Abstract

Partial label learning (PLL) is a typical weakly supervised learning problem, where each training example is associated with a set of candidate labels among which only one is true. Most existing PLL approaches assume that the incorrect labels in each training example are randomly picked as the candidate labels and model the generation process of the candidate labels in a simple way. However, these approaches usually do not perform as well as expected due to the fact that the generation process of the candidate labels is always instance-dependent. Therefore, it deserves to be modeled in a refined way. In this paper, we consider instance-dependent PLL and assume that the generation process of the candidate labels could decompose into two sequential parts, where the correct label emerges first in the mind of the annotator but then the incorrect labels related to the feature are also selected with the correct label as candidate labels due to uncertainty of labeling. Motivated by this consideration, we propose a novel PLL method that performs Maximum A Posteriori (MAP) based on an explicitly modeled generation process of candidate labels via decomposed probability distribution models. Experiments on benchmark and real-world datasets validate the effectiveness of the proposed method.

1 Introduction

Partial label learning (PLL) aims to deal with the problem where each instance is provided with a set of candidate labels, only one of which is the correct label. The problem of learning from partial label examples naturally arises in a number of real-world scenarios such as web data mining [21], multimedia content analysis [34, 2], ecoinformatics [19, 27], and etc.

A number of methods have been proposed to improve the practical performance of PLL. Identification-based PLL approaches [15, 24, 19, 3, 33] regard the ground-truth label as a latent variable and try to identify it. Average-based approaches [14, 8, 35] treat all the candidate labels equally and average the modeling outputs as the prediction. In addition, the risk-consistent method [8] and classifier-consistent methods [22] are proposed for deep model.

It is challenging to avoid overfitting on candidate labels, especially when the candidate labels depend on instances. Therefore, the previous methods assume that the candidate labels are instance-independent. Unfortunately, this often tends to be the case that the incorrect labels related to the feature are more likely to be picked as candidate label set for each instance. Recent work [31] has also shown that the presence of instance-dependent PLL imposes additional challenges but is more realistic in practice than the instance-independent case.

In this paper, we focus on the instance-dependent PLL via considering the essential generating process of candidate labels in PLL. To begin with, let us rethink meticulously how candidate labels arise in most manual annotation scenarios. When one annotates an instance, though the correct label has already emerged in the mind of the annotator first, the incorrect labels which are related to the feature of the instance confuse the annotator, then leading to the result that the correct label and some incorrect labels are packed together as the candidate labels. Therefore, the generating process of the candidate labels in instance-dependent PLL could be decomposed into two stages, i.e., the generation of the correct label of the instance and the generation of the incorrect labels related to the instance, which could be described by Categorical distribution and Bernoulli distribution, respectively.

Motivated by the above consideration, we propose a novel PLL method named IDGP, i.e., Instance-dependent partial label learning via Decomposed Generation Process. Before performing IDGP, the
distribution of the correct label and the incorrect label in the candidate label set of each training example should be modeled explicitly by decoupled probability distributions Categorical distribution and Bernoulli distribution. Then we perform Maximum A Posteriori (MAP) estimation on the PLL training dataset to deduce a risk minimizer. To optimize the risk minimizer, Dirichlet distribution and Beta distribution are leveraged to model the condition prior in it and estimate the parameters of Categorical distribution and Bernoulli distribution due to the conjugacy. Finally, we refine prior information by updating the parameters of the corresponding conjugate distributions iteratively to improve the performance of the predictive model in each epoch. Our contributions can be summarized as follows:

• We for the first time explicitly model the generation process of candidate labels in instance-dependent PLL. The entire generating process is decomposed into the generation of the correct label of the instance and the generation of the incorrect labels, which could be described by Categorical distribution and Bernoulli distribution, respectively.

• We optimize the models of Categorical distribution and Bernoulli distribution via MAP technique, where the corresponding conjugate distributions, i.e., Dirichlet distribution and Beta distribution are induced.

• We derive an estimation error bound of our approach, which demonstrates that the empirical risk minimizer would approximately converge to the optimal risk minimizer as the number of training data grows to infinity.

The rest of this paper is organized as follows. Section 2 briefly reviews related work in recent years. Section 3 presents the depiction of the generation model of candidate labels we proposed and the technical details of our approach IDGP. Section 4 reports the derivation of the estimation error bound. Section 5 reports and analyzes the results of comparative experiments on the corrupted benchmark datasets as well as real-world datasets. Finally, Section 6 concludes the whole paper.

2 Related Work

Recent years have witnessed a number of meaningful and effective works in the area of PLL. Each instance in the PLL training dataset is labeled with candidate labels, in which the correct label hides itself among some incorrect labels. Because it is hard to utilize supervision information in implicit labeling information, PLL is considered as a weak supervision learning framework [37, 25, 16, 20, 9]. An intuitive strategy to mine supervision information in candidate labels is disambiguation, which means distinguishing the correct label from the incorrect labels. In this way, PLL training examples can be learned by many traditional strong supervision methods once the correct label is identified. Disambiguation can be divided into two basic strategies. One is based on identification via regarding the ground-truth label as a latent variable to recognize it [15, 24, 19, 3, 33]. Another is average-based, treating all the labels equally and averaging their respective model outputs as the final prediction [14, 5, 35].

