Progressive Identification of True Labels for Partial-Label Learning

Jiaqi Lv¹ ¹  Miao Xu² ³  Lei Feng⁴  Gang Niu²  Xin Geng¹  Masashi Sugiyama² ⁵

Abstract

Partial-label learning (PLL) is a typical weakly supervised learning problem, where each training instance is equipped with a set of candidate labels among which only one is the true label. Most existing methods elaborately designed learning objectives as constrained optimizations that must be solved in specific manners, making their computational complexity a bottleneck for scaling up to big data. The goal of this paper is to propose a novel framework of PLL with flexibility on the model and optimization algorithm. More specifically, we propose a novel estimator of the classification risk, theoretically analyze the classifier-consistency, and establish an estimation error bound. Then we propose a progressive identification algorithm for approximately minimizing the proposed risk estimator, where the update of the model and identification of true labels are conducted in a seamless manner. The resulting algorithm is model-independent and loss-independent, and compatible with stochastic optimization. Thorough experiments demonstrate it sets the new state of the art.

1. Introduction

The increasing demand for massive data used in training modern machine learning models, such as deep neural networks (DNNs), makes it inevitable that not all examples are equipped with strong supervision information. Weak supervision may result in a detrimental effect on model training and has been widely studied (Zhou, 2017; Patrini et al., 2017; Kamnitsas et al., 2018; Lu et al., 2019; Gong et al., 2019). We focus on an important weakly supervised learning problem called partial-label learning (PLL) (Cour et al., 2011; Wang et al., 2019; Lyu et al., 2019), where each training instance is associated with a set of candidate labels among which exactly one is true, reducing the overhead of finding exact label from ambiguous candidates. This problem arises in many real-world tasks such as automatic image annotation (Chen et al., 2018), web mining (Luo & Orabona, 2010), etc.

Related research on PLL was pioneered by a multiple-label learning (MLL) method (Jin & Ghahramani, 2003). Despite the same form of supervision information, i.e., a set of candidate labels, that each training instance is assigned with, a vital difference between MLL and PLL is that the goal of PLL is identifying the only one true label among candidate labels whereas for MLL identifying an arbitrary label in the candidate label set is acceptable, i.e., treating all candidate labels equally. In practice, Jin & Ghahramani (2003) formulated an unconstrained learning objective to minimize the KL divergence between the class prior and the model-based conditional distribution, and solved it by EM algorithm, resulting in a procedure iterating between estimating the prior distribution and training the model. Jin & Ghahramani (2003) used the class prior as the fitting target of the model, in which the label with maximum prior probability can be naturally regarded as the “true label”. Coincidentally, the true label of training instance in PLL is obscured by candidate labels that hinder the learning. The key to success is also identifying the true label. Therefore, the foundational work enlightened the successors to identify the true label reasonably. Along this line, great efforts have been made in designing learning objectives with some elaborate constraints for making assumptions on the model (Chen et al., 2013; 2014) or better leveraging the prior knowledge (Nguyen & Caruana, 2008; Zhang & Yu, 2015; Yu & Zhang, 2017; Gong et al., 2017; Feng & An, 2019a) so as to recover the true labeling information.

However, the constrained learning objectives of the existing methods are coupled to some specific optimization algorithms. A non-linear time complexity in the total volume of data becomes a major limiting factor when these methods envision large amounts of training data. Furthermore, Bottou & Bousquet (2007) proved stochastic optimization,
which few of the existing methods are compatible with, is
the best choice for large-scale learning problems consid-
ering the estimation-optimization tradeoff. Therefore, the PLL
problem is still practically challenging in scaling up to big
data. In order to magnify PLL’s potential on big data, we retheck critically whether the original unconstrained method
in Jin & Ghahramani (2003) should be discarded, or is worth
further studying in the deep learning era?

Our answer is in two folds. First, the decade-old method
restricted to a simple linear model fed by the handcrafted
features, which may be incompetent to represent and discrimi-
nate. Moreover, few PLL work hitherto can be generalized
to deep network architectures. We would like to advance
the previous work to enjoy the leading-edge models and
optimizers from deep learning communities (Goodfellow
et al., 2016). Second, the essence of our investigation is an
algorithm agnostic in classification models. Thus we focus
on a method that does not benefit purely from the network
architecture, but also an innovative algorithm design.

In this paper, we aim at proposing a method of PLL with flex-
ibility on the model and stochastic optimization. Towards
this goal, we first derive a classifier-consistent risk estimator,
and then propose a novel algorithm that is compatible with
arbitrary multi-class classifier (ranging from linear to deep
model) and stochastic optimizers. Experiments verify its
superiority on synthetic and real-world partial-label datasets.
Our contributions can be summarized as follows:

- Theoretically, we propose a classifier-consistent risk esti-
mator for PLL, i.e., the classifier learned from partially
labeled data converges to the optimal one learned from
ordinarily labeled data under mild conditions. Then, we estab-
lish an estimation error bound for it.
- Practically, we propose a progressive identification
method. The proposed method operates in a mini-batched
training manner where the update of the model and the
identification of true labels are accomplished seamlessly.
This method is model-independent and loss-independent,
as well as viable for any stochastic optimization (e.g.,
Robbins & Monro, 1951; Duchi et al., 2011).

2. Background

In this section, we formalize ordinary multi-class clas-
sification, partial-label learning, and complementary-label
learning, and briefly review the related work.

2.1. Ordinary Multi-Class Classification

In ordinary multi-class classification, let \( X \subseteq \mathbb{R}^d \) be
the instance space and \( Y = [c] \) be the label space, where \( d \)
is the feature space dimension, \([c] := \{1, 2, \ldots, c\}\) and \( c > 2 \)
is the number of classes. Let \( p(x, y) \) be the underlying joint
density of random variables \( (X, Y) \in X \times Y \). The target
is to learn a classifier \( g : X \rightarrow \mathbb{R}^c \) that minimizes the
estimator of the classification risk:

\[
\mathcal{R}(g) = \mathbb{E}_{(X,Y) \sim p(x,y)}[\ell(g(X), e^Y)],
\]

in which \( \ell : \mathbb{R}^c \times e^Y \rightarrow \mathbb{R} \) is a proper loss, i.e., it is
continuous non-negative and \( \ell(\eta, \eta) = 0 \) only when \( \eta = \eta \).

\( e^Y = \{e^i : i \in Y\} \) denotes the standard canonical vector in
\( \mathbb{R}^c \), i.e., the \( i \)-element in \( e^i \) equals 1 and others equal
0. Typically, the predicted label \( \hat{Y} \) is assumed to take the
following form:

\[
\hat{Y} = \arg \max_{i \in Y} g_i(X),
\]

where \( g_k(\cdot) \) is the \( k \)-th element of \( g(\cdot) \) that is interpreted
as an estimate of the true label probability, i.e., \( g_k(X) = p(Y = k | X) \).
The hypothesis in \( \mathcal{G} \) with minimal error \( g^* = \arg \min_{g \in \mathcal{G}} \mathcal{R}(g) \) is
the ordinary optimal classifier. We say a method is classifier-consistent if
the learned classifier by the method converges to \( g^* \) by increasing the example size
(Patrini et al., 2017; Xia et al., 2019).

Since \( p(x, y) \) is usually unknown, the expectation in Eq. (1) is typically approximated by
the average over the training examples \( \{(x_i, y_i)\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} p(x, y) : \hat{\mathcal{R}}(g) = \frac{1}{n} \sum_{i=1}^n [\ell(g(x_i), e^{y_i})] \), and \( \hat{g} = \arg \min_{g \in \mathcal{G}} \hat{\mathcal{R}}(g) \) is
retruned by the empirical risk minimization (ERM) principle
(Vapnik, 1998).

