Can Label-Specific Features Help Partial-Label Learning?  
Ruo-Jing Dong, Jun-Yi Hang, Tong Wei*, Min-Ling Zhang  
School of Computer Science and Engineering, Southeast University, Nanjing 210096, China  
Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China  
Computer Experimental Teaching Center of Southeast University  
{dongrj, hangjy, weit, zhangml}@seu.edu.cn

Abstract  
Partial label learning (PLL) aims to learn from inexact data annotations where each training example is associated with a coarse candidate label set. Due to its practicability, many PLL algorithms have been proposed in recent literature. Among these methods, PLL aims to learn a classifier from training datasets with inexact ground-truth labels. From a different perspective, we propose to enrich the feature space and raise the question “Can label-specific features help PLL?” rather than learning from examples with identical features for all classes. Despite its benefits, previous label-specific features approaches rely on ground-truth labels to split positive and negative examples of each class and then conduct clustering analysis, which is not directly applicable in PLL. To remedy this problem, we propose an uncertainty-aware confidence region to accommodate false positive labels. We first employ graph-based label enhancement to yield smooth pseudo-labels and facilitate the confidence region split. After acquiring label-specific features, a family of binary classifiers is induced. Extensive experiments on both synthesized and real-world datasets are conducted and the results show that our method consistently outperforms eight baselines. Our code is released at https://github.com/meteoseeker/UCL.

Introduction  
Partial label learning (PLL) is a significant problem which has been studied a lot in the past decade (Cour, Sapp, and Taskar 2011; Chen, Patel, and Chellappa 2017; Feng et al. 2020; Lv et al. 2020; Wen et al. 2021; Wang et al. 2022). PLL aims to learn a classifier from training datasets with inexact labels, i.e., each instance is associated with a set of candidate labels among which only one is correct (Cour, Sapp, and Taskar 2011; Lyu et al. 2020; Bao, Hang, and Zhang 2021). Notice that, PLL is also known as superset label learning (Liu and Dietterich 2012; Zhou et al. 2018). PLL algorithms have been proposed in recent literature. Most prior PLL works attempt to identify the ground-truth labels from candidate sets and the classifier is trained afterwards by fitting the features of examples and their exact ground-truth labels. From a different perspective, we propose to enrich the feature space and raise the question “Can label-specific features help PLL?” rather than learning from examples with identical features for all classes. Despite its benefits, previous label-specific features approaches rely on ground-truth labels to split positive and negative examples of each class and then conduct clustering analysis, which is not directly applicable in PLL. To remedy this problem, we propose an uncertainty-aware confidence region to accommodate false positive labels. We first employ graph-based label enhancement to yield smooth pseudo-labels and facilitate the confidence region split. After acquiring label-specific features, a family of binary classifiers is induced. Extensive experiments on both synthesized and real-world datasets are conducted and the results show that our method consistently outperforms eight baselines. Our code is released at https://github.com/meteoseeker/UCL.

*Corresponding author  
Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
cannot be directly adapted to PLL because conventional label-specific features approaches rely on exact data annotations. Specifically, for each class, they first partition the data into positive and negative examples by the ground-truth labels, then conduct the clustering analysis to obtain cluster centers. The label-specific features are then calculated based on the distance between samples and clusters. However, the ground-truth labels are not accessible in PLL, which is the key obstacle to generating label-specific features.

To overcome this challenge, this paper constructs the Uncertainty-aware Label-specific features (UCL) that explores underlying true labels of training samples and incorporates the uncertainty to improve the label-specific features generation procedure. First, we employ a graph-based label enhancement to refine the given candidate labels, which can exploit the structural information of data. Then, for each class, the data can be split into three parts, i.e., positive, uncertain, and negative samples. This is realized based on the output pseudo-labeling confidence by the graph-based label enhancement module. The introduced uncertain region can incorporate false positive labels in the candidates. We conduct spectral clustering to each part of the examples and collect cluster centers as their representatives. After that, the label-specific features are obtained by computing the distances between examples and clusters. Finally, the one-vs-rest classifier is learned using the newly formed training dataset with label-specific features. Extensive experiments on synthesized and real-world datasets demonstrate the superiority of the proposed approach and the effectiveness of label-specific features. To sum up, our contributions are:

