LEARNING FROM POSITIVE AND UNLABELED DATA WITH A SELECTION BIAS

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider the problem of learning a binary classifier only from *positive* data and *unlabeled* data (*PU learning*). Recent methods of PU learning commonly assume that the labeled positive data are identically distributed as the unlabeled positive data. However, this assumption is unrealistic in many instances of PU learning because it fails to capture the existence of a *selection bias* in the labeling process. When the data has a selection bias, it is difficult to learn the Bayes optimal classifier by conventional methods of PU learning. In this paper, we propose a method to *partially identify* the classifier. The proposed algorithm learns a scoring function that preserves the order induced by the class posterior under mild assumptions, which can be used as a classifier by setting an appropriate threshold. Through experiments, we show that the method outperforms previous methods for PU learning on various real-world datasets.

1 INTRODUCTION

We consider a situation that there are only *positive* and *unlabeled* data, and train a binary classifier only from them (*PU learning*). This problem arises in various practical situations, such as information retrieval and outlier detection (Elkan & Noto, 2008; Ward et al., 2009; Scott & Blanchard, 2009; Blanchard et al., 2010; Li et al., 2009; Nguyen et al., 2011). One of the milestones of PU learning is Elkan & Noto (2008), who proposed a practically useful algorithm with theoretical analysis, and there is subsequent research called *unbiased PU learning* (du Plessis & Sugiyama, 2014; du Plessis et al., 2015) where an *unbiased estimator* of the classification risk is minimized.

We focus on the *case-control* scenario (a.k.a. the *problem setting based on two samples of data*; Ward et al., 2009; Niu et al., 2016). In this scenario, positive data are obtained separately from unlabeled data, and unlabeled data are sampled from the whole population. As Elkan & Noto (2008) explained, we cannot identify a classifier without an assumption on how positive data are labeled. Therefore "*selected completely at random*" (SCAR) is traditionally assumed, i.e., the positive labeled data are identically distributed as the positive unlabeled data (Elkan & Noto, 2008; du Plessis et al., 2015). The assumption of SCAR, however, is unrealistic in many instances of PU learning, e.g., a patient's electronic health record (Bekker & Davis, 2018a) and a recommendation system (Marlin & Zemel, 2009; Schnabel et al., 2016). In these cases, there is a *selection bias* (Heckman, 1979; Manski, 2008; Angrist & Pischke, 2008); the distribution of the positive data may differ between the labeled data and the unlabeled data.

Several recent related work has also proposed alternative assumptions (Bekker & Davis, 2018a;b). However, in order to weaken SCAR, they impose other additional assumptions. In this work, we consider a more natural assumption such that SCAR becomes its special case. We assume that the order over $\{\boldsymbol{x}\}$ induced by the conditional probability of observation $p(o = +1|\boldsymbol{x})$ is the same as the one induced by the conditional class probability $p(y = +1|\boldsymbol{x})$. We call this property the *invariance of order*. In the real-world application, there are many situations with a selection bias which follows the invariance of order. Among them, we list the following two examples.

Example 1: (Anomaly Detection)
The goal of *anomaly detection* is to find anomaly data in an unlabeled dataset based on another dataset that consists only of anomaly data. When the anomaly data is obtained, the more likely a

datum is an anomaly, the more likely it is noticed and gets labeled. Here, we consider transductive learning. It means, only from this two datasets, we classify the unlabeled dataset.

Example 2: (Face Recognition)
The goal of this task is to identify a user from a set of face images based on some pictures of the user. Users may tend to provide pictures which are more easily identifiable as themselves. In this case, the positive data is the user's face and the unlabeled data consists of all users including the user. The user may provide pictures in which the face can be seen clearly, while in the unlabeled data there may be a lot of pictures in which the user's face is covered by the hair.

Our problem setting is similar to the problem called *learning from instance-dependent noisy label* (Du & Cai, 2015; Bootkrajang, 2016). In this problem setting, positive and negative data are available, but some labels of data flip from positive to negative and negative to positive with the instance-dependent probability as a noise. Besides, there is existing work putting assumption similar to the invariance order (Du & Cai, 2015; Bootkrajang, 2016). We explain the difference between this work and our work just after Assumption 1

We name our problem setting *PU learning with a Selection Bias* (PUSB). In this paper, we propose a novel framework to deal with this problem setting. The experimental results show that our proposed method is appropriate for real-world applications compared to existing approaches for PU learning.

2 Problem Setting of PU Learning with a Selection Bias

We consider a binary classification problem to classify $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ into one of the two classes $\{-1, +1\}$. We assume that there exists a joint distribution $p(\boldsymbol{x}, y, o)$, where $y \in \{-1, +1\}$ is the class label of $\boldsymbol{x}$, and $o \in \{0, 1\}$ is the observation status of y (observed if $o = +1$ and unobserved if $o = 0$). In other words, x is labeled if $o = +1$, and it is unlabeled otherwise.