2.2. Partial-Label Learning

Different from ordinary multi-class classification, in PLL, the
labelling information is weakened into candidate label
set \( S \subseteq Y \), where \( S = \{P(Y)/\emptyset/\emptyset\} \) is the power set of
\( Y \) except for the empty set and the whole label set. Hence
we need to train a classifier with access only to the partially
labeled examples \( (X, S) \) drawn from the distribution with
probability density \( p(x, s) \), which is a marginal density of
the complete density \( p(x, y, s) \). Note that \( p(x, y, s) \) can be
decomposed into \( p(x, s) \) and the class-probability density
\( p(y|x, s) \), or equivalently into the marginal density \( p(x, y) \)
and the class-conditional density \( p(s|x, y) \). Then the basic
definition for PLL that the true label \( Y \) of an instance \( X \)
must be in its candidate label set \( S \) can be formulated as

\[
\Pr((X,Y) \sim p(x,y) , S \sim p(s|x,y) | Y \in S) = 1.
\]

The PLL risk estimator is defined over \( p(x, s) \):

\[
\mathcal{R}^{PLL}(g) = \mathbb{E}_{(X,S) \sim p(x,s)}[\ell^{PLL}(g(X), S)],
\]

where \( \ell^{PLL} : \mathbb{R}^c \times \mathcal{P}(Y) \rightarrow \mathbb{R} \). The goal of PLL is still
inducing a multi-class classifier to assign the true labels for
the unseen instances.

To estimate the risk in partially labeled data, defining \( \ell^{PLL} \) matters. Jin & Ghahramani (2003) defined
\( \ell_{PLL}(g(X), S) = \text{KL}[\hat{p}(y|X) || g(X)] \), in which \( \hat{p}(y|X) \) represents the unknown prior distribution, and iteratively solved it. Following Jin & Ghahramani (2003), many EM-based methods have proposed. They added some constraints to the optimization objective for better leveraging implicit information over feature space and label space, and disambiguated candidate labels via iterative refining procedure. Chen et al. (2013; 2014) proposed methods based on dictionary learning, which seeked the sparsest dictionary mapping the feature to class prior, and then Shrivastava et al. (2015) extended the linear dictionary by the kernel trick. These methods were optimized by K-SVD algorithm (Aharon et al., 2006). Feng & An (2019a) formulated a convex-concave problem and proved its optimization is equivalent to a set of QP problems. Feng & An (2019b) also formulated a QP problem to enlarge the gap of label priors between two instances with completely different candidate labels. In addition, some SVM-based methods with the maximum margin constraints (Nguyen & Caruana, 2008; Yu & Zhang, 2017) that were solved by the off-the-shelf implementation on multi-class SVM (Fan et al., 2008), and a few non-parametric methods (Hullermeier & Beringer, 2006; Zhang & Yu, 2015; Gong et al., 2017) were proposed.

These methods were solved in specific low-efficient manners and incompatible with high-efficient stochastic optimization. Thus they could hardly handle large-scale datasets. To the best of our knowledge, only the latest work (Yao et al., 2020a,b) used DNNs with stochastic optimizers as the backbone of their algorithms, however, they restricted the networks to some specific architectures while our method is flexible on the learning models, and they also lacked a theoretical understanding of their methods.

### 2.3. Complementary-Label Learning

Complementary-label learning (CLL) (Ishida et al., 2017; Yu et al., 2018; Ishida et al., 2019) uses the supervised information specifying a class that an example does not belong to, and hence it can be considered as an extreme PLL case with \( c - 1 \) candidate labels. Restricting each instance to associate with a single complementary label limits its potential because the labelers can easily annotate multiple candidate labels or complementary labels. Recently, Feng et al. (2020) proposed to learn with multiple complementary labels. Nonetheless, they proposed an explicit data generation process, while our work has no special requirement on this aspect. Empirical studies also show that our proposed method can achieve promising performance in different partial types (e.g. binomial or pair in Section 5).

### 3. Learning with Partial Labels

In this section, we define a classifier-consistent risk estimator, and theoretically establish an estimation error bound.

#### 3.1. Classifier-Consistent Risk Estimator

To make Eq. (3) estimable, an intuitive way is through a surrogate loss that treats all the candidate labels equally (Jin & Ghahramani, 2003): \( \ell_{PLL}(g(X), S) = \frac{1}{|S|} \sum_{i \in S} \ell(g(X), e^i) \). Nonetheless, the true label may be overwhelmed by the distractive outputs of the multiple false positive labels. Therefore, we consider that only the true label contributes to retrieving the classifier. Capturing this idea we define the PLL loss as the minimal loss over the candidate label set:

\[
\ell_{PLL}(g(X), S) = \min_{i \in S} \ell(g(X), e^i),
\]

which immediately leads to a new risk estimator:

\[
R_{PLL}(g) = E_{(X,S) \sim p(x,s)} \min_{i \in S} \ell(g(X), e^i).
\]

For a detailed description of Eq. (4), we can disassemble it by defining a function in terms of \( Y \) as a classifier \( g \)’s best guess at the true label:

\[
Y^g = \arg \min_{i \in S} \ell(g(X), e^i).
\]

Then correspondingly the PLL loss can be rewritten into an equivalent expression, i.e., \( g \) only needs to fit the best guess \( Y^g \) among \( S \):

\[
\ell_{PLL}(g(X), S) = \min_{i \in S} \ell(g(X), e^i) = \ell(g(X), e^{Y^g}).
\]

From now on, we explain under which conditions Eq. (5) is classifier-consistent. We start investigating this question from the following two lemmas.

**Lemma 1.** (Summarized from Sec.3.1 of Liu & Dietterich, 2014) The ambiguity degree is defined as

\[
\gamma = \sup_{(x,y) \sim p(x,y)} \Pr(Y \neq \hat{Y} | X = x, Y \sim p(x,y), \hat{Y} \neq Y).
\]

If \( \gamma < 1 \), i.e. under the small ambiguity degree condition, the PLL problem is ERM learnability.

\( \gamma \) is the maximum probability of a negative label \( \hat{Y} \) co-occurs with the true label \( Y \). The small ambiguity degree condition implies that except for the true label, no other labels will be one hundred percent included in the candidate label set, which guarantees a classification error made on any instance will be detected with probability at least \( 1 - \gamma \).

Then we provide a condition of the identifiability of the ordinary optimal classifier.

**Lemma 2.** (Yu et al., 2018) If \( \ell \) is the cross-entropy loss or mean squared error loss, the optimal classifier \( g^\ast \) of Eq. (1) satisfies \( g^\ast_{Y}(X) = p(Y = i|X) \).

With the above lemmas, we can prove Eq. (5) possess classifier-consistency under a reasonable assumption, that is, learning is conducted under the deterministic case.
Theorem 1. Under the deterministic scenario, if the small ambiguity degree condition is satisfied, and cross-entropy or mean squared error loss is used, then, the PLL optimal classifier $g^*_{PLL}$ of Eq. (5) is equivalent to the ordinary optimal classifier $g^*$ of Eq. (1), i.e., $g^*_{PLL} = g^*$.

It is natural to assume a deterministic learning scenario, i.e., the true label $Y_X$ of an instance is uniquely determined by the measurable function $g^*: p(Y = i|X) = g^*_i(X), Y_X = \arg \max_{c\in Y} g^*_c(X)$, for ensuring the basic definition for PLL, i.e., Eq. (2) always holds. In the light of Lemma 2, $g^*$ can be recovered by minimizing Eq. (5). The proof is given in Appendix.

3.2. Estimation Error Bound

Let $\tilde{R}_{PLL}$ be the empirical counterpart of $R_{PLL}$, and $\hat{g}_{PLL} = \arg \min \tilde{R}_{PLL}(g)$ be the empirical risk classifier. Suppose $G_c$ be a class of real functions, and $G = \oplus y \in [G_c]$ be a c-vale function class. Assume there is $C_g > 0$ such that $\sup_{y \in [G_c]} ||g||_* \leq C_g$, and the loss function $\ell(g(X), Y)$ is Lipschitz continuous for all $|g| \leq C_g$ with a Lipschitz constant $L_1$ and upper-bounded by $M$, i.e., $M = \sup_{X \in [X_c], |g| \leq C_g, Y \in Y} \ell(g(X), Y)$ The Rademacher complexity of $G$ over $p(x)$ with sample size $n$ is defined as $R_n(G)$ (Bartlett & Mendelson, 2002). Then we have the following estimation error bound.