At the beginning, many algorithms draw on experience from traditional classical machine-learning techniques to form PLL algorithms. And this directly leads to that they may not work or consume unbearable time as the number and dimensionality of training examples increase. [15, 19] attempt to maximize the likelihood of observing the candidate labels of each training example instead of the correct label. [14, 35] are inspired by the K-nearest neighbor technique and voting for each example only among the candidate labels of its neighbors. [24, 33] adopt maximum margin techniques to expand the classification margin over the partially labeled training examples by discriminating the outputs of multi-class SVM [28] on candidate labels and non-candidate labels. Ensemble learning is also reflected in PLL. For the bagging technique, [36] decomposes the multi-class classifier to some specific binary classifiers and average their outputs with the corresponding weight as the final prediction, which is free of any disambiguation operation towards the candidate labels. For boosting techniques, in each boosting round, the weight of each training example and the ground-truth confidence of its corresponding candidate labels are maintained and updated to train a base classifier.
Besides, [7] introduces self-training for PLL and performs pseudo-labeling in the PLL training dataset.

Then from [32], the deep neural network (DNN) with stochastic optimizers [26, 6] is first applied to PLL but the method restricts the networks to some specific architectures. On the heels of it, [22] first derives a classifier-consistent risk estimator and proposes a novel algorithm that is compatible with arbitrary multi-class classifier and stochastic optimizers. And since then, PLL has been able to enjoy the leading-edge models and optimizers from the deep learning community [10]. [8] considers the generation process of candidate labels and then deduces a risk-consistent method and a classifier-consistent method.

Most previous methods do not pay attention to the generation process of candidate labels or just simply assume that the candidate labels are instance-independent. For example, the algorithms along the traditional machine-learning line have little consideration about the generation process of candidate labels. [8] only assumes that the candidate label set comes from a random sampling process. However, as introduced in Section 1, the generation process is entirely instance-dependent as [31] has shown, where the incorrect labels related to the feature are more likely to be picked as the candidate labels. It is not similar to the assumption made by [8] that a labeling system uniformly samples a label set for each instance and asks the annotator whether the correct label is in the sampled label set. In this paper, we consider the instance-dependent PLL and explicitly model the generation process of instance-dependent candidate labels via decoupled probability distributions.

3 Proposed Method

In this section, we first introduce the explicit generation model of instance-dependent candidate labels, which decouples the distribution of the correct and candidate incorrect labels. Then we leverage Category distribution and Bernoulli distribution to depict them respectively with Dirichlet distribution dependent on the instance. The concrete generation model is demonstrated as follows.

First of all, we introduce the necessary notations for our approach. In PLL, the labeling information of every instance in the training dataset is corrupted from one correct label into candidate label set, which contains the correct label and candidate incorrect labels but is not a whole label set. Let $X \subseteq \mathbb{R}^q$ be the $q$-dimensional feature space of the instances and $Y = \{1, 2, \ldots, c\}$ be the label space with $c$ class labels. Then a PLL training data set can be formulated as $D = \{(x_i, S_i) | 1 \leq i \leq n\}$, where $x_i \in X$ denotes the $i$-th $q$-dimensional feature vector and $S_i \in C (C = \{C|2^Y \setminus \emptyset\})$ is the candidate label set annotated for $x_i$. $S_i$ consists of the correct label $y_i \in Y$ and the candidate incorrect label set $\overline{S_i}^y = S_i \setminus \{y_i\}$. For the convenience of description, we use a logical label vector $l_i = [l_1^i, l_2^i, \ldots, l_c^i] \in \{0, 1\}^c$ to represents whether the $j$-th label is the correct label $y_i$, i.e., $l_j^i = 1$ if $j = y_i$, otherwise $l_j^i = 0$. For the candidate incorrect label set $\overline{S_i}^y$, it can also be denoted by a logical label vector $\overline{l}_i^y = [\overline{l}_1^y, \overline{l}_2^y, \ldots, \overline{l}_c^y] \in \{0, 1\}^c$ to represent whether the $j$-th label is the candidate incorrect label, i.e., $\overline{l}_j^y = 1$ if $j \in \overline{S_i}^y$, $\overline{l}_j^y = 0$ if $j \notin \overline{S_i}^y$. In the task of PLL we aim to find a multi-class classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ according to $D$.

3.1 Generation Model of Candidate Labels

Focusing on instance-dependent partial label learning, we propose a novel generation model of candidate labels, which explicitly models the generation process and decouples the distribution of the candidate labels into the distribution of the correct label and candidate incorrect labels respectively. To form a candidate label set, we suppose the correct label emerges and is selected first according to a posterior distribution about the instance first. And then the candidate incorrect labels related to the feature of instance emerge to disturb the annotator and are sampled from another posterior distribution dependent on the instance. The concrete generation model is demonstrated as follows.

Given an instance $x_i$, its candidate label set $S_i$ is drawn from a probability distribution with the
following density:

\[ p(S_i|x_i) = \sum_{j \in S_i} p(y_i = j|x_i)p(S_i^j|x_i). \]  

(1)

Here, \( p(S_i|x_i) \) suggests our generation model is entirely dependent on the instance. For the given candidate label set \( S_i \), each label \( j \in S_i \) have the possibility \( p(y_i = j|x_i) \) sampled by the annotator as the correct label in the first stage of generation and the candidate incorrect labels \( S_i^j \) are then sampled with the possibility \( p(S_i^j|x_i) \) in the second stage to form the entire candidate label set \( S_i \). In this way, we decouple the generation process of the instance-dependent candidate labels.

