{x}_j) \in E) \)
- \( W_{ij} = 0 \ (\forall (\mathbf{x}_i, \mathbf{x}_j) \notin E) \)

Here, \( \mathbf{G}_j = [G_{ij}]_{m \times m} \) is the local Gram matrix for \( \mathbf{x}_j \) with elements \( G_{ij} = (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^\top. \) Apparently, OP(2) corresponds to a standard quadratic programming (QP) problem whose optimal solution can be obtained by any off-the-shelf QP solver. The weight matrix \( \mathbf{W} \) is constructed by solving OP(2) column-wisely and the resulting matrix is generally not symmetric. Accordingly, based on the local topological information embodied in \( \mathbf{G}, \) the manifold structure in the feature space will be exploited to help disambiguate the candidate label set.

Here, in order to generate the labeling confidence vectors \( \Lambda = [\lambda_1, \lambda_2, \ldots, \lambda_m], \) PL-LEAF exploits the smoothness assumption...
Table 1: The pseudo-code of PL-LEAF.

Inputs:
- $D$: the partial label training set $\{(x_i, S_i) \mid 1 \leq i \leq m\}$ ($x_i \in \chi, S_i \subseteq \psi, \chi = \mathbb{R}^d, \psi = \{y_1, y_2, \ldots, y_q\}$)
- $K$: the number nearest neighbors used for weighted graph construction
- $C_1, C_2$: the regularization parameters for regression loss function
- $x$: the unseen instance ($x \in \chi$)

Outputs:
- $y$: the predicted label for $x$

Process:
1. Set the weighted graph $G = (V, E, W)$ with $V = \{x_i \mid 1 \leq i \leq m\}$ and $E = \{(x_i, x_j) \mid x_i \in \text{KNN}(x_j), i \neq j\}$;
2. for $j = 1$ to $m$ do
3. Instantiate the $j$-th column $W_j$ of the weight matrix $W$ by solving OP(2) with QP procedure;
4. end for
5. Generate the labeling confidence vectors $A$ by solving OP(4) with QP procedure, or by solving OP(5) with alternating optimization;
6. Calculate the kernel matrix $K = [k(x, x_j)]_{m \times m}$ over training examples;
7. Calculate $\Psi = [\psi_1, \psi_2, \ldots, \psi_m]$ with $\psi_i = -\left(\frac{1}{|S_i|^2} \cdot 1_{S_i} - \frac{1}{|S_i|^2} \cdot 1_{\text{KNN}(x_i)}\right) (1 \leq i \leq m)$
8. Set $t = 0$;
9. Initialize $\Theta^{(0)}$ and $b^{(0)}$ with $\alpha_k^{(0)} = 0$ and $b_k^{(0)} = 0 (1 \leq k \leq q)$;
10. repeat
11. Calculate $\rho = [\rho_1, \rho_2, \ldots, \rho_m]^T$ according to Eq.(11), and set $D_\rho = [d_{ij}]_{m \times m}$ with $d_{ij} = \rho_i \delta_{ij}$;
12. for $k = 1$ to $q$ do
13. Obtain the solution $\alpha_k$ and $b_k$ by solving Eq.(17);
14. end for
15. Set the descending direction $P^{(t)}$ according to Eq.(16);
16. Update $\{\Theta^{(t+1)}, b^{(t+1)}\}$ by invoking line search procedure from $\{\Theta^{(t)}, b^{(t)}\}$ along $P^{(t)}$;
17. $t = t + 1$;
18. until $L(\Theta^{(t-1)}, b^{(t-1)}) - L(\Theta^{(t)}, b^{(t)}) < \tau \cdot L(\Theta^{(t-1)}, b^{(t-1)})$
19. Set the final predictive model with $\Theta^* = \Theta^{(t)}$ and $b^* = b^{(t)}$;
20. Return $y = f(x)$ according to Eq.(18).