Theorem 2. For any $\delta > 0$, we have with probability at least $1 - \delta$,

$$R_{PLL}(\hat{g}_{PLL}) - R_{PLL}(g^*_{PLL}) \leq 4\sqrt{2eL_1} \sum_{y=1}^c g_n(G_y) + 2M \sqrt{\log(2/\delta)}{2n}.$$  

The proof is given in Appendix. As $n \to \infty$, $R_n(G_y) \to 0$ for all parametric models with a bounded norm. Hence, $R_{PLL}(\hat{g}_{PLL}) \to R_{PLL}(g^*_{PLL})$ as the number of training data approaches infinity.

4. Benchmark Solution

In the previous section, based on a novel PLL loss Eq. (4) we provided a risk estimator Eq. (5) with intriguing theoretical properties. Although the estimator is without any assumption on the model used, the min operator makes optimization difficult, because if a wrong label $i$ in Eq. (5) is selected in the beginning, the optimization will focus on the wrong label till the end. Thus we design a benchmark solution to approximately solve Eq. (5) according to the idea behind Eq. (4) that only one (true) label should be taken into account, and consequently we can easily implement the algorithm over arbitrary multi-class classifier and powerful stochastic optimization.

Algorithm 1 PRODEN Algorithm

\begin{itemize}
    \item \textbf{input} $D$: the partial-label training set $\{(x_i, s_i)\}_{i=1}^n$;
    \item \textbf{input} $T$: number of epochs;
    \item \textbf{output} $\Theta$: model parameter for $g(x; \Theta)$;
    \item 1: Let $\mathcal{A}$ be an stochastic optimization algorithm;
    \item 2: Initialize uniform weights $w^0$ according to Eq. (9);
    \item 3: for $t = 1$ to $T$ do
    \item 4: Shuffle training set $D$ into $B$ mini-batches;
    \item 5: for $k = 1$ to $B$ do
    \item 6: Compute $L$ according to Eq. (8);
    \item 7: Set gradient $-\nabla L$;
    \item 8: Update $w$ according to Eq. (10);
    \item 9: Update $\Theta$ by $\mathcal{A}$;
    \item 10: \textbf{end for}
    \item 11: \textbf{end for}
\end{itemize}

In particular, we relax the min operator in Eq. (4) by the dynamic weights. In the first place, we require that $\ell$ can be decomposed into each label, i.e.,

$$\ell(g(X), e^Y) = \sum_{i=1}^c \ell(g_i(X), e^Y_i),$$

where $e^Y_i$ is the $i$-th element of $e^Y$. Many commonly used multi-class loss functions are decomposable, for example, $\ell$ can be the cross-entropy loss: $\ell_{CE}(g(X), e^Y) = -\sum_{i=1}^c e^Y_i \log(g_i(X))$ or mean squared error loss: $\ell_{MSE}(g(X), e^Y) = \sum_{i=1}^c (g_i(X) - e^Y_i)^2$. In this case, the empirical risk estimator is rewritten as follows:

$$\hat{R}_{PLL} = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^c w_{ij} \ell(g_j(x_i), e^*_{sj}),$$

where $e^*_{sj}$ is the $j$-th coordinate of $e^*_{si}$ and $e^*_{si} = \sum_{k \in s_i} e^k$. $w_i \in \Delta^{c-1}$ is the $c$-dimensional simplex and the $j$-th element of it, i.e., $w_{ij}$, is the corresponding label weight, i.e., the confidence of the $j$-th label being the true label of the $i$-th instance. With appropriate weights, i.e., $w_{ij} = 1$ if $\arg \min_{k \in s_i} \ell(g(x_i), e^k) = j$ and 0 otherwise, which is in consonance with Eq. (6), Eq. (8) is equivalent to the empirical counterpart of Eq. (5). Ideally, the label with weight 1 is exactly the true label, i.e, $j$ is the true label of $x_i$ if $w_{ij} = 1$, which means we have identified the true label successfully. To eventually achieve such an ideal situation, we propose an effective algorithm named PRODEN (PROgressive IDENTification).

Because the weights are latent, the minimizer of Eq. (8) cannot be solved directly. Inspiring by the EM algorithm,
we start by training an initial model with uniform weights:

$$w_{ij} = \begin{cases} 1/|s_i| & \text{if } j \in s_i, \\ 0 & \text{otherwise}. \end{cases}$$  \tag{9}

Thanks to the memorization effects (Arpit et al., 2017; Han et al., 2018), the networks will remember “frequent patterns” in the first few iterations. If the small ambiguity degree condition is satisfied, they tend to remember the true labels in the initial epochs, which guides us towards a discriminative classifier giving relatively low losses for more possible true labels. Next, the problem becomes how to retain the informative predictions and gradually filter out the false positive labels. Instead of estimating the weights in a separate E-step, we tackle it simply using the current predictions for slightly putting more weights on more possible labels:

$$w_{ij} = \begin{cases} g_j(x_i)/\sum_{k\in s_i} g_k(x_i) & \text{if } j \in s_i, \\ 0 & \text{otherwise}. \end{cases}$$  \tag{10}

In this fashion, the true labels are progressively identified, and the refined labels in turn help to improve the classifier. The overall procedure is summarized in Algorithm 1, whose identification step (Step 8) and optimization step (Step 9) are conducted in a seamless manner. Next we provide more detailed arguments about the superiority and generalization of the proposed method.

Firstly, PRODEN gets rid of the overfitting issues of EM methods. The previous iterative EM methods (Jin & Ghahramani, 2003; Tang & Zhang, 2017; Feng & An, 2019a) trained the model until convergence in the M-step, but overemphasizing the convergence may result in redundant computation and overfitting issues, as the model will eventually fit the initial inexact prior knowledge and make a less informative estimate in the E-step on which the subsequent learning is based. To mitigate these problems, our method advances the procedure by merging the E-step and M-step. Since the learning process has no clear separation of the E-step and M-step, the weights can be updated at any epoch such that the local convergence within each epoch is not necessary in our training process.

Secondly, PRODEN has great flexibility for models and loss functions. In the deep learning era, loss functions with different theoretical results are one of the key factors that affect the performance of DNNs (Ghosh et al., 2017). In this way, models are welcomed to be loss-independent that allows flexibility with any loss function (Ishida et al., 2019). However, existing PLL methods were restricted to some specific loss functions for optimizable, e.g., Jin & Ghahramani (2003) limited the loss function to the KL divergence; the loss function of Yao et al. (2020b) was a combination of multiple cross-entropy losses. Instead, our proposal is flexible enough to be compatible with a large group of decomposable losses. Moreover, we will show in Appendix that the proposed method is provably a generalization of Jin & Ghahramani (2003).

5. Experiments

In this section, we experimentally analyze the proposed method PRODEN, and compare with state-of-the-art PLL methods. The implementation is based on PyTorch (Paszke et al., 2019) and experiments were carried out with NVIDIA Tesla V100 GPU; it is available at https://github.com/Lvcrezia77/PRODEN.

Datasets We use four widely used benchmark datasets including MNIST (LeCun et al., 1998), Fashion-MNIST (H. Xiao & Vollgraf, 2017), Kuzushiji-MNIST (Clanuwat et al., 2018), and CIFAR-10 (Krizhevsky & Hinton, 2009), and five datasets from the UCI Machine Learning Repository (Krizhevsky & Hinton, 2009), including Yeast, Texture, Dermatology, Synthetic Control, and 20Newgroups. Similar to the corrupted strategy in label noise (Patrini et al., 2017; Han et al., 2018), we manually corrupt these datasets into partially labeled versions by a flipping probability $q$ where $q = \Pr(\hat{y} = 1|y = 0)$ gives the probability that false positive label $\hat{y}$ is flipped from negative label $y$. We adopt a binomial flipping strategy: $c - 1$ independent experiments are conducted on all training examples, each determining whether a negative label is flipped with probability $q$. Then for the examples that none of the negative labels is flipped, we additionally flip a random negative label to the candidate label set for ensuring all the training examples are partially labeled. In this paper, we conduct the experiments under both less-partial circumstances $q = 0.1$ and strong-partial circumstances $q = 0.7$. In addition, five real-world partial-label datasets are adopted, including Lost (Cour et al., 2011), Birdsong (Briggs et al., 2012), MSR-Cv2 (Liu & Dietterich, 2012), Soccer Player (Zeng et al., 2013), and Yahoo! News (Guillaumin et al., 2010).