Upon decomposition, we need the corresponding probability distribution to depict \( p(y_i|x_i) \) and \( p(S_i^j|x_i) \). We assume that the correct label \( y_i \) of each instance \( x_i \) is drawn from a Categorical distribution with the parameters \( \theta_i \), where \( \theta_i = [\theta_i^1, \theta_i^2, \ldots, \theta_i^c] \in [0, 1]^c \) is a \( c \)-dimension vector with the constraint \( \sum_{j=1}^c \theta_i^j = 1 \), i.e.,

\[ p(y_i|x_i, \theta_i) = \text{Cat}(y_i|\theta_i) = \prod_{j=1}^c (\theta_i^j)^{y_i^j}. \]  

(2)

In order to describe \( p(S_i^j|x_i, \theta_i) \), we further decouple the candidate incorrect labels by assuming that the distribution of each \( S_i^j \) is independent from each other, i.e.,

\[ p(S_i^j|x_i) = \prod_{j \in S_i} p(S_i^j|x_i) \prod_{j \notin S_i} (1- p(S_i^j|x_i)) \]

and the candidate incorrect label set \( S_i^j \) is drawn from a multinomial distribution with the parameters \( z_i \), where \( z_i = [z_i^1, z_i^2, \ldots, z_i^c] \in [0, 1]^c \) is a \( c \)-dimension vector, i.e.,

\[ p(S_i^j|x_i, z_i) = \prod_{j \in S_i} \text{Ber}(z_i^j|\theta_i^j) = \prod_{j \notin S_i} (z_i^j)^{\theta_i^j} (1 - z_i^j)^{1-\theta_i^j}. \]  

(3)

For the convenient estimation of \( \theta_i \), we introduce Dirichlet distribution, as its conditional prior, i.e., \( p(\theta_i|x_i) = \text{Dir}(\theta_i|\lambda_i) \), where \( \lambda_i = [\lambda_i^1, \lambda_i^2, \ldots, \lambda_i^c]^\top (\lambda_i^0 > 0) \) is a \( c \)-dimension vector and an output of our main branch inference model \( f \) parameterized by \( \Theta \) for an instance \( x_i \), i.e. \( \lambda_i = a \cdot \exp(f(x_i; \Theta)/\gamma) + b \), where \( a \), \( b \) and \( \gamma(a > 0, b > 0, \gamma > 0) \) are used to resolve the scale ambiguity. Note that the branch inference model \( f \) is the final predictive model for the task of PLL to accomplish the target \( X \mapsto Y \) by using \( \hat{y}_i = \arg \max_j \theta_i^j \).

Likewise, to estimate \( z_i \), we introduce Beta distribution, as the conditional prior parameterized by \( \alpha = [\alpha_i^1, \alpha_i^2, \ldots, \alpha_i^c]^\top \) and \( \beta = [\beta_i^1, \beta_i^2, \ldots, \beta_i^c]^\top (\alpha_i^0 > 0, \beta_i^0 > 0) \), i.e., \( p(z_i|x_i) = \prod_{j \in S_i} \text{Beta}(z_i^j|\alpha_i^j, \beta_i^j) \). We use an auxiliary branch model \( g \) parameterized by \( \Omega \) to output \( \lambda_i^j = [\alpha_i^j, \beta_i^j] \) at the same time, i.e., \( \lambda_i = a \cdot \exp(g(x_i; \Omega)/\gamma) + b \). We use the same constants scaling \( \lambda_i \) to simplify the complexity of our approaches.

It should be noted that the predictive model \( f(x; \Theta) \) or auxiliary model \( g(x; \Omega) \) can be any deep neural network once its output satisfies the corresponding constraint.

### 3.2 Optimization Using Maximum A Posterior

Based on the generation model of candidate labels denoted by Eq. (1), Eq. (2) and Eq. (3), by using the technique of Maximum Likelihood (ML) estimation we can immediately induce the log-likelihood loss function for the PLL training data set to be optimized:

\[ L_{ML} = - \sum_{i=1}^{n} \log p(S_i|x_i, \theta_i, z_i) = - \sum_{i=1}^{n} \log \sum_{j \in S_i} \theta_i^j \prod_{k \in S_i^j} z_i^k \prod_{k \notin S_i^j} (1 - z_i^k). \]  

(4)

The Eq. (4) clearly demonstrates how the distribution of the correct label interacts with that of the candidate incorrect labels after decoupling the generation. In the backpropagation of the training process, they provide weight coefficients for each other.
In order to bring in prior information for PLL, we further introduce IDGP, which performs MAP in the training data set. In IDGP, we are more concerned about maximizing the joint distribution \( p(\theta, z|x, S) \). This will lead to the following optimization problem:

\[
\mathcal{L}_{MAP} = \mathcal{L}_{ML} + \mathcal{L}_{reg},
\]

where

\[
\mathcal{L}_{reg} = -\sum_{i=1}^{n} \log p(\theta_i|x_i) + \log p(z_i|x_i).
\]

Compared to ML, our IDGP framework provides a natural way of leveraging prior information via optimizing the extra condition prior illustrated by the Eq. (6) in the training process, which is significant for PLL due to the implicit supervision information. \( \mathcal{L}_{reg} \) can be analytically calculated as follows:

\[
\mathcal{L}_{reg} = -\sum_{i=1}^{n} \sum_{j=1}^{c} (\lambda_{ji} - 1) \log \theta_{ji} + (\alpha_{ji} - 1) \log z_{ji} + (\beta_{ji} - 1) \log (1 - z_{ji}).
\]

As it is shown in the Eq. (7), IDGP transforms prior information into appropriate weights exerted on the corresponding label.