1. We for the first time study the label-specific features in PLL which has the potential to inspire more research on exploiting the feature space enhancement;
2. We extend the supervised label-specific features to PLL by proposing an uncertainty-aware confidence thresholding approach to deal with inexact annotations;
3. Our method achieves superior performance against eight existing PLL algorithms on ten datasets (five synthetic and five real-world datasets);
4. We study the case where the effect of label-specific features is not noticeable, which can guide the future design of the learning framework.

**Related Work**

**Partial Label Learning** Most existing PLL algorithms attempt to learn from ambiguous data annotations via candidate label disambiguation, which is conducted in the label space. Corresponding strategies can be roughly divided into two types. One intuitive strategy is the averaging-based method, which simply treats each candidate label of an instance equally for training. For instance, some approaches (Cour, Sapp, and Taskar 2011; Tang and Zhang 2017) attempt to distinguish the averaged assignments for all candidate labels from the assignments of non-candidate labels. Some other approaches (Hüllermeier and Beringer 2006; Zhang and Yu 2015; Gong et al. 2017) classify the unseen instance by voting the candidate labels of its neighbors. Despite the simplicity of the averaging-based strategy, one potential defect lies in that the learning process is usually affected by false positive labels because it does not filter out incorrect candidate labels.

Another strategy of candidate label disambiguation is to identify the ground-truth labels which are treated as latent variables during the model training phase. Generally, iterative optimization procedure such as EM is utilized to predict the latent variables, which can be achieved by likelihood maximum (Jin and Grahamani 2002; Liu and Dietterich 2012) or margin maximum approaches (Nguyen and Caruana 2008; Yu and Zhang 2016; Chai, Tsang, and Chen 2019). However, a potential drawback of this strategy is that the identified label may turn out to be a false positive rather than the ground-truth label, which hurts the performance.

Apart from the above strategies, some recent works propose to manipulate the feature space. These methods attempt to dig out the ignored information in feature space and make full use of it, which is flexible to conduct in practice. For instance, a few approaches (Zhang, Zhou, and Liu 2016; Wang, Zhang, and Li 2021) generate latent labeling confidence over candidate label set by utilizing the graph structure among training data. Some approaches (Wu and Zhang 2019; Bao, Hang, and Zhang 2021) conduct dimensionality reductions on training data by learning a projection matrix.

**Label-Specific Features** The key idea of label-specific features is to generate different instance representations for each class. The methods of label-specific features generation can be roughly divided into two categories, i.e., feature selection and prototype-based feature transformation.

Feature selection methods generate label-specific features by retaining a specific subset from the original feature set for each class. Representative works such as LLSF use Lasso regression (Huang et al. 2015) for label-specific features selection while considering label correlations. A subsequent work employs regularized optimization (Huang et al. 2017; Zhang et al. 2018a) for feature selection, which first maps the features to a high-dimensional space and then selects the most prominent features (Kashef and Nezamabadi-pour 2019).

Label-specific features can also be generated by transforming prototypes of each class label. For example, the seminal work Lift (Zhang and Wu 2014) provides a three-stage framework to perform prototype-based label-specific features transformation. To be specific, for each class, positive/negative prototypes are collected by performing the clustering analysis on its positive/negative training examples. Subsequently, label-specific features are generated via querying distances between examples and the gathered prototypes, which are utilized to induce a classification model. Extensive efforts have been made in this line of research to improve the multi-stage framework such as implementing feature reduction (Xu et al. 2016) or incorporating local pairwise label correlation (Weng et al. 2018). In addition, an end-to-end prototype-based approach is recently proposed (Hang et al. 2022) to reduce the defects brought by the decoupling nature of the previous multi-stage approaches.

Although label-specific features improve the performance in multi-label learning problems, it has not been studied in PLL yet. To fill this gap, this paper raises an intrinsic question “Can label-specific features help PLL?” and presents a
new effective method to improve the performance.