Baselines We first analyze PRODEN by comparing it with seven baseline methods based on DNNs, including three variants of PRODEN, two supervised methods, and two state-of-the-art PLL methods:

- PRODEN-itera means updating the label weights in the iterative EM manner instead of a seamless manner proposed in Section 4.
- PRODEN-sudden means performing a sudden identification, i.e., updating the weights $w_{ik} = 1$ if $\arg \max_{j \in s_i} g_j(x_i) = k$ and $w_{ij} = 0, \forall j \neq k$ in every iteration step.
- PRODEN-naive means never updating the uniform weights.
- PN-oracle means supervised learning from ordinary labels. It should achieve the best performance, and is used merely for a proof of concept.
We further compare PRODEN with six classical PLL methods.

- **PN-decomp** means decomposing one instance with multiple candidate labels into many (same) instances each one single label, so that we could use any ordinary multi-class classification methods.

- **$D^2$CNN** (Yao et al., 2020b) means a PLL learning method based on DNNs.

- **GA** (Ishida et al., 2019) means a CLL method based on DNNs and gradient ascent.

We employ multiple base models, including linear model, 5-layer perceptron (MLP), 12-layer ConvNet (Laine & Aila, 2017) and 32-layer ResNet (He et al., 2016) to show that our proposal is compatible with a wide family of learning models. The detailed descriptions of the datasets with their corresponding models are provided in Appendix. The optimizer is stochastic gradient descent (SGD) (Robbins & Monro, 1951) with momentum 0.9. We train each model 500 epochs with softmax function and cross-entropy loss. And PRODEN-itera updates label weights every 100 epochs.

We further compare PRODEN with six classical PLL methods that can hardly be implemented by DNNs on small-scale datasets.
datasets (UCI and real-world datasets). They are four parametric methods: SURE (Feng & An, 2019a), CLPL (Cour et al., 2011), ECOC (Zhang et al., 2017), PLSVM (Nguyen & Caruana, 2008), and two non-parametric methods: PLkNN (Hullermeier & Beringer, 2006) and IPAL (Zhang & Yu, 2015). Their hyper-parameters are selected according to the suggested parameter settings in original papers. Detailed information is provided in Appendix. For a fair comparison, PRODEN and all the parametric methods employ a linear model. We average the classification accuracy of PRODEN over the last 10 epochs as the results to prove that PRODEN is always stable and will not cause performance degradation due to overfitting when the number of epochs increases.

**Results on benchmark datasets** We record inductive results which indicate the classification accuracy in the test set (test accuracy). The means and standard deviations of test accuracy based on 5 random samplings in the binomial case are shown in Figure 1. The transductive accuracy which reflects the ability in identifying the true labels in the training set can be found in Appendix.

We first observe the performance when $q = 0.1$ (the left two columns). PRODEN is always the best method and comparable to PN-oracle with all the models. PRODEN-itera is comparable to PRODEN with a linear model, but its performance deteriorates drastically with complex models because of the overfitting issues. This phenomenon is consistent with the discussions in Section 4. In PRODEN-naive, the weights are always equally distributed, while in PRODEN-sudden, the sudden identification concentrates all the weights to the winners, resulting in their poorer performance.

Next, we compare these methods when $q = 0.7$ (the right two columns). This is significantly harder than $q = 0.1$. But when the task difficulty is relatively low (e.g. MNIST, Fashion-MNIST), PRODEN is still comparable to PN-oracle with such a large flipping probability, while the performance of the baselines is greatly reduced. The superiority always stands out for PRODEN compared with two latest deep methods GA and D$^2$CNN.

**Analysis on the ambiguity degree** It is intuitive that the higher the ambiguity degree is, the more difficult it is to find the true labels. We investigate the influence of ambiguity degree $\gamma$ through a pair flipping strategy: only the classes that are similar to the true class will be put into the candidate label set with probability $q$. We gradually move $q$ from 0.5 to 0.9 to simulate $\gamma (\gamma \rightarrow q$ as $n \rightarrow \infty$), and the experimental results on Kuzushiji-MNIST are reported in Figure 2, and phenomena on other datasets are similar and reported in Appendix. We can see that even when the ambiguity degree is high, PRODEN is still highly competitive. PRODEN tends to be less affected with increased ambiguity, while the baselines are affected severely.

**Results on small-scale datasets** We perform five-fold cross-validation, and use paired $t$-test at 5% significance level. Table 1 and Table 2 report the mean test accuracy with standard deviation on UCI datasets and the real-world datasets, respectively. ●/○ represents whether PRODEN is significantly better/worse than the comparing methods. Clearly, PRODEN is overall the best performing, confirming the advantage of PRODEN afforded not only by the network architecture, but also the progressive identification process.
Progressive Identification of True Labels for Partial-Label Learning

Table 1. Test accuracy (mean±std) on the UCI datasets in the binomial case.

<table>
<thead>
<tr>
<th></th>
<th>q</th>
<th>Yeast</th>
<th>Texture</th>
<th>Dermatology</th>
<th>Synthetic Control</th>
<th>20Newsgroups</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODEN</td>
<td>0.1</td>
<td>59.05±4.66%</td>
<td>99.64±0.16%</td>
<td>96.16±3.26%</td>
<td>98.33±1.32%</td>
<td>77.28±0.68%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>55.15±3.87%</td>
<td>99.33±0.32%</td>
<td>95.34±3.15%</td>
<td>95.00±4.75%</td>
<td>64.74±0.90%</td>
</tr>
<tr>
<td>PRODEN-itera</td>
<td>0.1</td>
<td>57.55±3.96%</td>
<td>99.73±0.13%</td>
<td>95.56±2.90%</td>
<td>82.38±4.18%</td>
<td>77.04±0.77%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>53.62±5.54%</td>
<td>94.79±1.63%</td>
<td>76.22±4.48%</td>
<td>48.33±7.96%</td>
<td>52.43±0.75%</td>
</tr>
<tr>
<td>GA</td>
<td>0.1</td>
<td>25.43±3.81%</td>
<td>96.77±0.51%</td>
<td>73.53±7.73%</td>
<td>64.70±0.42%</td>
<td>66.53±2.60%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>22.59±2.51%</td>
<td>95.76±0.62%</td>
<td>52.00±12.32%</td>
<td>49.72±6.38%</td>
<td>56.47±0.59%</td>
</tr>
<tr>
<td>D^2CNN</td>
<td>0.1</td>
<td>59.05±4.66%</td>
<td>98.80±0.31%</td>
<td>95.34±2.29%</td>
<td>84.50±11.28%</td>
<td>73.20±0.46%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>55.15±3.87%</td>
<td>97.23±0.71%</td>
<td>91.59±1.08%</td>
<td>71.00±13.02%</td>
<td>52.13±0.41%</td>
</tr>
<tr>
<td>SURE</td>
<td>0.1</td>
<td>55.52±4.92%</td>
<td>97.96±0.32%</td>
<td>95.16±2.25%</td>
<td>76.67±2.83%</td>
<td>69.82±0.26%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>49.32±3.95%</td>
<td>94.75±3.68%</td>
<td>90.96±2.02%</td>
<td>54.67±7.67%</td>
<td>62.66±1.21%</td>
</tr>
<tr>
<td>CLPL</td>
<td>0.1</td>
<td>57.90±4.35%</td>
<td>98.78±0.22%</td>
<td>94.79±3.27%</td>
<td>78.00±3.10%</td>
<td>76.26±0.77%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>54.88±7.78%</td>
<td>95.09±1.01%</td>
<td>92.33±2.67%</td>
<td>63.33±7.02%</td>
<td>50.37±0.57%</td>
</tr>
<tr>
<td>ECOC</td>
<td>0.1</td>
<td>59.01±3.72%</td>
<td>99.15±0.27%</td>
<td>94.71±2.08%</td>
<td>96.67±2.08%</td>
<td>77.67±1.11%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>53.37±2.37%</td>
<td>97.69±0.82%</td>
<td>91.47±3.02%</td>
<td>91.67±2.08%</td>
<td>61.32±2.45%</td>
</tr>
<tr>
<td>PLSVM</td>
<td>0.1</td>
<td>54.73±2.89%</td>
<td>93.75±1.99%</td>
<td>92.88±2.25%</td>
<td>92.50±2.12%</td>
<td>76.25±1.22%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>42.34±1.92%</td>
<td>50.69±5.62%</td>
<td>89.32±4.98%</td>
<td>85.50±2.61%</td>
<td>59.12±0.41%</td>
</tr>
<tr>
<td>PL&amp;NN</td>
<td>0.1</td>
<td>54.28±3.64%</td>
<td>97.20±0.28%</td>
<td>94.52±2.74%</td>
<td>94.83±2.08%</td>
<td>42.39±0.97%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>28.72±1.01%</td>
<td>96.78±3.10%</td>
<td>85.48±3.70%</td>
<td>85.50±6.36%</td>
<td>18.01±0.34%</td>
</tr>
<tr>
<td>IPAL</td>
<td>0.1</td>
<td>50.56±2.83%</td>
<td>99.33±0.26%</td>
<td>95.34±2.29%</td>
<td>98.33±0.83%</td>
<td>75.01±0.69%</td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td>41.10±3.71%</td>
<td>97.84±0.57%</td>
<td>95.02±2.03%</td>
<td>94.83±6.55%</td>
<td>56.93±0.50%</td>
</tr>
</tbody>
</table>

Table 2. Test accuracy (mean±std) on the real-world datasets.