That the manifold structure in the feature space should also be preserved in the label space:

$$\min_{\lambda} \sum_{j=1}^{m} \left\| \lambda_j - \sum_{(x_i, x_j) \in E} W_{ij} \cdot \lambda_i \right\|_2^2 \quad (3)$$

s.t.: $\lambda_{jk} = 0$ ($\forall 1 \leq j \leq m, y_k \notin S_j$)
$\lambda_{jk} \geq 0$ ($\forall 1 \leq j \leq m, y_k \in S_j$)
$\sum_{y_k \in S_j} \lambda_{jk} = 1$ ($\forall 1 \leq j \leq m$)

According to the constraint $\sum_{(x_i, x_j) \in E} W_{ij} = 1$, the above OP(3) can be re-written as:

$$\min_{\lambda} \sum_{i=1}^{m} \sum_{j=1}^{m} \gamma_{ij} \cdot \lambda_i^T \cdot \lambda_j \quad (4)$$

s.t.: $\lambda_{jk} = 0$ ($\forall 1 \leq j \leq m, y_k \notin S_j$)
$\lambda_{jk} \geq 0$ ($\forall 1 \leq j \leq m, y_k \in S_j$)
$\sum_{y_k \in S_j} \lambda_{jk} = 1$ ($\forall 1 \leq j \leq m$)

Here, $\gamma_{ij} = W_{ij}^T + \delta_{ij} W_{ii} - 2W_{ij}$ where $W_{ii}$, $\delta_{ij}$ and $1_{m \times m}$ represent the $i$-th row of weight matrix $W$, the Kronecker's delta and the $m \times m$ matrix of 1's respectively.

Note that OP(4) corresponds to a standard QP problem with $mq$ variables and $m(q + 1)$ constraints, whose computational complexity would be demanding when $mq$ is large. To improve efficiency, we can choose to solve OP(4) with alternating optimization strategy where a series of QP subproblems with $q$ variables and $q + 1$ constraints are optimized iteratively. Without loss of generality, in each alternating optimization iteration, one labeling confidence vector $\lambda_j$ is optimized by fixing the values of other labeling confidence vectors $\lambda_k (j \neq k)$:

$$\min_{\lambda_j} \sum_{i=1}^{m} \sum_{j=1, j \neq i}^{m} (r_{ij} + r_{ji}) \cdot \lambda_i^T \cdot \lambda_j + \sum_{y_k \in S_i} \lambda_{jk} - 1 \quad (5)$$

s.t.: $\lambda_{ik} = 0$ ($\forall y_k \notin S_i$)
$\lambda_{ik} \geq 0$ ($\forall y_k \in S_i$)
$\sum_{y_k \in S_i} \lambda_{ik} = 1$

It is interesting to notice that, to some extent, the disambiguation results returned by existing strategies can be viewed as simplified versions of PL-LEAF’s normalized labeling confidence vectors. For identification-based disambiguation [16, 18], a single label $y_k$ in $S_i$ is identified as the ground-truth label leading to unimodal labeling confidence vector with $\lambda_{ik} = 1$ and $\lambda_{ik} = 0 (k \neq k)$. For averaging-based disambiguation [8, 13], all candidate labels in $S_i$ are treated equally leading to uniform labeling confidence vector with $\lambda_{ik} = \frac{1}{|S_i|} (y_k \in S_i)$. Therefore, the normalized labeling confidence vector considered by PL-LEAF would accommodate...
more flexibility in modeling the disambiguation results compared to the unimodal or uniform setup.

### 3.2 Predictive Model Induction

Following the first stage of feature-aware disambiguation, the original PL training set $\mathcal{D}$ has been transformed into its disambiguated counterpart: $\mathcal{D}_{\text{dis}} = \{(\mathbf{x}_i, \lambda_i) \mid 1 \leq i \leq m\}$. In the second stage, PL-LEAF aims to induce the predictive model $f: \mathcal{X} \mapsto \mathcal{Y}$ based on $\mathcal{D}_{\text{dis}}$. Considering that the response variables (normalized labeling confidences) for each training example in $\mathcal{D}_{\text{dis}}$ are actually real-valued, it is natural to induce the predictive model by performing multi-output regression. Among various choices of multi-output regression techniques, we choose to adapt the multi-regression support vector machines (MSVR) [6, 21, 23] such that kernel trick can be readily incorporated to accommodate nonlinear modeling.

Let $\phi(\cdot): \mathbb{R}^d \mapsto \mathbb{R}^n$ be the (implicit) nonlinear mapping from the original feature space to the higher-dimensional Reproducing Kernel Hilbert Space (RKHS) via kernel function $\kappa: \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. Furthermore, let $\{(\mathbf{t}_k, b_k) \mid 1 \leq k \leq q\}$ denote the multi-output regression model in the RKHS with one linear predictor $(\mathbf{t}_k, b_k)$ for each class label $y_k \in \mathcal{Y}$. Then, PL-LEAF induces the regression model by minimizing the following loss function:

$$L(\Theta, \mathbf{b}) = \frac{1}{2} \sum_{k=1}^{q} ||\mathbf{t}_k||^2 + C_1 \sum_{i=1}^{m} L_1(u_i) + C_2 \sum_{i=1}^{m} v_i$$

(6)

Here, $\Theta = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_q]$ and $\mathbf{b} = [b_1, b_2, \ldots, b_q]^\top$ represent the regression model’s weight matrix and bias vector respectively.