**The Proposed Approach**

**Problem Setup** The goal of PLL is to learn a classifier \( f : X \to \mathbb{Y} \) using the training set \( D = (X, Y) \). We denote \( X = [x_1, \cdots, x_N] \in \mathbb{R}^{N \times d} \) as the feature matrix and \( Y = [y_1, \cdots, y_N] \in \{0, 1\}^{N \times L} \) as the corresponding label matrix, in which \( y_{ij} = 1 \) signifies that the \( j \)-th label is a candidate label of the \( i \)-th instance \( x_i \), whilst \( y_{ij} = 0 \) signifies the \( j \)-th label is a non-candidate label. We denote \( N \) and \( L \) as the number of training examples and classes, respectively.

**Main Idea** This paper explores the merit of label-specific features for PLL. The main obstacle is that the generation of label-specific features requires ground-truth labels in previous works (Zhang and Wu 2014; Wei, Tu, and Li 2019), which is not accessible in PLL. Therefore, we first introduce the graph-based label enhancement procedure to refine candidate labels; then an uncertainty-aware label-specific features generation approach is presented to deal with the uncertainty of being the ground-truth labels.

**Graph-based Label Enhancement**

We propose a label enhancement approach to propagate pseudo-labels on the undirected graph \( G = (V, E) \) by leveraging similarities between data. \( V \) and \( E \) denote the set of graph vertices and edges, respectively. In graph \( G \), the similarities between vertices are encoded by a weight matrix \( W = [w_1, \cdots, w_N] \). Considering the efficiency and scalability, we adopt the \( k \)-NN matrix to represent the similarity of each pair of data points \((x_i, x_j)\), \(1 \leq i,j \leq N\) as follows:

\[
w_{ij} := \begin{cases} 
ex((-\|x_i - x_j\|^2) / 2\sigma^2), & \text{if } i \neq j \land x_i \in \text{NN}_k(x_j) \\
0, & \text{otherwise} \end{cases}
\]

Here, we adopt the Gaussian kernel to measure the similarities and \( \sigma^2 \) is the bandwidth of the kernel, which is a popular choice in previous related works. Moreover, we employ the KD-Tree algorithm (Bentley 1975) to effectively compute the \( k \) nearest neighbors denoted by \( \text{NN}_k \), which reduces the computational burden. To capture high-order graph information, prior works design models on the assumption that labels vary smoothly over the edges of the graph. To do so, we first normalize the weight matrix by \( w_{ij} = w_{ij} / \sum_{k=1}^N w_{ik} \) and then propagate candidate labels for label disambiguation (Zhang and Yu 2015). Let \( \hat{Y} \) denote the soft pseudo-label matrix, which is iteratively updated during the label propagation process. Specifically, at the \( t \)-iteration, we have:

\[
\hat{Y}^t = \beta \hat{W}^T \hat{Y}^{t-1} + (1 - \beta) P
\]

Here, \( P = [p_1, \cdots, p_N] \) is the normalized partial label matrix, i.e., \( p_{ij} = y_{ij} / \sum_{k=1}^L y_{ik}, \forall 1 \leq i \leq N \), parameter \( \beta \in (0, 1) \) controls the balance between propagated pseudo-labels and initial labels. Noted that we set \( \hat{Y}^0 = X \) as initialization. In Eq. (2), all nodes propagate candidate labels to their neighbors according to edge weights. After each iteration, we normalize \( \hat{Y}^t \) by:

\[
\hat{y}_{ij}^t = \hat{y}_{ij}^t \cdot y_{ij} / \sum_{k=1}^L \hat{y}_{ik}^t \cdot y_{ik}, \forall 1 \leq i \leq N
\]

The label propagation process continues until convergence or reaches pre-defined maximum iterations. Based on the enhanced label matrix, we introduce the uncertainty-aware label-specific features approach in the next section.