Due to the conjugacy of Dirichlet and Categorical distribution, we can estimate \( \hat{\theta}_{ji} \) by using

\[
\hat{\theta}_{ji} = \mathbb{E}[\theta_{ji}|x_i, \lambda_i] = \frac{\alpha_{ji} + \lambda_{ji}}{\sum_{k=1}^{c} \lambda_{ki} + \alpha_{ji}},
\]

where \( \alpha_{ji} \) denotes the number of occurrences of label \( j \) for \( x_i \), i.e., \( \alpha_{ji} = 1 \) if \( j \in S_i \), otherwise \( \alpha_{ji} = 0 \). Similarly, we can leverage the conjugacy of Beta and Bernoulli distribution to estimate \( \hat{z}_{ji} \), i.e.,

\[
\hat{z}_{ji} = \mathbb{E}[z_{ji}|x_i, \alpha_{ji}, \beta_{ji}] = \frac{\alpha_{ji} + \lambda_{ji}}{\alpha_{ji} + \beta_{ji} + \lambda_{ji}}.
\]

In IDGP, the induction of prior information reflects in estimating parameters of Dirichlet and Beta. We use \( \lambda_i \) at the \( t \) epoch, which is denoted by \( \hat{\lambda}_{ji}^{(t)} \), to estimate \( \lambda_{ji}^{(t+1)} \) at the \( t + 1 \) epoch. And at the same time because the non-candidate labels can never be sampled to be the correct label, we set the value of each non-candidate label to a minor value \( \epsilon \) like \( 10^{-3} \). Specifically, the estimation vector \( \hat{\lambda}_{ji}^{(t+1)} \) will be calculated by

\[
\hat{\lambda}_{ji}^{(t+1)} = \begin{cases} 
\lambda_{ji}^{(t)} + \hat{\lambda}_{ji}^{(t)} / \sum_{k \in S_i} \lambda_{ki}^{(t)}, & \text{if } j \in S_i \\
\epsilon, & \text{otherwise.}
\end{cases}
\]

In the similar way we estimate \( \hat{\alpha}_{ji}^{(t+1)} \) and \( \hat{\beta}_{ji}^{(t+1)} \) with the assistance of \( \hat{\alpha}_{ji}^{(t)} \) and \( \hat{\beta}_{ji}^{(t)} \), i.e., the estimation vector \( \hat{\alpha}_{ji}^{(t+1)} \) and \( \hat{\beta}_{ji}^{(t+1)} \) will be calculated by

\[
\begin{align*}
\hat{\alpha}_{ji}^{(t+1)} &= \hat{\alpha}_{ji}^{(t)} + \alpha_{ji}^{(t)} / (\alpha_{ji}^{(t)} + \beta_{ji}^{(t)}) \\
\hat{\beta}_{ji}^{(t+1)} &= \hat{\beta}_{ji}^{(t)} + \beta_{ji}^{(t)} / (\alpha_{ji}^{(t)} + \beta_{ji}^{(t)}).
\end{align*}
\]

Here, the initial values of the parameters in Dirichlet are assigned by \( \hat{\lambda}_{ji}^{(1)} = 1 \) if \( j \in S_i \), otherwise \( \epsilon \). And the initial values of the parameters in Beta are assigned by \( \hat{\alpha}_{ji}^{(1)} = 1 \) and \( \hat{\beta}_{ji}^{(1)} = 1 \). Note that at \( t \) epoch \( \hat{\lambda}_{ji}, \hat{\alpha}_{ji}, \hat{\beta}_{ji} \) are all constants and do not perform backpropagation of their gradients while \( \theta^{(t)} \) and \( z^{(t)} \) back propagate their gradients normally.

In the above way, we refine the prior information epoch by epoch. The prior information comes from the memorization effects of the neural network. It makes the neural network always likely to recognize and remember the correct label in priority, leading to a kind of initial disambiguation at
According to the Eq.(5), the empirical risk estimator for the predictive model
\[ R(f) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i), S_i), \]

where
\[ \mathcal{L}(f(x_i), S_i) = -\log \sum_{j \in S_i} \theta_i^j (1 - z_i^j) \prod_{k \in S_i} z_i^k - \sum_{j=1}^c (\lambda_i^j - 1) \log \theta_i^j. \]

Then we define a function space as:
\[ \mathcal{G} = \{(x, S) \mapsto \mathcal{L}(f(x), S) | f \in \mathcal{F}\}. \]

For further analysis, we give an upper bound of \( \mathcal{L}(f(x_i), S_i) \) as follows:
\[ \mathcal{L}(f(x_i), S_i) \leq -\left( K_i + \sum_{j=1}^c u_i^j \ell(f(x_i), e^j) \right), \]

where \( e^j \) denotes the standard canonical vector in \( \mathbb{R}^c \), \( u_i^j = \lambda_i^j - 1 \) if \( j \in S_i \), otherwise \( u_i^j = \lambda_i^j - 1 \), and \( K_i = \log |S_i| + \frac{1}{|S_i|} \sum_{j \in S_i} \log(1 - z_i^j) \prod_{k \in S_i} z_i^k \). We denote \( K = \max\{K_1, K_2, ..., K_n\} \) and scale \( u_i^j \) to \([0, \rho]\) during the training process. The detailed induction of Eq.(15) can be seen in Appendix A.1.