As shown in Eq.(6), the first term of $L(\Theta, \mathbf{b})$ controls the complexity of the induced model. In addition, the second term of $L(\Theta, \mathbf{b})$ is defined based on the $\epsilon$-insensitive loss function:

$$L_1(u) = \begin{cases} 0, & u < \epsilon \\ (u - \epsilon)^2, & u \geq \epsilon \end{cases}$$

(7)

For each example $(\mathbf{x}_i, \lambda_i)$ in $\mathcal{D}_{\text{dis}}$, the corresponding input to the $\epsilon$-insensitive loss function $L_1(\cdot)$ is set as:$u_i = ||\mathbf{e}_i|| = \sqrt{\mathbf{e}_i \cdot \mathbf{e}_i}$ with $\mathbf{e}_i = \lambda_i - \Theta^\top \phi(\mathbf{x}_i) - \mathbf{b}$. In this way, the outputs of all linear predictors are considered simultaneously to yield a unique input to $L_1(\cdot)$ such that the dependencies among all the class labels can be exploited by the $\epsilon$-insensitive term. The third term of $L(\Theta, \mathbf{b})$ considers the partial label loss for each example which is set as:

$$v_i = -\left(\mathbf{t}_i \cdot 1_{S_i}^\top - \frac{1}{|S_i|} \cdot 1_{S_i}^\top \cdot \mathbf{t}_i \cdot \phi(\mathbf{x}_i) + \mathbf{b}\right)$$

(8)

Here, for candidate label set $S_i$ and its complementary set $\bar{S}_i$ in $\mathcal{Y}$, $1_{S_i}$ ($1_{\bar{S}_i}$) corresponds to a $q$-dimensional vector whose $k$-th element equals to 1 if $y_k \in S_i$ ($y_k \in \bar{S}_i$) and 0 otherwise. In other words, the third term enforces the property that the average output from candidate labels should be larger than the average output from non-candidate ones, which has been widely-used in designing effective partial label learning algorithms [8, 13, 26].

To minimize $L(\Theta, \mathbf{b})$, PL-LEAF employs the gradient-based iterative method named Iterative Re-Weighted Least Square (IRWLS) [21, 23]. Specifically, in each iteration the descending direction is determined analytically by solving linear systems of equations. Let $\{\Theta(t), b(t)\}$ denote the current model after $t$-th iteration, the $\epsilon$-insensitive function $L_1(u_i)$ is firstly approximated by its first-order Taylor expansion:

$$\tilde{L}_1(u_i) = L_1(u_i(t)) + \frac{dL_1(u_i)}{du} \bigg|_{u_i(t)} \cdot \frac{(e(t))^\top}{u_i(t)} \cdot (e_i - e_i(t))$$

(9)

Here, $u_i(t)$ and $e_i(t)$ are calculated based on the current model $\{\Theta(t), b(t)\}$. Then, a quadratic approximation to $\tilde{L}(u_i)$ is further constructed to ease analytical solution to the descending direction:

$$\tilde{L}_1(u_i) = L_1(u_i(t)) + \frac{dL_1(u_i)}{du} \bigg|_{u_i(t)} \cdot \frac{u_i^2 - (u_i(t))^2}{2u_i(t)}$$

(10)

where

$$\rho_i = \frac{1}{u_i(t)} \frac{L_1(u_i)}{du} \bigg|_{u_i(t)} = \begin{cases} 0, & u_i(t) < \epsilon \\ \frac{2(u_i(t) - \epsilon)}{u_i(t)^2}, & u_i(t) \geq \epsilon \end{cases}$$

(11)

and $T_i$ is a constant which does not depend on $\{\Theta, \mathbf{b}\}$. 