**Uncertainty-aware Label-specific Features**

Unlike previous works which divide training instances into positive and negative categories, our approach partitions the training examples into positive, negative, and uncertain parts, which are respectively denoted by \( P, N, \) and \( U \). Specifically, for the \( j \)-th class, we have:

\[
P_j = \{ x_i \mid (x_i, y_i) \in D, y_{ij} > \tau_h \},
N_j = \{ x_i \mid (x_i, y_i) \in D, y_{ij} < \tau_l \},
U_j = \{ x_i \mid (x_i, y_i) \in D, \tau_l \leq y_{ij} \leq \tau_h \}
\]

Here, \( \tau_h \) and \( \tau_l \) are confidence thresholding parameters that are set as constants in our experiments.

The rationale behind Eq. (4) is that we leverage the confidence to represent the uncertainty of an example. For each class, examples associated with high confidence are classified as positive ones, which indicates the current label achieves an overwhelming victory against the other candidates, whilst examples associated with low confidence are categorized as negative ones, which usually are not related to the current label or in some cases have a large number of equivalent candidate labels. However, there exists a portion of examples difficult to classify, which are usually labeled with few highly competitive candidate labels. Examples uncertain about their observed labels are usually confusing and might have a misleading impact on feature generation if they participated in the following clustering tasks. Thus, we single out and deal with uncertain examples separately.

Moreover, PLL usually suffers from the class imbalance problem (Zhang and Zhou 2013), where the number of positive examples for each class is much less than the number of negative ones, i.e., \(|P_j| \ll |N_j|\), Following the thought of LIFT, clustering information gained from positive examples as well as negative examples is treated with equal importance. By setting a ratio parameter \( r \), we obtain the number of clusters of positive \( m_j \), negative \( c_j \), and uncertain \( c_j \) parts for the \( j \)-th class as follows:

\[
m_j = \lceil r \cdot \min(|P_j|, |N_j|) \rceil,
c_j = \lceil r \cdot |U_j| \rceil
\]

Notice that in a representative work LIFT, it chooses the k-means clustering to explore the local structures underlying positive and negative instances. However, in our approach we instead choose spectral clustering (Shi and Malik 2000; Von Luxburg 2007) inspired by spectral instance alignment (Zhang, Fang, and Li 2015), which considers the correlation between positive and negative instances. The goal of spectral clustering is to divide data (represented as a similarity graph) into several groups where the similarity between data points is high in the same group whilst low in different groups. After collecting the cluster centers for each part, we construct the label-specific features transformation \( \phi_j : X \to Z_j \) for
Algorithm 1: The label-specific features approach in UCL

Inputs:
D: the PL training set \( D = \{ (X, Y) \} \)
\( r \): the ratio parameter controlling the number of clusters
\( k \): the number of nearest neighbors
\( \tau_h, \tau_l \): the high/low partition thresholds
\( \alpha \): the discount parameter

Output: feature transformation \( \phi \)

1: calculate similarity matrix \( W \) using Eq. (1)
2: refine the pseudo-label matrix \( Y \) using Eq. (2)
3: for \( j = 1 \) to \( L \) do
4: partition the training examples into positive, negative, and uncertain parts using Eq. (4)
5: perform spectral clustering on three parts respectively
6: set \( \phi_j \) with the cluster centers using Eq. (6)
7: end for
8: return the label-specific features transformation \( \phi_j \)

the \( j \)-th class as follows:
\[
\phi_j(x) = \left[ d \left( x, p^+_j \right), \cdots, d \left( x, p^+_m \right),
\right.
\[
\alpha \times d \left( x, n^{-}_j \right), \cdots,\alpha \times d \left( x, n^{-}_m \right)
\]

Here, \( p^+_j, n^-_j, u^-_j \) are clustering centers for positive, negative, and uncertain parts, respectively. \( d(\cdot, \cdot) \) is the distance measure and is set to Euclidean distance in our experiments. Function \( \phi_j \) transforms the original \( d \)-dimensional input feature space to a \( (2n_j+c_j) \)-dimensional label-specific feature space. \( \alpha \) is a discount parameter to put a larger weight on positive and negative clusters. Algorithm 1 summarizes the pseudo-code of label-specific features generation procedure.