<table>
<thead>
<tr>
<th></th>
<th>Lost</th>
<th>Birdsong</th>
<th>MSRCv2</th>
<th>Soccer Player</th>
<th>Yahoo! News</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODEN</td>
<td>76.57±1.47%</td>
<td>72.01±0.44%</td>
<td>45.27±1.73%</td>
<td>55.99±0.58%</td>
<td>67.40±0.55%</td>
</tr>
<tr>
<td>PRODEN-itera</td>
<td>68.09±3.78%</td>
<td>68.02±0.76%</td>
<td>42.79±2.80%</td>
<td>53.50±0.94%</td>
<td>67.04±0.67%</td>
</tr>
<tr>
<td>D^2CNN</td>
<td>69.61±5.48%</td>
<td>66.58±1.49%</td>
<td>40.17±1.99%</td>
<td>49.06±0.15%</td>
<td>56.39±0.89%</td>
</tr>
<tr>
<td>GA</td>
<td>48.21±4.44%</td>
<td>29.77±1.43%</td>
<td>22.30±2.71%</td>
<td>51.88±0.44%</td>
<td>34.32±0.95%</td>
</tr>
<tr>
<td>SURE</td>
<td>71.61±3.44%</td>
<td>58.04±1.22%</td>
<td>31.57±2.48%</td>
<td>49.16±0.20%</td>
<td>45.73±0.90%</td>
</tr>
<tr>
<td>CLPL</td>
<td>76.17±1.81%</td>
<td>67.56±1.12%</td>
<td>43.64±0.24%</td>
<td>49.88±4.29%</td>
<td>53.74±0.95%</td>
</tr>
<tr>
<td>ECOC</td>
<td>63.93±5.45%</td>
<td>71.47±1.24%</td>
<td>46.78±2.84%</td>
<td>55.51±0.54%</td>
<td>64.78±0.78%</td>
</tr>
<tr>
<td>PLSVM</td>
<td>72.86±5.45%</td>
<td>60.46±1.99%</td>
<td>38.97±4.62%</td>
<td>46.15±1.00%</td>
<td>60.46±1.48%</td>
</tr>
<tr>
<td>PL&amp;NN</td>
<td>35.00±4.71%</td>
<td>64.22±1.14%</td>
<td>41.60±2.30%</td>
<td>49.18±0.26%</td>
<td>40.30±0.90%</td>
</tr>
<tr>
<td>IPAL</td>
<td>71.25±1.40%</td>
<td>71.19±1.54%</td>
<td>52.36±2.87%</td>
<td>54.41±0.56%</td>
<td>66.22±0.80%</td>
</tr>
</tbody>
</table>

6. Conclusion

In this paper, we focused on proposing a novel method for PLL which is compatible with flexible multi-class classifiers and stochastic optimization. We first proposed a classifier-consistent risk estimator based on the minimal loss incurred by candidate labels. The estimator has good theoretical properties, while the min operator makes it difficult to be solved by the epoch-wise optimization of neural networks. Therefore, we proposed a progressive identification method named PRODEN for approximately minimizing the proposed risk estimator, whose idea is relaxing min to a weighted combination. Learning the weights and the classifier are then conducted in a seamless manner for mitigating the overfitting problem. At last, experiments demonstrated that the proposed method could successfully train various models, and it compared favorably with state-of-the-art methods.
Acknowledgements

JL and XG were supported by the National Key Research & Development Plan of China (No. 2017YFB1002801), the National Science Foundation of China (6162203), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Collaborative Innovation Center of Wireless Communications Technology. MX was supported by the Open Project Program of the State Key Lab. for Novel Software Technology, Nanjing University, Nanjing, 210093, P.R.China. LF was supported by MOE, NRF, and NTU, Singapore. GN and MS were supported by JST AIP Acceleration Research Grant Number JPMJCR20U3, Japan.

References


Guillaumin, M., Verbeek, J., and Schmid, C. Multiple instance metric learning from automatically labeled bags
Progressive Identification of True Labels for Partial-Label Learning


Progressive Identification of True Labels for Partial-Label Learning


Supplementary Material

A. Proof of Lemma 2

**Cross-Entropy loss**  According to (Masnadi-Shirazi & Vasconcelos, 2009), since the $\ell_{CE}$ is non-negative, minimizing the conditional risk $E_{p(y|x)}[\ell_{CE}(g(X), Y)|X]$, $\forall X \in \mathcal{X}$, is an alternative of minimizing $R(g)$. The conditional risk can be written as

$$C(g) = - \sum_{i=1}^{c} p(Y = i|X) \log(g_i(X)), \quad \text{s.t.} \quad \sum_{i=1}^{c} g_i(X) = 1.$$  

By the Lagrange Multiplier method (Bertsekas, 1997), we have

$$= - \sum_{i=1}^{c} p(Y = i|X) \log(g_i(X)) + \lambda \left( \sum_{i=1}^{c} g_i(X) - 1 \right).$$

To minimize , we take the partial derivative of $C$ with respect to $g_i$ and set it be 0:

$$g_i^*(X) = \frac{1}{\lambda} p(Y = i|X).$$

Because $\sum_{i=1}^{c} g_i^*(X) = 1$ and $\sum_{i=1}^{c} g_i^*(X) = 1$, we have

$$\sum_{i=1}^{c} g_i^*(X) = \frac{1}{\lambda} \sum_{i=1}^{c} p(Y = i|X) = 1.$$  

Therefore, we can obtain $\lambda = 1$ that ensures $g_i^*(X) = p(Y = i|X), \forall i \in [c], \forall X \in \mathcal{X}$, which concludes the proof.

**Mean squared error loss**  Analogously, if the mean squared error loss is used, we can write the optimization problem as

$$C(g) = \sum_{i=1}^{c} (p(Y = i|X) - g_i(X))^2, \quad \text{s.t.} \quad \sum_{i=1}^{c} g_i(X) = 1.$$  

By the Lagrange Multiplier method, we have

$$= \sum_{i=1}^{c} (p(Y = i|X) - g_i(X))^2 - \lambda \left( \sum_{i=1}^{c} g_i(X) - 1 \right).$$

By setting the derivative to 0, we obtain

$$g_i^*(X) = \frac{\lambda'}{2} + p(Y = i|X).$$

Because $\sum_{i=1}^{c} g_i^*(X) = 1$ and $\sum_{i=1}^{c} g_i^*(X) = 1$, we have

$$\sum_{i=1}^{c} g_i^*(X) = \frac{\lambda'c}{2} + \sum_{i=1}^{c} p(Y = i|X).$$

Since $c \neq 0$, we can obtain $\lambda' = 0$. In this way, $g_i^*(X) = p(Y = i|X), \forall i \in [c], \forall X \in \mathcal{X}$, which concludes the proof. □
B. Proof of Theorem 1

First we prove $g^*$ is the optimal classifier for PLL by substituting the $g^*$ into the PLL risk estimator Eq. (5):

$$R_{PLL}(g^*) = E_{(X,S) \sim p(x,s)} \min_{i \in S} \ell(g^*(X), e_i) = \int \sum_{S \in S} \min_{i \in S} \ell(g^*(X), e_i)p(s|x)p(x)dX$$

$$= \int \sum_{Y \in Y} \sum_{S \in S} \min_{i \in S} \ell(g^*(X), e_i)p(s|x)p(y|x)p(x)dX$$

$$= \int \sum_{Y \in Y} \ell(g^*(X), e^y_p)p(s|x)p(y|x)p(x)dX$$

$$= \int \sum_{Y \in Y} \ell(g^*(X), e^y_p)p(x,y)dX = R(g^*) = 0.$$ 

where we have used $\min_{i \in S} \ell(g^*(X), e_i) = \ell(g^*(X), e^y_p)$ because $\ell$ is a proper loss and the deterministic assumption is made. This indicates that the PLL risk has been minimized by $g^*$.