<p>| Table 2: Characteristics of the experimental data sets. |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|</p>
<table>
<thead>
<tr>
<th>Data set</th>
<th># Examples</th>
<th># Features</th>
<th># Class Labels</th>
<th>Avg. # CLs</th>
<th>Task Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>FG-NET</td>
<td>1,002</td>
<td>262</td>
<td>78</td>
<td>7.48</td>
<td>facial age estimation [20]</td>
</tr>
<tr>
<td>Lost</td>
<td>1,122</td>
<td>108</td>
<td>16</td>
<td>2.23</td>
<td>automatic face naming [8]</td>
</tr>
<tr>
<td>MSRCv2</td>
<td>1,758</td>
<td>48</td>
<td>23</td>
<td>3.16</td>
<td>object classification [16]</td>
</tr>
<tr>
<td>BirdSong</td>
<td>4,998</td>
<td>38</td>
<td>13</td>
<td>2.18</td>
<td>bird song classification [3]</td>
</tr>
<tr>
<td>Soccer Player</td>
<td>17,472</td>
<td>279</td>
<td>171</td>
<td>2.09</td>
<td>automatic face naming [25]</td>
</tr>
<tr>
<td>Yahoo! News</td>
<td>22,991</td>
<td>163</td>
<td>219</td>
<td>1.91</td>
<td>automatic face naming [12]</td>
</tr>
</tbody>
</table>

Illuminating, the predictive model

(II) $r = 1, p \in \{0.1, 0.2, \ldots, 0.7\}$ [Figure 2]

(III) $r = 2, p \in \{0.1, 0.2, \ldots, 0.7\}$ [Figure 3]

(IV) $r = 3, p \in \{0.1, 0.2, \ldots, 0.7\}$ [Figure 4]
equating the corresponding gradient to zero:

\[ \frac{\partial L}{\partial \theta_k} = 0 \]

whose solution for each

\[ (14) \]

can be expressed as a linear system of equations:

\[ \sum_{i=1}^{m} \rho_i \phi(x_i)(\lambda_{ik} - \phi(x_i)^T \theta_k - b_k) + C_2 \sum_{i=1}^{m} \psi_{ik} \phi(x_i) = 0 \]

In contrast to standard least square objective function \( \sum_{k=1}^{q} \theta_k^2 \), the minimization of \( L(\theta, b) \) can be regarded as a weighted least square problem along with partial label loss regularization. Minimization of \( L(\theta, b) \) can be decoupled for each class label, whose solution for each \( (\theta_k, b_k) \) (1 \( \leq k \leq q \)) is found by equating the corresponding gradient to zero:

\[ \nabla_{\theta_k} L(\theta, b) = \theta_k - \left( C_1 \sum_{i=1}^{m} \rho_i \phi(x_i)(\lambda_{ik} - \phi(x_i)^T \theta_k - b_k) + C_2 \sum_{i=1}^{m} \psi_{ik} \phi(x_i) \right) \]

\[ \nabla_{b_k} L(\theta, b) = \]

\[ -C_1 \sum_{i=1}^{m} \rho_i (\lambda_{ik} - \phi(x_i)^T \theta_k - b_k) + C_2 \sum_{i=1}^{m} \psi_{ik} = 0 \]

Here, \( \psi_{ik} \) corresponds to the \( k \)-th component of the \( q \)-dimensional vector \( \psi_k = -\left( \frac{1}{\|S_k\|} \cdot 1_{S_k} - \frac{1}{\|S_I\|} \cdot 1_{S_I} \right) \). Accordingly, Eqs.(13) and (14) can be expressed as a linear system of equations:

\[
\begin{bmatrix}
C_1 \Phi^T D_\rho \Phi + I & C_1 \Phi^T \rho \\
C_1 \rho^T \Phi & C_1 \Phi^T \rho
\end{bmatrix}
\begin{bmatrix}
\theta_k \\
b_k
\end{bmatrix} =
\begin{bmatrix}
C_1 \Phi^T D_\rho \Phi \lambda_k - C_2 \Phi^T \rho \psi_k \\
C_1 \rho^T \Phi \lambda_k - C_2 \Phi^T \rho \psi_k
\end{bmatrix}
\]

Here, \( \Phi = [\phi(x_1), \phi(x_2), \ldots, \phi(x_m)]^T \), \( D_\rho = [d_{ij}]_{m \times m} \) with \( d_{ij} = \rho_i \delta_{ij}, \rho = [\rho_1, \rho_2, \ldots, \rho_m]^T, \lambda^* = [\lambda_{1k}, \lambda_{2k}, \ldots, \lambda_{mk}]^T, \psi^k = [\psi_{1k}, \psi_{2k}, \ldots, \psi_{mk}]^T. \)

Let \( \bar{\Theta} = [\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_m] \) and \( \bar{b} = [\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_m]^T \) be the solution obtained by solving Eq.(15) for each class label, the descending direction for the next iteration would be:

\[ \mathbf{P} = \begin{bmatrix} \bar{\Theta} - \Theta(t) \\ (\bar{b} - b(t))^T \end{bmatrix} \]

The subsequent model \( \{\Theta(t+\theta), b(t+\theta)\} \) is then updated by invoking line search procedure from \( \{\Theta(t), b(t)\} \) along the descending direction \( \mathbf{P}(t) \) [19].