UCL can cooperate with many existing PLL classifiers such as PL-kNN (Hüllermeier and Beringer 2006), PL-SVM (Nguyen and Caruana 2008), and SURE (Feng and An 2019). In the experiments of this paper, we choose SURE as the default classifier for its superior performance.

Experiments

We conduct experiments on ten datasets (five synthetic and five real-world datasets) to compare the performance of our method with eight baselines. On each dataset, ten-fold cross-validation is performed and the mean accuracy, as well as standard deviation, are reported. In addition, we use a pairwise t-test at 0.05 significance level for two independent samples to investigate whether our method UCL is significantly superior/inferior (win/loss) to comparing methods.

Comparing Methods

To show the effectiveness of UCL, we compared it with eight existing PLL algorithms, each configured with suggested parameters in the respective literature.

- PL-kNN (Hüllermeier and Beringer 2006): a \( k \)-nearest neighbor approach that makes predictions by averaging the labeling information of neighboring examples [suggested configuration: \( k \in \{5, 6, \cdots, 10\} \)].
- PL-SVM (Nguyen and Caruana 2008): a maximum margin approach that learns from PL examples by optimizing margin-based objective function [suggested configuration: \( \lambda \in \{10^{-3}, 10^{-2}, \cdots, 10^3\} \)].
- SURE (Feng and An 2019): a self-training based approach which trains the desired model and performs pseudo-labeling jointly by solving a convex-concave optimization problem [suggested configuration: regularization parameters \( \lambda = 0.3, \beta = 0.05 \)].
- CENDA (Bao, Hang, and Zhang 2021): a dimensionality reduction approach via confidence-based dependence maximization which combines other algorithms such as SURE to work [suggested configuration: \( thr = 0.999, \mu = 0.5, k = 8 \)].
- DELIN (Wu and Zhang 2019): a dimensionality reduction approach which adapts linear discriminant analysis and combines other algorithms such as SURE to work [suggested configuration: \( r = 0.6, k = 8, T = 75 \)].
- IPAL (Zhang and Yu 2015): an instance-based approach that disambiguates candidate labels by an adapted label propagation scheme [suggested configuration: \( \alpha \in \{0, 0.1, \cdots, 1\}, k \in \{5, 6, \cdots, 10\} \)].
- AGGD (Wang, Zhang, and Li 2021): a disambiguation approach which performs adaptive graph construction, candidate label disambiguation and predictive model induction with alternating optimization [suggested configuration: \( k = 10, T = 20, \lambda = 1, \mu = 1, \gamma = 0.05 \)].
- PL-LE (Xu, Lv, and Geng 2019): an approach which considers generalized label distribution and learns from PL examples via label enhancement [suggested configuration: \( k = 20, \lambda = 0.01, c1 = 1, c2 = 1 \)].

Results on Controlled UCI Datasets

Table 1 summarizes the characteristics of five UCI datasets (Dua and Graff 2017), ranging from small datasets to quite large datasets. Following the widely-used controlled protocol, an artificial partial label dataset is derived from one multi-class UCI dataset by configuring three controlling parameters \( p, r \) and \( \epsilon \) (Cour, Sapp, and Taskar 2011; Liu and Dietnerich 2012; Zhang and Yu 2015; Feng and An 2019; Wang, Zhang, and Li 2021). Here, \( p \) controls the proportion of examples that are partially labeled, \( r \) controls the number of false positive labels in the candidate label set, and \( \epsilon \) is the probability of picking co-occurring label as a false positive label. As shown in Table 1, Configuration (I) exams varying \( r \) and Configuration (II) exams varying \( \epsilon \).