On the other hand, we prove $g^*$ is the only solution to Eq. (5) by contradiction, namely, there is at least one other solution $h$ enables $R_{PLL}(h) = 0$, and predicts different label $Y^h \neq Y_X$ for at least one instance $X$. Hence for any $S \ni Y_X$ we have

$$\min_{i \in S} \ell(h(X), e_i) = \ell(h(X), e^Y) = 0.$$ 

Nevertheless, the above equality is always true unless $Y^h$ is invariably included in the candidate label set of $X$, i.e., $Pr_{S \ni p(x,y)}(Y^h \in S) = 1$. Obviously, this contradicts the small ambiguity degree condition. Therefore, there is one, and only one minimizer of the PLL risk estimator, which is the same as the minimizer learned from ordinarily labeled data. The proof is complete.

C. Proof of Theorem 2

First, we show the uniform deviation bound, which is useful to derive the estimation error bound.

**Lemma 3.** For any $\delta > 0$, we have with probability at least $1 - \delta$,

$$\sup_{g \in G} \left| R_{PLL}(g) - \tilde{R}_{PLL}(g) \right| \leq 2\mathcal{R}_n(\ell_{PLL} \circ G) + M \sqrt{\frac{\log(2/\delta)}{2n}}$$

**Proof.** Consider the one-side uniform deviation $\sup_{g \in G} R_{PLL}(g) - \tilde{R}_{PLL}(g)$. Since the loss function $\ell$ is upper-bounded by $M$, the change of it will be no more than $M/n$ after replacing some $x$. Then, by McDiarmid’s inequality (McDiarmid, 1989), for any $\delta > 0$, with probability at least $1 - \delta/2$, the following holds:

$$\sup_{g \in G} R_{PLL}(g) - \tilde{R}_{PLL}(g) \leq E \left[ \sup_{g \in G} R_{PLL}(g) - \tilde{R}_{PLL}(g) \right] + M \sqrt{\frac{\log(2/\delta)}{2n}}.$$ 

By symmetrization (Vapnik, 1998), it is a routine work to show that

$$E \left[ \sup_{g \in G} R_{PLL}(g) - \tilde{R}_{PLL}(g) \right] \leq 2\mathcal{R}_n(\ell_{PLL} \circ G).$$

The one-side uniform deviation $\sup_{g \in G} R_{PLL}(g) - \tilde{R}_{PLL}(g)$ can be bounded similarly.

Then we upper bound $\mathcal{R}_n(\ell_{PLL} \circ G)$. 


Lemma 4. Suppose $\ell_{PLL}$ is defined as Eq. (4), it holds that

$$\mathcal{R}_n(\ell_{PLL} \circ \mathcal{G}) \leq c \mathcal{R}_n(\ell \circ \mathcal{G}) \leq \sqrt{2cL_{\ell}} \sum_{y=1}^{n} \mathcal{R}_n(\mathcal{G}_y).$$

Proof. By definition of $\ell_{PLL}$, $\ell_{PLL} \circ \mathcal{G}(x_i, S_i) = \min_{y \in S} \ell \circ \mathcal{G}(x_i, y) = \min_{y \in [c]} \ell \circ \mathcal{G}(x_i, y)$. Given sample sized $n$, we first prove the result in the case $c = 2$. The min operator can be written as

$$\min\{z_1, z_2\} = \frac{1}{2}[z_1 + z_2 - |z_1 - z_2|].$$

In this way, we can write

$$\mathcal{R}_n(\ell_{PLL} \circ \mathcal{G}) = \mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(g(x_i), s_i) \right]$$

(11)

$$= \mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \min\{\ell(g(x_i), y_1), \ell(g(x_i), y_2)\} \right]$$

$$= \mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{2n} \sum_{i=1}^{n} \sigma_i \left[\ell(g(x_i), y_1) + \ell(g(x_i), y_2) - |\ell(g(x_i), y_1) - \ell(g(x_i), y_2)|\right] \right]$$

$$\leq \mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{2n} \sum_{i=1}^{n} \sigma_i \ell(g(x_i), y_1) \right] + \mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{2n} \sum_{i=1}^{n} \sigma_i \ell(g(x_i), y_2) \right]$$

$$+ \mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{2n} \sum_{i=1}^{n} \sigma_i |\ell(g(x_i), y_1) - \ell(g(x_i), y_2)| \right]$$

$$= \frac{1}{2} \left( \mathcal{R}_n(\ell \circ \mathcal{G}) + \mathcal{R}_n(\ell \circ \mathcal{G}) \right) + \mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{2n} \sum_{i=1}^{n} \sigma_i |\ell(g(x_i), y_1) - \ell(g(x_i), y_2)| \right].$$

Since $x \mapsto |x|$ is a 1-Lipschitz function, by Talagrand’s contraction lemma (Ledoux & Talagrand, 2013), the last term can be bounded:

$$\mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{2n} \sum_{i=1}^{n} \sigma_i |\ell(g(x_i), y_1) - \ell(g(x_i), y_2)| \right]$$

(12)

$$\leq \mathbb{E}_{\sigma} \left[ \sup_{g \in \mathcal{G}} \frac{1}{2n} \sum_{i=1}^{n} \sigma_i (\ell(g(x_i), y_1) - \ell(g(x_i), y_2)) \right] \leq \frac{1}{2} \left( \mathcal{R}_n(\ell \circ \mathcal{G}) + \mathcal{R}_n(\ell \circ \mathcal{G}) \right).$$

Combining Eq. (11) and Eq. (12) yields $\mathcal{R}_n(\ell_{PLL} \circ \mathcal{G}) \leq \mathcal{R}_n(\ell \circ \mathcal{G}) + \mathcal{R}_n(\ell \circ \mathcal{G})$. The general case can be derived from the case $c = 2$ using $\min\{z_1, \ldots, z_c\} = \min\{z_1, \min\{z_2, \ldots, z_c\}\}$ and an immediate recurrence.

Then we apply the Rademacher vector contraction inequality (Maurer, 2016),

$$\mathcal{R}_n(\ell \circ \mathcal{G}) \leq \sqrt{2L_{\ell}} \sum_{y=1}^{c} \mathcal{R}_n(\mathcal{G}_y).$$

The proof is completed. □
Table 3. Summary of benchmark datasets and models.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Train</th>
<th># Test</th>
<th># Feature</th>
<th># Class</th>
<th>Model g(x; θ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>60,000</td>
<td>10,000</td>
<td>784</td>
<td>10</td>
<td>Linear model, MLP (depth 5)</td>
</tr>
<tr>
<td>Fashion-MNIST</td>
<td>60,000</td>
<td>10,000</td>
<td>784</td>
<td>10</td>
<td>Linear model, MLP (depth 5)</td>
</tr>
<tr>
<td>Kuzushiji-MNIST</td>
<td>60,000</td>
<td>10,000</td>
<td>784</td>
<td>10</td>
<td>Linear model, MLP (depth 5)</td>
</tr>
<tr>
<td>CIFAR-10</td>
<td>50,000</td>
<td>10,000</td>
<td>3,072</td>
<td>10</td>
<td>ConvNet (Laine &amp; Aila, 2017), ResNet (He et al., 2016)</td>
</tr>
</tbody>
</table>