According to the Representer Theorem [22], under fairly general conditions, the predictive model can be expressed by a linear combination of the training examples in the RKHS, i.e. \( \theta_k = \sum_{i=1}^{m} \alpha_{ik} \phi(x_i) = \Phi^T \alpha_k. \)

By introducing the kernel trick into Eqs.(13) and (14), the linear system of Eq.(15) can be expressed as follows:

\[ \begin{bmatrix}
C_1 K + D_\rho^{-1} & C_1 1 \\
C_1 \rho^T K & C_1 \rho^T \rho
\end{bmatrix}
\begin{bmatrix}
\alpha_k \\
b_k
\end{bmatrix} =
\begin{bmatrix}
C_1 \Phi \lambda^* - C_2 \Phi^T \rho \psi_k \\
C_1 \rho^T \Phi \lambda^* - C_2 \Phi^T \rho \psi_k
\end{bmatrix}
\]

Here, \( K = [\kappa(x_i, x_j)]_{m \times m} \) is the kernel matrix over training examples. Based on the kernel trick, the line search procedure can be readily performed in terms of \( \alpha_k \) and \( b_k \) as well.

Let \( \Theta^* \) and \( b^* \) be the resulting model after the whole iterative optimization process, PL-LEAF makes prediction on the class label of unseen instance \( x \) as follows:

\[
f(x) = \arg \max_{y \in Y} \theta_k^T \phi(x) + b_k
\]

\[
= \arg \max_{y \in Y} \sum_{i=1}^{m} \alpha_{ik} \kappa(x_i, x) + b_k
\]
Table 1 summarizes the pseudo-code of PL-LEAF.\(^2\) Given the PL training set, a weighted graph is constructed to characterize the manifold structure of feature space which is then utilized to disambiguate the candidate label set (Steps 1-5). After that, a predictive model based on kernelized multiregression SVR is learned via gradient-based iterative optimization (Steps 6-18).\(^3\) Finally, prediction on the unseen instance is made via the learned predictive model (Steps 19-20).

4. EXPERIMENTS

4.1 Experimental Setup

To evaluate the performance of PL-LEAF, two series of comparative experiments are conducted on controlled UCI data sets \(^2\) as well as real-world partial label data sets. Characteristics of the experimental data sets are summarized in Table 2.

Following the widely-used controlling protocol in partial label learning research [5, 8, 16, 24, 27], an artificial PL data set can be generated from a multi-class UCI data set with three controlling parameters \(p, r\) and \(\varepsilon\). Here, \(p\) controls the proportion of examples which are partially labeled (i.e. \(|S_i| > 1\)), \(r\) controls the number of false positive labels in the candidate label set (i.e. \(|S_i| = r + 1\)), and \(\varepsilon\) controls the co-occurring probability between one coupling candidate label and the ground-truth label. As shown in Table 2, a total of 28 (4x7) parameter configurations have been considered for each UCI data set.

In addition to artificial data sets, a number of real-world PL data sets have been collected from several task domains.\(^4\) For the task of facial age estimation (FG-NET [20]), human faces with landmarks are represented as instances while ages annotated by ten crowdsourced labelers together with the ground-truth age are regarded as candidate labels. For the task of automatic face naming (Lost [8], Soccer Player [25] and Yahoo! News [12]), faces cropped from an image or video frame are represented as instances while names extracted from the associated captions or subtitles are regarded as candidate labels. For the task of bird song classification (BirdSong [3]), singing syllables of the birds are represented as instances while bird species jointly singing during a 10-seconds period are regarded as candidate labels. For the task of object classification (MSRCv2 [16]), image segmentations are represented as instances while objects appearing within the same image are regarded as candidate labels. The average number of candidate labels (Avg. #CLs) for each real-world PL data set is also recorded in Table 2.