Table 2 illustrates the classification accuracy of each algorithm when \( r = 1, 2, \) and \( 3 \) (Configuration (I)), respectively. In these three settings, \( r \) extra labels are randomly chosen to be the false positive labels. That is, the number of candidate labels for each instance is \( r+1 \). Figure 1 shows the classification accuracy of each algorithm as \( \epsilon \) ranges from 0.1 to 0.7 when \( p = 1 \) and \( r = 1 \) (Configuration (II)). In this setting, a specific label is selected as the coupling candidate label that co-occurs with
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Examples</th>
<th>Features</th>
<th>Classes</th>
<th>Task Domain</th>
<th>Configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoo</td>
<td>101</td>
<td>16</td>
<td>7</td>
<td>animal classification</td>
<td>(I) $p = 1, r \in {1, 2, 3}$</td>
</tr>
<tr>
<td>glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
<td>glass classification</td>
<td>(II) $p = 1, r = 1, \epsilon \in {0.1, 0.2, \ldots, 0.7}$</td>
</tr>
<tr>
<td>yeast</td>
<td>1,484</td>
<td>8</td>
<td>10</td>
<td>yeast classification</td>
<td></td>
</tr>
<tr>
<td>tmc2007</td>
<td>8,670</td>
<td>981</td>
<td>18</td>
<td>text anomaly detection</td>
<td></td>
</tr>
<tr>
<td>sport</td>
<td>9,120</td>
<td>1738</td>
<td>19</td>
<td>human activity recognition</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Characteristics of the controlled UCI datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>zoo</th>
<th>glass</th>
<th>yeast</th>
<th>tmc2007</th>
<th>sport</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.9009 ± 0.1054</td>
<td>0.7106 ± 0.0742</td>
<td>0.5983 ± 0.0596</td>
<td>0.6656 ± 0.0197</td>
<td>0.8978 ± 0.0116</td>
</tr>
<tr>
<td>UCL (ours)</td>
<td>0.8800 ± 0.1229</td>
<td>0.6970 ± 0.1079</td>
<td>0.5950 ± 0.0528</td>
<td>0.6505 ± 0.0180</td>
<td>0.8860 ± 0.0044</td>
</tr>
<tr>
<td>PL-KNN</td>
<td>0.7936 ± 0.1318</td>
<td>0.6409 ± 0.0941</td>
<td>0.5755 ± 0.0494</td>
<td>0.4285 ± 0.0204</td>
<td>0.7936 ± 0.1318</td>
</tr>
<tr>
<td>PL-SVM</td>
<td>0.7736 ± 0.1374</td>
<td>0.4820 ± 0.0942</td>
<td>0.3450 ± 0.0544</td>
<td>0.6449 ± 0.0152</td>
<td>0.6820 ± 0.0151</td>
</tr>
<tr>
<td>SURE</td>
<td>0.9100 ± 0.1287</td>
<td>0.6784 ± 0.0873</td>
<td>0.5929 ± 0.0531</td>
<td>0.6451 ± 0.0170</td>
<td>0.7601 ± 0.0157</td>
</tr>
<tr>
<td>SURE-CENDA</td>
<td>0.8718 ± 0.1047</td>
<td>0.6314 ± 0.0971</td>
<td>0.5977 ± 0.0486</td>
<td>0.6374 ± 0.0176</td>
<td>0.7777 ± 0.0143</td>
</tr>
<tr>
<td>SURE-DELIN</td>
<td>0.8909 ± 0.1102</td>
<td>0.6175 ± 0.1255</td>
<td>0.5835 ± 0.0402</td>
<td>0.6074 ± 0.0164</td>
<td>0.7456 ± 0.0092</td>
</tr>
<tr>
<td>IPAL</td>
<td>0.8418 ± 0.1168</td>
<td>0.6788 ± 0.1078</td>
<td>0.5027 ± 0.0603</td>
<td>0.5945 ± 0.0210</td>
<td>0.9041 ± 0.0125</td>
</tr>
<tr>
<td>AGGD</td>
<td>0.8818 ± 0.1217</td>
<td>0.6833 ± 0.1067</td>
<td>0.5963 ± 0.0539</td>
<td>0.6459 ± 0.0162</td>
<td>0.7776 ± 0.0135</td>
</tr>
<tr>
<td>PL-LE</td>
<td>0.8909 ± 0.1198</td>
<td>0.6883 ± 0.1367</td>
<td>0.5970 ± 0.0496</td>
<td>0.6449 ± 0.0182</td>
<td>0.7498 ± 0.0164</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>zoo</th>
<th>glass</th>
<th>yeast</th>
<th>tmc2007</th>
<th>sport</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.8318 ± 0.1248</td>
<td>0.6649 ± 0.1227</td>
<td>0.5930 ± 0.0442</td>
<td>0.6541 ± 0.0202</td>
<td>0.8711 ± 0.0102</td>
</tr>
<tr>
<td>UCL (ours)</td>
<td>0.7545 ± 0.1350</td>
<td>0.5574 ± 0.1074</td>
<td>0.5088 ± 0.0504</td>
<td>0.3789 ± 0.0216</td>
<td>0.3013 ± 0.0183</td>
</tr>
<tr>
<td>PL-KNN</td>
<td>0.4845 ± 0.1699</td>
<td>0.1959 ± 0.0913</td>
<td>0.3120 ± 0.0385</td>
<td>0.6326 ± 0.0166</td>
<td>0.6103 ± 0.0235</td>
</tr>
<tr>
<td>PL-SVM</td>
<td>0.8127 ± 0.0964</td>
<td>0.5749 ± 0.0917</td>
<td>0.5916 ± 0.0390</td>
<td>0.6255 ± 0.0197</td>
<td>0.6795 ± 0.0101</td>
</tr>
<tr>
<td>SURE</td>
<td>0.7736 ± 0.1105</td>
<td>0.5615 ± 0.0940</td>
<td>0.5808 ± 0.0441</td>
<td>0.6301 ± 0.0214</td>
<td>0.7200 ± 0.0179</td>
</tr>
<tr>
<td>SURE-CENDA</td>
<td>0.7436 ± 0.0913</td>
<td>0.5524 ± 0.1055</td>
<td>0.5680 ± 0.0450</td>
<td>0.6069 ± 0.0176</td>
<td>0.7088 ± 0.0129</td>
</tr>
<tr>
<td>SURE-DELIN</td>
<td>0.3921 ± 0.1447</td>
<td>0.5472 ± 0.1239</td>
<td>0.4488 ± 0.0455</td>
<td>0.5664 ± 0.0236</td>
<td>0.8922 ± 0.0137</td>
</tr>
<tr>
<td>IPAL</td>
<td>0.7927 ± 0.0970</td>
<td>0.5898 ± 0.0990</td>
<td>0.5963 ± 0.0432</td>
<td>0.6290 ± 0.0181</td>
<td>0.7523 ± 0.0154</td>
</tr>
<tr>
<td>AGGD</td>
<td>0.7727 ± 0.1144</td>
<td>0.5753 ± 0.0971</td>
<td>0.5889 ± 0.0430</td>
<td>0.6232 ± 0.0192</td>
<td>0.7206 ± 0.0141</td>
</tr>
<tr>
<td>PL-LE</td>
<td>0.8518 ± 0.1428</td>
<td>0.6509 ± 0.1012</td>
<td>0.5876 ± 0.0459</td>
<td>0.6301 ± 0.0141</td>
<td>0.7364 ± 0.0154</td>
</tr>
</tbody>
</table>