Based on Lemma 3 and 4, the estimation error bound Eq. (7) is proven through

\[
\mathcal{R}_{PLL}(\hat{g}_{PLL}) - \mathcal{R}_{PLL}(g^*_{PLL}) = \left( \mathcal{R}_{PLL}(\hat{g}_{PLL}) - \hat{\mathcal{R}}_{PLL}(\hat{g}_{PLL}) \right) + \left( \hat{\mathcal{R}}_{PLL}(\hat{g}_{PLL}) - \hat{\mathcal{R}}_{PLL}(g^*_{PLL}) \right)
\]

\[
\leq \left( \mathcal{R}_{PLL}(\hat{g}_{PLL}) - \hat{\mathcal{R}}_{PLL}(\hat{g}_{PLL}) \right) + \left( \hat{\mathcal{R}}_{PLL}(\hat{g}_{PLL}) - \hat{\mathcal{R}}_{PLL}(g^*_{PLL}) \right)
\]

\[
\leq 2 \sup_{g \in \mathcal{G}} \left| \mathcal{R}_{PLL}(g) - \hat{\mathcal{R}}_{PLL}(g) \right|
\]

\[
\leq 2 \log \left( \frac{2}{\delta} \right) \frac{\sqrt{cL\ell} \sum_{y=1}^{c} \mathcal{R}_n(\mathcal{G}_y) + 2M \sqrt{\log \left( \frac{2}{\delta} \right) \frac{1}{2n}}}{1}. \]

D. Supplementary Theorem on Section 4

Theorem 3. The learning objective in Jin & Ghahramani (2003) is a special case of Eq. (8).

Proof. Recall the learning objective in Jin & Ghahramani (2003) is formulated as:

\[
\hat{\mathcal{R}}_{PLL} = \frac{1}{n} \sum_{i=1}^{n} \text{KL}[z_i || g(x_i)] = \frac{1}{n} \sum_{i=1}^{n} z_i \log \frac{z_i}{g(x_i)},
\]

(13)

where KL divergence is used and \( z_i \) represents the prior probability of \( x_i \).

Then in Eq. (8), the loss function can be specified as the cross-entropy loss: \( \ell_{CE}(g_j(x_i), e_s^y) = -e_s^y \log(g_j(x_i)) \), which is linear in the second term, i.e.,

\[
\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij} \ell_{CE}(g_j(x_i), e_s^y) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \ell_{CE}(g_j(x_i), w_{ij} e_s^y).
\]

Thus, the weights \( w \) can be moved into the loss function and yields:

\[
\hat{\mathcal{R}}_{PLL} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} \ell_{CE}(g_j(x_i), z_{ij}) = -\frac{1}{n} \sum_{i=1}^{n} z_i \log g(x_i),
\]

(14)

where \( z_{ij} = w_{ij} \mathbb{I}(j \in s_i) \). The optimal \( g^* \) of Eq. (14) is essentially equivalent to \( g^* \) learned from Eq. (13). Therefore, our method is a strict extension of (Jin & Ghahramani, 2003).

E. Benchmark Datasets

E.1. Setup

Table 3 describes the benchmark datasets and the corresponding models of them.
**Progressive Identification of True Labels for Partial-Label Learning**

**MNIST** This is a grayscale image dataset of handwritten digits from 0 to 9 where the size of the images is $28 \times 28$. The linear model is a linear-in-input model: $d$-10, and MLP refers to a 5-layer FC with ReLU as the activation function: $d$-300-300-300-300-10. Batch normalization (Ioffe & Szegedy, 2015) was applied before hidden layers. For both models, the softmax function was applied to the output layer, and $\ell_2$-regularization was added. The two models were trained by SGD with the default momentum parameter ($\beta = 0.9$), and the batch size was set to 256.

**Fashion-MNIST** This is a grayscale image dataset similarly to MNIST. In Fashion-MNIST, each instance is a $28 \times 28$ grayscale image and associated with a label from 10 fashion item classes. The models and optimizer were the same as MNIST.

**Kuzushiji-MNIST** This is another grayscale image dataset similarly to MNIST. In Kuzushiji-MNIST, each instance is a $28 \times 28$ grayscale image and associated with a label from 10 cursive Japanese (Kuzushiji) characters. The models and optimizer were the same as MNIST.

**CIFAR-10** This dataset consists of 60,000 $32 \times 32 \times 3$ colored image in RGB format in 10 classes.

The detailed architecture of ConvNet (Laine & Aila, 2017) is as follows.

- 0th (input) layer: $(32*32*3)$-
  - 1st to 4th layers: [C(3*3, 128)]*3-Max Pooling-
  - 5th to 8th layers: [C(3*3, 256)]*3-Max Pooling-
  - 9th to 11th layers: C(3*3, 512)-C(3*3, 256)-C(3*3, 128)-
  - 12th layers: Average Pooling-10

where C(3*3, 128) means 128 channels of 3*3 convolutions followed by Leaky-ReLU (LReLU) active function (Maas et al., 2013). $\cdot$ *3 means 3 such layers, etc.

The detailed architecture of ResNet (He et al., 2016) was as follows.

- 0th (input) layer: $(32*32*3)$-
  - 1st to 11th layers: C(3*3, 16)-[C(3*3, 16), C(3*3, 16)]*5-
  - 12th to 21st layers: [C(3*3, 32), C(3*3, 32)]*5-
  - 22nd to 31st layers: [C(3*3, 64), C(3*3, 64)]*5-
  - 32nd layer: Average Pooling-10

where [ · , · ] means a building block (He et al., 2016). These two models were trained by SGD with the default momentum parameter and the batch size was 256.

An example of a binomial flipping with $q = 0.1$ and of a pair flipping with $q = 0.5$ used on MNIST are below, respectively:

$$\begin{bmatrix}
1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1
0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1
0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1
0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1
0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1
0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1 & 0.1
0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1 & 0.1
0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1 & 0.1
0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1 & 0.1
0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 1
\end{bmatrix}$$

$$\begin{bmatrix}
1 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 1 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 1 & 0.5 & 0 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 1 & 0.5 & 0 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 1 & 0.5 & 0 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.5 & 0 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.5 & 0
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.5
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5
\end{bmatrix}$$

**E.2. Transductive Results**

Figure 3 illustrates the transductive results on the benchmark datasets, i.e., the ability in identifying the true labels in the training set. We can see that PRODEN has a strong ability to find the true labels.

**E.3. Test Results in the Pair Case**

Figure 4 illustrates the test results on the benchmark datasets in the pair case. They show a similar phenomenon to Figure 2 that PRODEN is affected slightly and CCN is affected severely when the ambiguity degree goes large.
Table 4 summaries the characteristic of the UCI datasets. We normalized these dataset by the Z-scores by convention and use the linear model trained by SGD with momentum 0.9.

F. UCI datasets

F.1. Characteristic of the UCI Datasets and Setup

Table 4 summaries the characteristic of the UCI datasets. We normalized these dataset by the Z-scores by convention and use the linear model trained by SGD with momentum 0.9.

F.2. Comparing Methods

The comparing PLL methods are listed as follows.

- SURE (Feng & An, 2019a): an iterative EM-based method [suggested configuration: $\lambda, \beta \in \mathbb{R}$]
Progressive Identification of True Labels for Partial-Label Learning

Figure 4. Test accuracy on MNIST, Fashion-MNIST, and CIFAR-10 in the pair case.

\[\{0.001, 0.01, 0.05, 0.1, 0.3, 0.5, 1\}\].

- **CLPL** (Cour et al., 2011): a parametric method that transforms the PLL problem to the binary learning problem [suggested configuration: SVM with squared hinge loss].

- **ECOC** (Zhang et al., 2017): a disambiguation-free method that adapts the binary decomposition strategy to PLL [suggested configuration: \(L = \log_2(l)\)].

- **PLSVM** (Nguyen & Caruana, 2008): a SVM-based method that differentiates candidate labels from non-candidate labels by maximizing the margin between them [suggested configuration: \(\lambda \in \{10^{-3}, \ldots, 10^3\}\)].

- **PLkNN** (Hullermeier & Beringer, 2006): a non-parametric approach that adapts k-nearest neighbors method to handle partially labeled data [suggested configuration: \(k \in \{5, 6, \ldots, 10\}\)].

- **IPAL** (Zhang & Yu, 2015): a non-parametric method that applies the label propagation strategy to iteratively update the weight of each candidate label [suggested configuration: \(\alpha = 0.95, k = 10, T = 100\)].