To show the effectiveness of feature-aware disambiguation, PL-LEAF is compared against four state-of-the-art partial label learning approaches with diverse properties, each configured with parameters suggested in respective literatures:

\begin{itemize}
  \item PL-KNN [13]: A K-nearest neighbor approach to partial label learning via averaging-based disambiguation [suggested configuration: \(K = 10\)];
  \item CLPL [8]: A discriminative approach to partial label learning via averaging-based disambiguation [suggested configuration: SVM with squared hinge loss];
  \item PL-SVM [18]: A maximum-margin approach to partial label learning via identification-based disambiguation [suggested configuration: SVM with squared hinge loss];
\end{itemize}

\(^2\)Code package for PL-LEAF is publicly-available at: http://cse.seu.edu.cn/PersonalPage/zhangml/Resources.htm#kdd16

\(^3\)In this paper, the \(\varepsilon\)-insensitive function \(L_\varepsilon(\cdot)\) is instantiated with \(\varepsilon = 0.1\) and the convergence condition in Step 18 is instantiated with \(\tau = 10^{-10}\).

\(^4\)These data sets are publicly-available at: http://cse.seu.edu.cn/PersonalPage/zhangml/Resources.htm#partial_data
configuration: regularization parameter pool with $\{10^{-3}, \ldots, 10^{3}\}$.

- **LSB-CMM [16]**: A maximum-likelihood approach to partial label learning via identification-based disambiguation [suggested configuration: $q$ mixture components].

Parameters for PL-LEAF (Table 1) are set as $K = 10$, $C_1 = 10$ and $C_2 = 1$. In this paper, ten runs of 50%/50% random train/test splits are performed on each artificial as well as real-world PL data set. Accordingly, the mean predictive accuracies (with standard deviation) are recorded for all comparing algorithms.

### 4.2 Experimental Results

#### 4.2.1 Controlled UCI Data Sets

In Figure 1, the classification accuracy of each comparing algorithm is illustrated where the co-occurring probability $\varepsilon$ varies from 0.1 to 0.7 with step-size 0.1 ($p = 1$, $r = 1$). For any ground-truth label $y \in \mathcal{Y}$, one extra label $y' \neq y$ is designated as the coupling label which co-occurs with $y$ in the candidate label set with probability $\varepsilon$. Otherwise, any other class label would be chosen to co-occur with $y$. In Figures 2 to 4, the classification accuracy of each comparing algorithm is illustrated where the proportion $p$ varies from 0.1 to 0.7 with step-size 0.1 ($r = 1, 2, 3$). Together with the ground-truth label, $r$ class labels in $\mathcal{Y}$ will be randomly picked up to constitute the candidate label set for each partially labeled example.

As shown in Figures 1 to 4, in most cases, PL-LEAF achieves superior or competitive performance against the comparing algorithms. Based on pairwise $t$-test at 0.05 significance level, Table 3 summarizes the win/tie/loss counts between PL-LEAF and each comparing algorithm. Out of the 168 statistical tests (28 configurations $\times$ 6 UCI data sets), it is shown that:

- Comparing to averaging-based disambiguation approaches, PL-LEAF achieves superior performance against PL-KNN and CLPL in 66.0% and 90.4% cases respectively. Furthermore, the performance of PL-LEAF is inferior to PL-KNN in only 17.9% cases and has never been outperformed by CLPL.

- Comparing to identification-based disambiguation approaches, PL-LEAF achieves superior performance against PL-SVM and LSB-CMM in 75.5% and 54.7% cases. Furthermore, the performance of PL-LEAF is inferior to LSB-CMM in only 11.9% cases and has been outperformed by PL-SVM in only 2 out of 168 cases.

---

$^5$For PL-LEAF, Gaussian kernel $k(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{\sigma^2}\right)$ is employed in Step 6 with $\sigma$ being the average distance among each pair of training examples. Furthermore, on the two artificial (usps, pendigits) and two real-world (Soccer Player, Yahoo! News) data sets with large scale, alternating optimization is employed in Step 5. Sensitivity analysis on PL-LEAF’s parameter configuration is conducted in Subsection 4.3.
Figure 4: Classification accuracy of each comparing algorithm changes as $p$ (proportion of partially labeled examples) increases (with three false positive candidate labels $[r = 3]$).

Table 4: Classification accuracy (mean±std) of each comparing algorithm on the real-world partial label data sets. In addition, •/◦ indicates whether the performance of PL-LEAF is statistically superior/inferior to the comparing algorithm on each data set (pairwise $t$-test at 0.05 significance level).