**Definition 1.** Let \( x_1, x_2, ..., x_n \in \mathcal{X} \) be \( n \) i.i.d. random variables drawn from a probability distribution and \( \sigma_1, \sigma_2, ..., \sigma_n \in \{-1, +1\} \) be Rademacher variables with even probabilities, \( \mathcal{H} = \{h : \mathcal{X} \to \mathbb{R}\} \) be a class of measurable functions. Then the expected Rademacher complexity of \( \mathcal{H} \) is defined as
\[ \mathcal{R}_n(\mathcal{H}) = \mathbb{E}_{x, \sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i) \right]. \]
Then the expected Rademacher complexity of $G$ can be defined as follows:

$$
\tilde{R}_n(G) = \mathbb{E}_{x,S,\sigma} \left[ \sup_{g \in G} \frac{1}{n} \sum_{i=1}^{n} \sigma_i g(x_i, S_i) \right].
$$

(16)

**Lemma 1.** Suppose the loss function $L(f(x), S)$ is bounded by $M$, i.e., $M = \sup_{x \in X, S \in C, f \in F} L(f(x), S)$. Then for any $\xi > 0$, with probability at least $1 - \xi$,

$$
\sup_{f \in F} \left| R(f) - \hat{R}(f) \right| \leq 2\tilde{R}_n(G) + \frac{M}{2} \sqrt{\frac{\log \frac{2}{\xi}}{2n}}.
$$

The proof of Lemma 1 is provided in Appendix A.2.

**Lemma 2.** Assume the loss function $\ell(f(x), e^\imath)$ is $L$-Lipschitz with respect to $f(x)$ (0 < $L$ < $\infty$) for all $\imath \in \mathcal{Y}$. Then, the following inequality holds:

$$
\tilde{R}_n(G) \leq 2\rho_cL \sum_{\imath \in \mathcal{Y}} \tilde{R}_n(\mathcal{H}_\imath) + K,
$$

where

$$
\mathcal{H}_\imath = \{ h : x \mapsto f_\imath(x) | f \in F \},
$$

$$
\tilde{R}_n(\mathcal{H}_\imath) = \mathbb{E}_{x,\sigma} \left[ \sup_{h \in \mathcal{H}_\imath} \frac{1}{n} \sum_{i=1}^{n} h(x_i) \right].
$$

The proof of Lemma 1 is provided in Appendix A.3.

Based on Lemma 1 and Lemma 2, we induce an estimation error bound for our IDGP method. Let $\hat{f} = \arg \min_{f \in F} \hat{R}(f)$ be the empirical risk minimizer and $f^\star = \arg \min_{f \in F} R(f)$ be the true minimizer. The function space $\mathcal{H}_\imath$ for the label $\imath \in \mathcal{Y}$ is defined as $\{ h : x \mapsto f_\imath(x) | f \in F \}$. Let $\tilde{R}_n(\mathcal{H}_\imath)$ be the expected Rademacher complexity of $\mathcal{H}_\imath$ with sample size $n$, then we have the following theorem.

**Theorem 1.** Assume the loss function $\ell(f(x), e^\imath)$ is $L$-Lipschitz with respect to $f(x)$ (0 < $L$ < $\infty$) for all $\imath \in \mathcal{Y}$ and $L(f(x), S)$ is bounded by $M$, i.e., $M = \sup_{x \in X, S \in C, f \in F} L(f(x), S)$. Then, for any $\xi > 0$, with probability at least $1 - \xi$,

$$
R(\hat{f}) - R(f^\star) \leq 4\sqrt{2\rho_cL} \sum_{\imath \in \mathcal{Y}} \tilde{R}_n(\mathcal{H}_\imath) + M \sqrt{\frac{\log \frac{2}{\xi}}{2n}} + 4K.
$$

The proof is provided in Appendix A.4. Theorem 1 means that when $n \to \infty$ the empirical risk minimizer $\hat{f}$ will converge to the optimal risk minimizer $f^\star$ as to all parametric models with a bounded norm.

**5 Experiments**

In this section, we validate the effectiveness of the IDGP by performing it on benchmark datasets and real-world datasets and comparing its results against five DNN-based algorithms and five classical PLL methods. Furthermore, ablation study and sensitive analysis of parameters are also conducted to explore the IDGP.
We implement five DNN-based algorithms on five widely used benchmark datasets in deep learning, including MNIST [18], Kuzushiji-MNIST [4], Fashion-MNIST [29], CIFAR-10 and CIFAR-100 [17]. For the reason that each example is annotated by the correct label instead of a candidate label set, we take the same strategy as [31], which for the first time consider instance-dependent PLL, to corrupt the five benchmark datasets into PLL datasets.

Besides, five frequently used real-world datasets, which comes from different application domain, are employed to further validate the effectiveness and generation ability of our algorithm in practical scenarios, including Lost [5], MSRCv2 [19], BirdSong [1], Soccer Player [34] and Yahoo!News [11]. We implement six DNN-based algorithms and five classical PLL methods on these real-world datasets.

In the implementation, we run five trials on each benchmark dataset (with 20% split from the training dataset for validation) and perform five-fold cross-validation on real-world PLL datasets. The mean accuracy and standard deviation of all comparing algorithms are kept in record.