Table 2: Classification accuracy of each algorithm on the UCI datasets. Furthermore, •/○ indicate UCL is statistically superior/inferior to the comparing algorithm (t-test at 0.05 significance level for two independent samples).

the ground-truth label with probability $\epsilon$, and any other label would be randomly chosen to be a false positive label with probability $1 - \epsilon$. As shown in Table 2 and Figure 1, UCL consistently outperforms other comparing algorithms. To further statistically compare UCL with other algorithms, the detailed win/tie/loss counts between UCL and the comparing algorithms are recorded in Table 3. From the above results, we have the following observations:

- According to Table 2, there is an indication that UCL achieves better performance against other algorithms when adding more false positive labels.
- UCL achieves superior or at least comparable performance against five algorithms including PL-KNN, PL-SVM, SURE-CENDA, SURE-DELIN and AGGD in all cases. Specifically, UCL outperforms them in 82%, 96%, 56%, 66% and 38% cases successively.
- UCL achieves superior performance against SURE, IPAL, and PL-LE in 42%, 60% and 42% cases, respectively.
Results on Real-World Datasets

In addition to five synthetic datasets, we further test our method on five real-world partial-labeled datasets which are collected from various task domains, i.e., Lost (Cour, Sapp, and Taskar 2011), MSRCv2 (Liu and Dietterich 2012), Mik- flickr (Huiskes and Lew 2008), BirdSong (Briggs, Fern, and Raich 2012), and Soccer Player (Zeng et al. 2013). Table 4 shows the characteristics of real-world datasets.