In PU learning, there are two distinguished sampling schemes called *one sample* and *two samples* of data (Niu et al., 2016). They are also called the *censoring scenario* and *case-control scenario*, respectively (Elkan & Noto, 2008). In the censoring scenario, a set of unlabeled data is sampled from the marginal density $p(\boldsymbol{x})$. Then, if a data point $\boldsymbol{x}$ is positive, this positive data gets labeled with probability $p(o = +1|\boldsymbol{x})$; if $\boldsymbol{x}$ is negative, it is never labeled. In the case-control scenario, a set of positive data is drawn from the positive conditional density $p(\boldsymbol{x}|y = +1)$ and a set of unlabeled data is drawn from $p(\boldsymbol{x})$. As Niu et al. (2016) stated, the case-control scenario is slightly more general than the censoring scenario setting. It is because the censoring scenario assumes the access to samples generated by $p(\boldsymbol{x})$, $p(\boldsymbol{x}|o = +1)$, and $p(\boldsymbol{x}|o = 0)$, but the case-control scenario only assumes the access to samples generated by $p(\boldsymbol{x})$ and $p(\boldsymbol{x}|o = +1)$. Therefore we focus on the case-control scenario.

Suppose that we have a positive dataset $\{\boldsymbol{x}_i\}_{i=1}^n$ and an unlabeled dataset $\{\boldsymbol{x}'_i\}_{i=1}^{n'}$

$$\{\boldsymbol{x}_i\}_{i=1}^n \sim p(\boldsymbol{x}|y = +1, o = +1),$$
$$\{\boldsymbol{x}'_i\}_{i=1}^{n'} \sim p(\boldsymbol{x}).$$

We assume that negative data are never labeled.

In addition, we do not assume SCAR. Therefore, $p(\boldsymbol{x}|y = +1)$ may differ from $p(\boldsymbol{x}|y = +1, o = +1)$. In the case they differ, we say that there is a *selection bias*.

The quantity $\pi = p(y = +1)$ is called the *class-prior*. In our problem setting, we consider the case where the class-prior is known. For example, in anomaly detection, the percentage of anomaly in the whole batch of products can be reported based on past experiences. Although there are various methods for estimating the class-prior in the traditional framework of the case-control scenario (du Plessis et al., 2016; Ramaswamy et al., 2016; Jain et al., 2016; Kato et al., 2018), we cannot estimate the class-prior in our problem setting under a theoretical guarantee. In Section 5, we show how misspecified class priors affect the performance of a classifier. As we explain later, even if we do not know the class-prior, we only have to change the last step of our algorithm.

Our goal is to obtain a classifier $h : \mathcal{X} \to \{-1, 1\}$ only from $\{\boldsymbol{x}_i\}_{i=1}^n$, $\{\boldsymbol{x}'_i\}_{i=1}^{n'}$, and π, which is justifiable under some kind of evaluation measure.

2.1 IDENTIFICATION STRATEGY

As stated by Elkan & Noto (2008), even if the class prior is given, we cannot estimate $p(y=+1|x)$ only from positive data and unlabeled data without any assumption in PU learning. In the case-control scenario, a standard assumption is SCAR, i.e. $p(x|y=+1,o=+1)=p(x|y=+1,o=0)$, so that $p(y=+1|x)$ can be estimated from the data in principle. However, in many instances of PU learning, the SCAR assumption is unreasonable. Therefore, we relax SCAR and accommodate a selection bias. We can see how SCAR makes the class posterior identifiable in the following equation:

$$p(y=+1|\boldsymbol{x})=\frac{p(\boldsymbol{x},y=+1)}{p(\boldsymbol{x})}=\frac{p(\boldsymbol{x}|y=+1)\pi}{p(\boldsymbol{x})}\underbrace{=}_{\text{SCAR}}\frac{p(\boldsymbol{x}|y=+1,o=+1)\pi}{p(\boldsymbol{x})}.$$

Under the assumption of SCAR, we can estimate each of $p(\boldsymbol{x}|y=+1)$ and $p(\boldsymbol{x})$ from samples, in principle. However, without assuming SCAR, $p(\boldsymbol{x}|y=+1)$ may differ from $p(\boldsymbol{x}|y=+1,o=+1)$, and $p(y=+1|\boldsymbol{x})$ is not identifiable.

Therefore, instead of estimating $p(y=+1|\boldsymbol{x})$ and estimation of the Bayes-optimal classifier and $p(y=+1|\boldsymbol{x})$, we consider extracting some useful information of $p(y=+1|\boldsymbol{x})$ to learn a classifier. This kind of approach is known as "partial identification" (Manski, 2008) in statistics and economics. The idea is to extract some useful information of a function instead of attempting to identify the whole function. Firstly, we introduce an assumption that is weaker than SCAR.