### 5.1 Datasets

| Dataset    | MNIST        | Kuzushiji-MNIST | Fashion-MNIST | CIFAR-10 | CIFAR-100 |
|------------|--------------|----------------|---------------|----------|-----------|
| IDGP       | 97.85 ± 0.05% | 86.32 ± 0.16% | 86.49 ± 0.14% | 82.31 ± 0.19% | 50.38 ± 0.27% |
| CALLEN     | 97.83 ± 0.07% | 86.12 ± 0.22% | 86.39 ± 0.05% | 82.02 ± 0.15% | 46.30 ± 0.29% |
| PRODEN     | 97.87 ± 0.08% | 86.09 ± 0.28% | 86.38 ± 0.25% | 81.32 ± 0.21% | 49.99 ± 0.34% |
| RC         | 97.73 ± 0.04% | 85.37 ± 0.33% | 86.32 ± 0.20% | 80.66 ± 0.25% | 48.39 ± 0.28% |
| CC         | 97.50 ± 0.28% | 83.06 ± 0.12% | 85.70 ± 0.14% | 76.57 ± 0.05% | 43.94 ± 0.90% |
| D2CNN      | 96.49 ± 0.10% | 79.04 ± 0.63% | 83.36 ± 0.72% | 51.55 ± 0.48% | 16.21 ± 0.16% |

Table 1: Classification accuracy (mean ± std) of each comparing approach on benchmark datasets for instance-dependent PLL.

| Dataset    | Lost | MSRCv2 | BirdSong | Soccer Player | Yahoo!News |
|------------|------|--------|----------|---------------|------------|
| IDGP       | 77.54 ± 3.82% | 54.54 ± 1.71% | 72.45 ± 0.23% | 55.99 ± 0.28% | 67.62 ± 0.12% |
| CALLEN     | 72.19 ± 2.64% | 43.75 ± 2.64% | 71.24 ± 0.45% | 54.49 ± 0.79% | 67.41 ± 0.11% |
| PRODEN     | 76.02 ± 3.96% | 43.91 ± 0.64% | 71.51 ± 0.77% | 55.55 ± 1.31% | 67.15 ± 0.13% |
| RC         | 76.54 ± 3.91% | 44.41 ± 0.73% | 71.56 ± 0.88% | 54.23 ± 0.89% | 67.04 ± 0.32% |
| CC         | 75.22 ± 3.12% | 45.73 ± 1.84% | 71.13 ± 2.23% | 54.08 ± 0.93% | 67.23 ± 0.71% |
| D2CNN      | 69.59 ± 0.01% | 40.85 ± 2.65% | 66.71 ± 1.23% | 48.63 ± 0.54% | 56.76 ± 1.01% |
| CLPL       | 74.25 ± 2.34% | 44.16 ± 2.43% | 65.88 ± 1.21% | 50.23 ± 1.13% | 53.41 ± 1.32% |
| PL-SVM     | 71.68 ± 2.82% | 39.34 ± 4.01% | 50.78 ± 1.44% | 38.47 ± 1.26% | 51.47 ± 1.22% |
| PL-KNN     | 32.17 ± 2.59% | 44.18 ± 2.65% | 64.78 ± 0.71% | 49.51 ± 1.33% | 40.78 ± 0.41% |
| IPAL       | 72.22 ± 3.32% | 52.69 ± 1.66% | 70.33 ± 0.11% | 54.53 ± 0.38% | 66.17 ± 0.75% |
| PLL        | 72.57 ± 3.85% | 47.66 ± 0.76% | 70.82 ± 0.28% | 53.44 ± 0.97% | 59.83 ± 0.56% |

Table 2: Classification accuracy (mean ± std) of comparing algorithms on the real-world datasets.

### 5.2 Baselines

We compare IDGP with five DNN-based methods:

- **VALEN** [31]. An instance-dependent PLL framework that performs label enhancement during the training process and iteratively recovers the latent label distributions.

- **RC** [8]. A risk-consistent PLL method induced by an importance reweighting strategy and using the softmax output of the model to approximate the posterior.

- **CC** [8]. A classifier-consistent PLL method taking advantage of the transition matrix that describes the probability of the candidate label set given an ordinary label.
Table 3: Classification accuracy (mean±std) for comparison against IDGP-ML.

| Dataset       | IDGP   | IDGP-ML |
|---------------|--------|---------|
| MNIST         | 97.85±0.05% | 91.79±0.12% |
| Kuzushiji-MNIST | 86.32±0.17% | 76.67±0.34% |
| Fashion-MNIST | 86.49±0.12% | 78.38±0.29% |
| CIFAR-10      | 82.31±0.29% | 69.47±0.45% |
| CIFAR-100     | 50.12±0.24% | 40.23±0.57% |

5.3 Experimental Results

The performance of each DNN-based method on corrupted benchmark dataset is summarized in Table 1 where the best results are highlighted in bold and ○/○ indicates whether IDGP statically win/lose to the comparing method on each dataset additionally (pairwise t-test at 0.05 significance level). From the table, we can overall see that IDGP outperform each DNN-based methods in Kuzushiji-MNIST, Fashion-MNIST, CIFAR-10 and CIFAR-100 and only lose to PRODEN in MNIST. Simultaneously, we can learn from Figure that IDGP is enough stable during the training process compared to other DNN-based algorithms.

Table 2 demonstrates the ability of IDGP to solve the PLL problem in realworld datasets. Our method has stronger competence than others to generalize in most real-world datasets except MSRCv2, where classical PLL methods such as IPAL and PLL are more effective.
5.4 Further Analysis

To demonstrate the effectiveness of the iteratively refined prior information introduced by IDGP, we remove the loss function $L_{reg}$ to reverse the IDGP to IDGP-ML, which only uses the log-likelihood function for optimization. The performance of IDGP-ML against IDGP is also measured by the classification accuracy (with pairwise t-test at 0.05 significance level). As is illustrated in Table 3, IDGP achieves superior performance on all benchmark datasets compared to IDGP-ML with the assistance of the prior information which IDGP provides and improves epoch by epoch.