Table 5 reports the mean classification accuracy and the standard deviation of each method on five real-world datasets. From the results, we can observe that:

- UCL outperforms PL-KNN, PL-SVM, SURE-DELIN and PL-LE significantly on all of the real-world datasets.
- Out of the 40 cases (8 comparing algorithms and 5 datasets), UCL is statistically superior to all the comparing algorithms in 85% cases and outperforms all the comparing algorithms in 95% cases.
- UCL is never significantly outperformed by any comparing algorithms in all experiments.

Further Understanding of Our Method

Parameter Sensitivity Analysis

In the proposed UCL approach, three trade-off parameters $\tau_h$, $\tau_l$, and $\alpha$ need to be manually searched. Figure 2 shows how three parameters affect classification accuracy in four datasets (Lost, MSRCv2, glass, tmc2007), two of which are real-world datasets and another two are synthetic datasets. Specifically, we vary $\tau_l$ ranging from 0.1 to 0.4 when $\tau_h = 0.6$, $\alpha = 0.5$; vary $\tau_h$ ranging from 0.5 to 0.9 when $\tau_l = 0.2$, $\alpha = 0.5$; vary $\alpha$ ranging from 0.1 to 1.0 when $\tau_l = 0.2$, $\tau_h = 0.6$. As shown in Figure 2, the performance of UCL is generally stable across a wide range of each parameter, which means UCL achieves robust classification performance without costing much effort on parameter fine-tuning.

Ablation Studies on the Partitioning Strategy

Notice that our method UCL splits the data for every class into three partitions, i.e., positive, negative, and uncertain. To verify the effectiveness and necessity of our partitioning strategy, we compare the classification accuracy of UCL with its two variants, i.e., UCL without a partition (denoted by “no partition”) and UCL with only positive and negative partitions (denoted by “PN partition”). Specifically, in UCL without partition, we conduct spectral clustering on all training instances where cluster centers are utilized to produce label-specific features. Moreover, in PN PARTITION, we follow the original strategy of LIFT which partitions training instances into positive and negative ones for subsequent label-specific features generation. As mentioned above, LIFT solves multi-label learning problems where training examples can be classified into two parts according to their ground-truth labels while in PLL the partition is more complicated due to the inexact information. Thus, our
In Table 6, we test the CL combination of U with another two popular PLL classifiers, i.e., UCL-KNN and UCL-SVM, on five real-world datasets. We can observe that in the t-test at 0.05 significance level UCL-KNN achieves 2/2/1 (win/tie/loss) compared to PL-KNN, and UCL-SVM achieves 3/2/0 (win/tie/loss) compared to PL-SVM. This indicates that there is still much room for improving the versatility of our approach. Notice that our default choice of classifier, i.e., SURE, is more powerful than k-NN and SVM, we recommend combining label-specific features with strong classifiers.

**Conclusion**

This paper for the first time studies the utility of label-specific features for PLL. It answers the fundamental question "Can label-specific features help PLL?" in the affirmative. Our proposed approach UCL consistently improves the performance on both synthetic datasets and real-world datasets. However, when combined with weak classifiers, the positive effect of label-specific features is unnoticeable, and the performance may even deteriorate. We believe this paper can motivate more research by generating more informative label-specific features and combining them with more powerful classifiers such as deep neural networks.
Acknowledgments
This research was supported by the National Science Foundation of China (62206049, 62176055). We thank the Big Data Center of Southeast University for providing the facility support on the numerical calculations.

References