Assumption 1 (Invariance of Order)**.** *For any $\boldsymbol{x}_i,\boldsymbol{x}_j\in\mathcal{X}$, we have*

$$p(y=+1|\boldsymbol{x}_i)\leq p(y=+1|\boldsymbol{x}_j)\Leftrightarrow p(o=+1|\boldsymbol{x}_i)\leq p(o=+1|\boldsymbol{x}_j).$$

Although Assumption 1 does not allow one to construct an unbiased estimator of the risk functional, we try to partially identify $p(y=+1|\boldsymbol{x})$ under this assumption. Our problem setting can be regarded as a generalization of the traditional case-control scenario because SCAR is a special case of the invariance of order.

A similar assumption can be found in the literature of learning from instance-dependent noisy label (Du & Cai, 2015; Bootkrajang, 2016), which consider a probabilistic label flipping that is proportional to $p(y=+1|\boldsymbol{x})$. However, in order to apply methods of learning from noisy label to PU learning, we need to assume the censoring scenario and these methods are not applicable to our problem setting based on the case-control scenario. The censoring scenario is a special case of learning from noisy label where only negative data is contaminated, i.e., some positive labels flip to negative labels. Thus, in the censoring scenario, unlabeled data can be regarded as negative-labeled data contaminated by positive data. On the other hand, in the case-control scenario, unlabeled data is generated from the marginal distribution p(x), i.e., we cannot observe samples generated from p(xlo = 0). Therefore, our problem setting, namely the case-control scenario with invariance of order, is different from the existing work of learning from instance-dependent noisy label. In addition, our method is also applicable to the censoring scenario when the invariance of order holds because the unlabeled data of the case-control scenario can be made from positive and unlabeled data of the censoring scenario.

In Example 1, $p(y=+1|\boldsymbol{x})$ is the probability that a given input $\boldsymbol{x}$ is anomaly, while $p(o=+1|\boldsymbol{x})$ is the probability that a given input $\boldsymbol{x}$ gets labeled in the dataset. In Example 2, a positively labeled data is an image $\boldsymbol{x}$ that is known to belong to a user. Here, $p(o=+1|\boldsymbol{x})$ is the probability that the user provides the picture $\boldsymbol{x}$ as a training datum.

3 THE STRATEGY FOR PARTIAL IDENTIFICATION AND CLASSIFICATION

As we discussed in Section 2.1, we cannot estimate $p(y=+1|\boldsymbol{x})$ when there is a selection bias even if the class prior is given. Our idea of partial identification is based on the following theorem. Let $r(\boldsymbol{x})$ be

$$r(\boldsymbol{x})=\frac{p(\boldsymbol{x}|y=+1,o=+1)}{p(\boldsymbol{x})}. \tag{1}$$

Theorem 1 (Order preserving property of the score function)**.** *Suppose that Assumption 1 holds. Then, for any $\boldsymbol{x}_i,\boldsymbol{x}_j\in\mathcal{X}$,*

$$p(y=+1|\boldsymbol{x}_i)\leq p(y=+1|\boldsymbol{x}_j)\Leftrightarrow r(\boldsymbol{x}_i)\leq r(\boldsymbol{x}_j)$$

holds.

A proof is provided in Appendix A. Even though we cannot estimate $p(y = +1|\cdot)$, Theorem 1 implies that we can still extract the total order in $\mathcal{X}$ induced by $p(y = +1|\cdot)$ if we can estimate $r(\cdot)$. Therefore, we propose to estimate r and use it as a score function that captures the total order induced by $p(y = +1|\cdot)$. After obtaining an estimator of r (denoted by $\hat{r}$), we set a threshold $\theta \in \mathbb{R}$ and use $h(\boldsymbol{x}) = \mathtt{sign}(r(\boldsymbol{x}) - \theta)$ as a classifier. There are various ways of determining a threshold. For instance, we put labels from data with the highest density ratio under a constraint of the number of data to which we can put labels (Hido et al., 2011). Here, we introduce one useful principle for choosing θ. We consider using a threshold θ_π defined by the following equation,

$$\pi = \int \mathbf{1}[r(\boldsymbol{x}) \geq \theta_\pi] p(\boldsymbol{x}) d\boldsymbol{x}. \tag{2}$$

The intuition behind the definition of θ_π is that the proportion of the positive data in the test data points should not deviate so much from the true class-prior. This intuition becomes clearer in Section 4.3.

In Section 4, we discuss detailed methods for estimating r and setting θ based on data. Our approach is summarized in the form of pseudo code in Algorithm 1. In the rest of this section, we theoretically justify θ_π defined in (2).