Furthermore, we study the influence of the two hyper-parameters $a$, $\gamma$ to our algorithm, which decides the scale of Dirichlet and Beta distribution parameters. Figure illustrates the sensitivity of IDGP in MNIST, Fashion-MNIST, Lost, Yahoo!News when $a$ varies from 0.001 to 1000 and $\gamma$ varies from 0.1 to 3. We can easily find that as for benchmark datasets like MNIST and Fashion-MNIST the performance of IDGP is stable and effective in the case that $a$ is around 10 and $\gamma$ is around 1.5. For the small-scale real-world datasets like Lost, $a$ and $\gamma$ are recommended to be around 1 and 0.5, respectively. For the large-scale real-world datasets like Yahoo!News, $a$ and $T$ are suggested to be around 0.001 and 0.5, respectively.

6 Conclusion

In this paper, we for the first time explicitly decompose and model the generation process of instance-dependent candidate labels, which is more realistic. Then based on the decomposition generation process, a novel PLL approach IDGP is proposed by us to further introduce and refine the prior information in every training epoch via MAP.

References

[1] Forrest Briggs, Xiaoli Z Fern, and Raviv Raich. Rank-loss support instance machines for miml instance annotation. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 534–542, 2012.

[2] Ching-Hui Chen, Vishal M Patel, and Rama Chellappa. Learning from ambiguously labeled face images. IEEE transactions on pattern analysis and machine intelligence, 40(7):1653–1667, 2017.

[3] Yi-Chen Chen, Vishal M Patel, Rama Chellappa, and P Jonathon Phillips. Ambiguously labeled learning using dictionaries. IEEE Transactions on Information Forensics and Security, 9(12):2076–2088, 2014.

[4] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. arXiv preprint arXiv:1812.01718, 2018.

[5] Timothee Cour, Ben Sapp, and Ben Taskar. Learning from partial labels. The Journal of Machine Learning Research, 12:1501–1536, 2011.

[6] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research, 12(7), 2011.

[7] Lei Feng and Bo An. Partial label learning with self-guided retraining. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 3542–3549, 2019.

[8] Lei Feng, Jiaqi Lv, Bo Han, Miao Xu, Gang Niu, Xin Geng, Bo An, and Masashi Sugiyama. Provably consistent partial-label learning. arXiv preprint arXiv:2007.08929, 2020.

[9] Chen Gong, Hong Shi, Tongliang Liu, Chuang Zhang, Jian Yang, and Dacheng Tao. Loss decomposition and centroid estimation for positive and unlabeled learning. IEEE transactions on pattern analysis and machine intelligence, 43(3):918–932, 2019.
[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[11] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multiple instance metric learning from automatically labeled bags of faces. In *European conference on computer vision*, pages 634–647. Springer, 2010.

[12] B Han, Q Yao, X Yu, G Niu, M Xu, W Hu, I Tsang, and M Sugiyama. Robust training of deep neural networks with extremely noisy labels. In *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS)*, volume 2, page 4, 2020.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Eyke Hüllermeier and Jürgen Beringer. Learning from ambiguously labeled examples. *Intelligent Data Analysis*, 10(5):419–439, 2006.

[15] Rong Jin and Zoubin Ghahramani. Learning with multiple labels. In *NIPS*, volume 2, pages 897–904. Citeseer, 2002.

[16] Konstantinos Kamnitsas, Daniel Castro, Loic Le Folgoc, Ian Walker, Ryutarou Tanno, Daniel Rueckert, Ben Glocker, Antonio Criminisi, and Aditya Nori. Semi-supervised learning via compact latent space clustering. In *International Conference on Machine Learning*, pages 2459–2468. PMLR, 2018.

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Liping Liu and Thomas G Dietterich. A conditional multinomial mixture model for superset label learning. In *Advances in neural information processing systems*, pages 548–556. Citeseer, 2012.

[20] Nan Lu, Gang Niu, Aditya Krishna Menon, and Masashi Sugiyama. On the minimal supervision for training any binary classifier from only unlabeled data. *arXiv preprint arXiv:1808.10585*, 2018.

[21] Jie Luo and Francesco Orabona. Learning from candidate labeling sets. Technical report, MIT Press, 2010.

[22] Jiaqi Lv, Miao Xu, Lei Feng, Gang Niu, Xin Geng, and Masashi Sugiyama. Progressive identification of true labels for partial-label learning. In *International Conference on Machine Learning*, pages 6500–6510. PMLR, 2020.

[23] Thomas Minka. Estimating a dirichlet distribution. 2000.

[24] Nam Nguyen and Rich Caruana. Classification with partial labels. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–559, 2008.

[25] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1944–1952, 2017.