<table>
<thead>
<tr>
<th></th>
<th>PL-LEAF</th>
<th>PL-KNN</th>
<th>CLPL</th>
<th>PL-SVM</th>
<th>LSB-CMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>FG-NET</td>
<td>0.072±0.010</td>
<td>0.037±0.008●</td>
<td>0.047±0.017●</td>
<td>0.058±0.010●</td>
<td>0.056±0.008●</td>
</tr>
<tr>
<td>FG-NET (MAE3)</td>
<td>0.411±0.012</td>
<td>0.284±0.035●</td>
<td>0.240±0.045●</td>
<td>0.343±0.022●</td>
<td>0.344±0.026●</td>
</tr>
<tr>
<td>FG-NET (MAE5)</td>
<td>0.550±0.018</td>
<td>0.438±0.033●</td>
<td>0.343±0.055●</td>
<td>0.473±0.016●</td>
<td>0.478±0.025●</td>
</tr>
<tr>
<td>Lost</td>
<td>0.664±0.020</td>
<td>0.332±0.030●</td>
<td>0.670±0.024</td>
<td>0.639±0.056</td>
<td>0.591±0.019●</td>
</tr>
<tr>
<td>MSRCv2</td>
<td>0.459±0.013</td>
<td>0.417±0.012●</td>
<td>0.375±0.020●</td>
<td>0.417±0.027●</td>
<td>0.431±0.008●</td>
</tr>
<tr>
<td>BirdSong</td>
<td>0.706±0.012</td>
<td>0.637±0.009●</td>
<td>0.624±0.009●</td>
<td>0.671±0.018●</td>
<td>0.692±0.015●</td>
</tr>
<tr>
<td>Soccer Player</td>
<td>0.515±0.004</td>
<td>0.494±0.004●</td>
<td>0.347±0.004●</td>
<td>0.430±0.004●</td>
<td>0.506±0.006●</td>
</tr>
<tr>
<td>Yahoo! News</td>
<td>0.597±0.004</td>
<td>0.403±0.004●</td>
<td>0.457±0.005●</td>
<td>0.615±0.002●</td>
<td>0.594±0.007</td>
</tr>
</tbody>
</table>

4.2.2 Real-World Data Sets

Table 4 reports the detailed predictive performance of each comparing algorithm on the real-world PL data sets, where the outcomes of pairwise $t$-tests at 0.05 significance level are also recorded. Note that the average number of candidate labels (Avg. #CLs) for the FG-NET data set (i.e. 7.48 as shown in Table 2) is quite large, which makes the task of facial age estimation (based on training examples with partial labels) rather challenging. Furthermore, the state-of-the-art performance on this data set (based on training examples with ground-truth labels) corresponds to more than 3 years of mean average error (MAE) between the predicted age and the ground-truth age [20]. In Table 4, two extra classification accuracies are reported on the FG-NET data set where an unseen example is regarded to be correctly classified if the difference between the predicted age and the ground-truth age is less than 3 years (MAE3) or 5 years (MAE5).

As shown in Table 4, it is impressive to observe that:

- On the FG-NET (with its MAE3 and MAE5 variants), MSRCv2, BirdSong and Soccer Player data sets, the performance of PL-LEAF is superior to all the other comparing algorithms.
- On the Lost data set, the performance of PL-LEAF is superior to PL-KNN and LSB-CMM, and comparable to CLPL and PL-SVM.
- On the Yahoo! News data set, the performance of PL-LEAF is superior to PL-KNN and CLPL, comparable to LSB-CMM, and inferior to PL-SVM.

4.3 Further Analysis

In addition to Table 4 reporting inductive performance on test examples, it is also interesting to study the transductive performance of each comparing algorithm on classifying training examples [8, 24]. For each PL training example $(x_i, S_i)$, its ground-truth label is predicted by consulting the candidate label set $S_i$, i.e. predicting $\hat{y}_i \in S_i$ with largest modeling output. In other words, transductive
Table 5: Transductive accuracy (mean±std) of each comparing algorithm on the real-world partial label data sets. In addition, ∗/○ indicates whether the performance of PL-LEAF is statistically superior/inferior to the comparing algorithm on each data set (pairwise t-test at 0.05 significance level).