F.3. Results

Table 5 provides additional experiments to investigate the performances of each comparing methods on the UCI datasets with the pair flipping strategy. It shows that PRODEN generally achieves superior performance against other parametric comparing methods. Our advantage is a less obvious compared with the non-parametric method IPAL, whereas the performance of PRODEN could be easily increased by employing a deeper network.
Progressive Identification of True Labels for Partial-Label Learning

Table 4. Summary of UCI datasets and models.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Train</th>
<th># Test</th>
<th># Feature</th>
<th># Class</th>
<th>Model $g(x; \Theta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>1,335</td>
<td>149</td>
<td>8</td>
<td>10</td>
<td>Linear model</td>
</tr>
<tr>
<td>Texture</td>
<td>4,950</td>
<td>550</td>
<td>40</td>
<td>11</td>
<td>Linear model</td>
</tr>
<tr>
<td>Dermatology</td>
<td>329</td>
<td>37</td>
<td>34</td>
<td>6</td>
<td>Linear model</td>
</tr>
<tr>
<td>Synthetic-Control</td>
<td>540</td>
<td>60</td>
<td>60</td>
<td>6</td>
<td>Linear model</td>
</tr>
<tr>
<td>20Newsgroups</td>
<td>16,961</td>
<td>1,885</td>
<td>300</td>
<td>20</td>
<td>Linear model</td>
</tr>
</tbody>
</table>

Table 5. Test accuracy (mean±std) on the UCI datasets in the pair case.

<table>
<thead>
<tr>
<th>q</th>
<th>Yeast</th>
<th>Texture</th>
<th>Dermatology</th>
<th>Synthetic Control</th>
<th>20Newsgroups</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODEN</td>
<td>0.5 56.38±4.71%</td>
<td>99.71±0.12%</td>
<td>96.16±3.27%</td>
<td>98.05±1.58%</td>
<td>78.05±0.97%</td>
</tr>
<tr>
<td></td>
<td>0.9 44.03±4.12%</td>
<td>99.35±0.31%</td>
<td>93.68±6.80%</td>
<td>96.33±1.12%</td>
<td><strong>70.71±0.72%</strong></td>
</tr>
<tr>
<td>PRODEN-itera</td>
<td>0.5 56.26±4.74%</td>
<td>99.13±0.40%</td>
<td>93.15±2.56%</td>
<td>87.40±7.25%</td>
<td>76.90±0.92%</td>
</tr>
<tr>
<td></td>
<td>0.9 42.62±5.52%</td>
<td>78.74±4.09%</td>
<td>68.14±6.89%</td>
<td>57.90±5.08%</td>
<td>57.15±0.71%</td>
</tr>
<tr>
<td>GA</td>
<td>0.5 24.39±4.40%</td>
<td>94.25±0.48%</td>
<td>73.81±6.18%</td>
<td>64.68±3.08%</td>
<td>63.68±0.67%</td>
</tr>
<tr>
<td></td>
<td>0.9 16.50±3.43%</td>
<td>61.85±2.92%</td>
<td>48.03±11.42%</td>
<td>37.15±6.91%</td>
<td>45.86±0.95%</td>
</tr>
<tr>
<td>D^2CNN</td>
<td>0.5 56.38±4.71%</td>
<td>98.71±0.28%</td>
<td>95.89±3.75%</td>
<td>78.87±11.94%</td>
<td>74.38±0.90%</td>
</tr>
<tr>
<td></td>
<td>0.9 41.52±7.03%</td>
<td>86.45±4.87%</td>
<td>87.84±6.58%</td>
<td>62.92±12.36%</td>
<td>64.16±0.36%</td>
</tr>
<tr>
<td>SURE</td>
<td>0.5 51.69±3.81%</td>
<td>98.18±0.17%</td>
<td>95.71±2.49%</td>
<td>78.67±5.26%</td>
<td>70.21±0.88%</td>
</tr>
<tr>
<td></td>
<td>0.9 37.61±3.40%</td>
<td>98.00±0.42%</td>
<td>93.42±6.23%</td>
<td>52.33±6.49%</td>
<td>61.01±0.93%</td>
</tr>
<tr>
<td>CLPL</td>
<td>0.5 55.06±4.74%</td>
<td>98.80±0.22%</td>
<td>94.52±3.36%</td>
<td>75.83±4.29%</td>
<td>77.92±0.76%</td>
</tr>
<tr>
<td></td>
<td>0.9 44.66±4.69%</td>
<td>90.21±4.77%</td>
<td>87.67±3.06%</td>
<td>52.33±4.65%</td>
<td>65.03±0.32%</td>
</tr>
<tr>
<td>ECOC</td>
<td>0.5 54.55±3.92%</td>
<td>99.47±0.17%</td>
<td>94.71±2.29%</td>
<td>96.67±2.08%</td>
<td><strong>78.67±1.11%</strong></td>
</tr>
<tr>
<td></td>
<td>0.9 42.51±5.19%</td>
<td>69.69±4.82%</td>
<td>92.97±6.61%</td>
<td>94.50±1.32%</td>
<td>70.11±1.63%</td>
</tr>
<tr>
<td>PL SVM</td>
<td>0.5 52.59±2.04%</td>
<td>93.38±2.22%</td>
<td>93.97±4.50%</td>
<td>91.83±3.08%</td>
<td>76.21±2.30%</td>
</tr>
<tr>
<td></td>
<td>0.9 41.89±3.90%</td>
<td>82.24±6.58%</td>
<td>93.15±4.22%</td>
<td>80.67±9.42%</td>
<td>70.76±2.16%</td>
</tr>
<tr>
<td>PL rNN</td>
<td>0.5 53.60±2.81%</td>
<td>97.11±0.28%</td>
<td>94.52±3.06%</td>
<td>94.83±2.60%</td>
<td>43.77±0.72%</td>
</tr>
<tr>
<td></td>
<td>0.9 43.80±4.50%</td>
<td>92.98±0.64%</td>
<td>92.05±4.38%</td>
<td>89.83±4.54%</td>
<td>38.77±0.90%</td>
</tr>
<tr>
<td>IPAL</td>
<td>0.5 51.80±5.08%</td>
<td>99.30±0.37%</td>
<td>95.89±2.74%</td>
<td><strong>98.50±1.37%</strong></td>
<td>76.83±0.51%</td>
</tr>
<tr>
<td></td>
<td>0.9 36.60±2.58%</td>
<td>98.95±0.58%</td>
<td><strong>95.34±2.29%</strong></td>
<td><strong>98.50±1.09%</strong></td>
<td>69.15±0.73%</td>
</tr>
</tbody>
</table>

G. Characteristic of the Real-world Datasets and Setup

Table 6 summarizes the characteristic of the real-world datasets and the corresponding models. The preprocessing, model and optimizer were the same as UCI datasets.
Table 6. Summary of real-world partial label datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Examples</th>
<th># Feature</th>
<th># Class</th>
<th># Avg. CLs</th>
<th>Task Domain</th>
<th>Model $g(x; \Theta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lost</td>
<td>1122</td>
<td>108</td>
<td>16</td>
<td>2.23</td>
<td>automatic face naming (Panis &amp; Lanitis, 2014)</td>
<td>Linear model</td>
</tr>
<tr>
<td>BirdSong</td>
<td>4998</td>
<td>38</td>
<td>13</td>
<td>2.18</td>
<td>bird song classification (Briggs et al., 2012)</td>
<td>Linear model</td>
</tr>
<tr>
<td>MSRCv2</td>
<td>1758</td>
<td>48</td>
<td>23</td>
<td>3.16</td>
<td>object classification (Liu &amp; Dietterich, 2012)</td>
<td>Linear model</td>
</tr>
<tr>
<td>Soccer Player</td>
<td>17472</td>
<td>279</td>
<td>171</td>
<td>2.09</td>
<td>automatic face naming (Zeng et al., 2013)</td>
<td>Linear model</td>
</tr>
<tr>
<td>Yahoo! News</td>
<td>22991</td>
<td>163</td>
<td>219</td>
<td>1.91</td>
<td>automatic face naming (Guillaumin et al., 2010)</td>
<td>Linear model</td>
</tr>
</tbody>
</table>