[26] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.
A Appendix

A.1 Calculation Details of Eq.(15)

According to multivariate basic inequality, we can obtain:

\[-\log \sum_{j \in S_i} \theta_i^j (1 - z_i^j) \prod_{k \in S_i^j} z_i^k \leq -\log |S_i| - \frac{1}{|S_i|} \sum_{j \in S_i} \left( \log \theta_i^j + \log(1 - z_i^j) \prod_{k \in S_i^j} z_i^k \right). \] (17)


Then we can calculate Eq. (15) as

$$L(f(x_i), S_i) = -\log \sum_{j \in S_i} \theta_i^j (1 - z_i^j) \prod_{k \in S_i^j} z_k^j - \sum_{j=1}^c (\lambda_i^j - 1) \log \theta_i^j$$

$$\leq -\log |S_i| - \frac{1}{|S_i|} \sum_{j \in S_i} \left( \log \theta_i^j + \log(1 - z_i^j) \prod_{k \in S_i^j} z_k^j \right) - \sum_{j=1}^c (\lambda_i^j - 1) \log \theta_i^j$$

$$= -\left( \log |S_i| + \frac{1}{|S_i|} \sum_{j \in S_i} \log(1 - z_i^j) \prod_{k \in S_i^j} z_k^j \right) + \sum_{j \in S_i} \left( \lambda_i^j - 1 + \frac{1}{|S_i|} \right) \log \theta_i^j + \sum_{j \in S_i} (\lambda_i^j - 1) \log \theta_i^j$$

$$= -\left( K_i + \sum_{j=1}^c w_i^j \ell(f(x_i), e^j) \right)$$

where $e^j$ denotes the standard canonical vector in $\mathbb{R}^c$, $w_{ij} = \lambda_i^j - 1 + \frac{1}{|S_i|}$ if $j \in S_i$, otherwise $w_{ij} = 0$, and $K_i = \log |S_i| + \frac{1}{|S_i|} \sum_{j \in S_i} \log(1 - z_i^j) \prod_{k \in S_i^j} z_k^j$.

### A.2 Proof of Lemma 1

In order to prove this lemma, we first show that the one direction $\sup_{f \in \mathcal{F}} R(f) - \hat{R}(f)$ is bounded with probability at least $1 - \frac{\xi}{2}$, and the other direction can be similarly shown. Suppose an example $(x_i, S_i)$ is replaced by another arbitrary example $(x_i', S_i')$, then the change of $\sup_{f \in \mathcal{F}} R(f) - \hat{R}(f)$ is no greater than $M/(2n)$, the loss function $L$ are bounded by $M$. By applying McDiarmid’s inequality, for any $\xi > 0$, with probability at least $1 - \frac{\xi}{2}$,

$$\sup_{f \in \mathcal{F}} R(f) - \hat{R}(f) \leq E \left[ \sup_{f \in \mathcal{F}} R(f) - \hat{R}(f) \right] + \frac{M}{2} \sqrt{\frac{\log \frac{2}{\xi}}{2n}} \quad (19)$$

By symmetrization, we can obtain

$$E \left[ \sup_{f \in \mathcal{F}} R(f) - \hat{R}(f) \right] \leq 2 \mathcal{R}_n(G). \quad (20)$$

By further taking into account the other side $\sup_{f \in \mathcal{F}} R(f) - \hat{R}(f)$, we have for any $\xi > 0$, with probability at least $1 - \frac{\xi}{2}$,

$$\sup_{f \in \mathcal{F}} |R(f) - \hat{R}(f)| \leq 2 \mathcal{R}_n(G) + \frac{M}{2} \sqrt{\frac{\log \frac{2}{\xi}}{2n}} \quad (21)$$

### A.3 Proof of Lemma 2

The upper bound loss function of $L$ is denoted by

$$\bar{L}(f(x_i, S_i)) = -\left( K_i + \sum_{j=1}^c w_i^j \ell(f(x_i), e^j) \right)$$

Correspondingly, the function space for $\bar{L}$ can be defined as:

$$\mathcal{G} = \{ (x, S) \mapsto \bar{L}(f(x), S) | f \in \mathcal{F} \}$$
Then the expected Rademacher complexity of $\tilde{G}$ can be defined as follows:

$$\tilde{R}_n(G) = \mathbb{E}_{x, S, \sigma} \left[ \sup_{\tilde{g} \in \tilde{G}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \tilde{g}(x, S_i) \right]$$

(24)

For each example $(x_i, S_i)$, we bound $w_i^j$ in $[0, \rho]$. Hence, we can obtain $\tilde{R}_n(G) \leq \tilde{R}_n(\tilde{G}) \leq \rho c \tilde{R}_n(\ell \circ F) + K$ where $\ell \circ F$ denotes $\{ \ell \circ f | f \in F \}$. Since $H_i = \{ h: x \mapsto f_y(x) | f \in F \}$ and the loss function $\ell (f(x, e'))$ is $L$-Lipschitz for all $e' \in Y$, by the Rademacher vector contraction inequality, we have $\tilde{R}_n(\ell \circ F) \leq \sqrt{2L} \sum_{e' \in Y} \tilde{R}_n(H_i)$. Then the proof is completed.

A.4 Proof of Theorem 1

Theorem is proven through

$$R(\hat{f}) - R(f^*) = R(\hat{f}) - \tilde{R}(\hat{f}) + \tilde{R}(\hat{f}) - \tilde{R}(f^*) + \tilde{R}(f^*) - R(f^*)$$

$$\leq R(\hat{f}) - \tilde{R}(\hat{f}) + \tilde{R}(f^*) - R(f^*)$$

$$\leq 2 \sup_{f \in F} \left| R(f) - R(f) \right|$$

$$\leq 4 \tilde{R}_n(G) + M \sqrt{\frac{\log \frac{\delta}{2}}{2n}}$$

(25)

$$\leq 4 \sqrt{2} \rho c L \sum_{e' \in Y} \tilde{R}_n(H_i) + M \sqrt{\frac{\log \frac{\delta}{2}}{2n}} + 4K$$

14