<table>
<thead>
<tr>
<th></th>
<th>PL-LEAF</th>
<th>PL-KNN</th>
<th>CLPL</th>
<th>PL-SVM</th>
<th>LSB-CMM</th>
<th>PL-LEAF†</th>
</tr>
</thead>
<tbody>
<tr>
<td>FG-NET</td>
<td>0.148±0.009</td>
<td>0.173±0.017○</td>
<td>0.158±0.018</td>
<td>0.136±0.021</td>
<td>0.138±0.019</td>
<td>0.142±0.010</td>
</tr>
<tr>
<td>FG-NET (MAE3)</td>
<td>0.567±0.015</td>
<td>0.559±0.023</td>
<td>0.548±0.017○</td>
<td>0.502±0.030○</td>
<td>0.541±0.022○</td>
<td>0.556±0.014</td>
</tr>
<tr>
<td>FG-NET (MAE5)</td>
<td>0.710±0.014</td>
<td>0.721±0.021</td>
<td>0.684±0.021○</td>
<td>0.644±0.021○</td>
<td>0.684±0.010○</td>
<td>0.698±0.010 ○</td>
</tr>
<tr>
<td>Lost</td>
<td>0.809±0.022</td>
<td>0.596±0.018○</td>
<td>0.855±0.007○</td>
<td>0.814±0.033○</td>
<td>0.755±0.010○</td>
<td>0.788±0.025 ○</td>
</tr>
<tr>
<td>MSRCv2</td>
<td>0.645±0.015</td>
<td>0.603±0.015</td>
<td>0.612±0.035○</td>
<td>0.656±0.026</td>
<td>0.603±0.016○</td>
<td>0.629±0.016 ○</td>
</tr>
<tr>
<td>BirdSong</td>
<td>0.822±0.014</td>
<td>0.766±0.015</td>
<td>0.820±0.014</td>
<td>0.831±0.013</td>
<td>0.827±0.017</td>
<td>0.787±0.016 ○</td>
</tr>
<tr>
<td>Soccer Player</td>
<td>0.702±0.003</td>
<td>0.674±0.002○</td>
<td>0.742±0.005○</td>
<td>0.733±0.007○</td>
<td>0.688±0.003○</td>
<td>0.669±0.003 ○</td>
</tr>
<tr>
<td>Yahoo! News</td>
<td>0.827±0.002</td>
<td>0.720±0.003○</td>
<td>0.832±0.002○</td>
<td>0.861±0.003○</td>
<td>0.861±0.002○</td>
<td>0.822±0.002 ○</td>
</tr>
</tbody>
</table>

Figure 5: Parameter sensitivity analysis for PL-LEAF on the Lost, MSRCv2 and BirdSong data sets. (a) Classification accuracy of PL-LEAF changes as K increases from 6 to 14 with step-size 2; (b) Classification accuracy of PL-LEAF changes as C1 increases from 6 to 14 with step-size 2; (c) Classification accuracy of PL-LEAF changes as C2 increases from 0.6 to 1.4 with step-size 0.2.

As the feature-aware disambiguation stage of PL-LEAF finishes, the generated labeling confidence vector λi can also be used to predict the ground-truth label of x, as ̂yi = arg maxλi∈Λ |λi|λi < 0.5. The resulting transductive performance is reported in Table 5 (denoted as PL-LEAF†) for reference purpose. In most cases, the transductive performance of PL-LEAF† is close to PL-LEAF indicating the usefulness of generated labeling confidence vectors.

As shown in Table 1, to study the sensitivity of PL-LEAF w.r.t. its parameters K, C1 and C2, Figure 5 illustrates how PL-LEAF performs under different parameter configurations. For clarity of illustration, three data sets (Lost, MSRCv2 and BirdSong) are chosen here for sensitivity analysis while similar observations also hold on other data sets.

It is obvious from Figure 5 that the performance of PL-LEAF is stable across a broad range of each parameter. This property is quite desirable as one can make use of PL-LEAF to achieve robust classification performance without the need of parameter fine-tuning. Therefore, the parameter configuration for PL-LEAF in Subsection 4.1 (K=10, C1=10, C2=1) naturally follows from these observations.

5. CONCLUSION

In this paper, a novel approach named PL-LEAF is proposed to learning from partial label examples. Different from existing strategies, PL-LEAF aims to disambiguate the candidate label set by manipulating useful information in the feature space. Specifically, PL-LEAF generates normalized labeling confidence vectors based on manifold relationships among training examples, and then induces the predictive model based on multi-output regression analysis. Comparative studies across comprehensive partial label data sets clearly verify the effectiveness of the proposed approach.

For PL-LEAF, an important future work is to investigate ways to perform manifold structure discovery and labeling confidence generation simultaneously. Secondly, it is worth studying whether there are better techniques to exploit the generated labeling confidence vectors, such as fitting probabilistic models [10]. Thirdly, it is also interesting to explore other ways to fulfill the feature-aware disambiguation strategy.
References