Property of θ_π: Let us consider the case where a classifier is given as $h(\boldsymbol{x}) = \mathtt{sign}(r(\boldsymbol{x}) - \theta)$. Then four population quantities, *true positives* (TP), *true negatives* (TN), *false positives* (FP), and *false negatives* (FN) (Lipton et al., 2014), that depend on $r(\cdot)$ and θ is written as follows:

$$TP = \int_{\{\boldsymbol{x}: r(\boldsymbol{x}) \geq \theta\}} p(y = +1|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}, \quad FP = \int_{\{\boldsymbol{x}: r(\boldsymbol{x}) \geq \theta\}} p(y = -1|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x},$$
$$TN = \int_{\{\boldsymbol{x}: r(\boldsymbol{x}) < \theta\}} p(y = -1|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}, \quad FN = \int_{\{\boldsymbol{x}: r(\boldsymbol{x}) < \theta\}} p(y = +1|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}.$$

Then, the *precision* and the *recall* of a classifier are expressed as $\mathrm{precision} = \left(\frac{TP}{TP+FP}\right)$ and $\mathrm{recall} = \left(\frac{TP}{TP+FN}\right)$, respectively. For θ_π, we have the following result.

Theorem 2. *If we use $\theta = \theta_\pi$, then* $\mathrm{precision} = \mathrm{recall}$ *holds.*

A proof is shown in Appendix B. A threshold which makes the precision and the recall the same is known as *precision–recall breakeven point* (BEP) (Sammut & Webb, 2010). BEP is originally used to evaluate a generic classification model with a score function and a threshold. In addition, we can interpret BEP as a point which balances a prediction result. As explained by Powers (2015), a classifier using BEP as a threshold put the same weight to the false positives and false negatives. Knowing BEP is also useful for deciding on a threshold which put unbalanced weight on the precision and the recall because we can tell if we are weighing precision more or recall more. If the threshold is higher than BEP, we can tell that we are using a classifier that puts more weight on the precision, and vice versa. By setting the threshold relative to BEP, we can choose which of precision and recall is prioritized.

4 ALGORITHM

Here, we propose two directions for estimating $r(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=+1, o=+1)}{p(\boldsymbol{x})}$ under the IO assumption, namely minimizing a *pseudo classification risk* and *direct density ratio estimation*. We also discuss how to set θ based on the data. A pseudo-code of our algorithm is shown in Algorithm 2.

4.1 ESTIMATION OF r BY MINIMIZING THE PSEUDO CLASSIFICATION RISK

Firstly, we introduce the minimization of the pseudo classification risk. The idea is to minimize the classification risk used in traditional PU learning (du Plessis et al., 2014; 2015) as if there is no selection bias. Under a selection bias, we cannot construct unbiased risk function, but the minimizer can be substituted for the density ratio in (1).

Conventional PU risk formulation: Let $\ell : \mathbb{R} \times \{\pm 1\} \to \mathbb{R}^+$ be a loss function, where $\mathbb{R}^+$ is the set of non-negative real values, and $\mathcal{F}$ be a set of measurable functions from $\mathcal{X}$ to $[\epsilon, 1-\epsilon]$, where ϵ is a small positive value. This ϵ is introduced to make the following optimization problem well-defined.

du Plessis et al. (2015) showed that the classification risk of $f \in \mathcal{F}$ in the traditional PU problem setting with SCAR can be expressed as

$$R_{\mathrm{PU}}(f) = \pi \mathbb{E}_{\mathrm{p}}[\ell(f(X), +1)] - \pi \mathbb{E}_{\mathrm{p}}[\ell(f(X), -1)] + \mathbb{E}_{\mathrm{u}}[\ell(f(X), -1)], \tag{3}$$

where $\mathbb{E}_{\mathrm{p}}$ and $\mathbb{E}_{\mathrm{u}}$ are the expectations over $p(\boldsymbol{x}|y=+1)$ and $p(\boldsymbol{x})$, respectively. When there is no selection bias, we can simply replace the expectations with the corresponding sample averages to obtain an unbiased estimator of the classification risk.

The pseudo classification risk: In our problem setting, we only have samples from $p(\boldsymbol{x}|y = +1, o = +1)$ and not from $p(\boldsymbol{x}|y = +1)$. Therefore, we cannot use our sample to obtain an empirical version of R_{PU}. However, we still consider the *pseudo classification risk* of $f \in \mathcal{F}$:

$$R_{\mathrm{PU}}^{\mathtt{bias}}(f, \ell) = \pi \mathbb{E}_{\mathrm{p}}^{\mathtt{bias}}[\ell(f(X), +1)] - \pi \mathbb{E}_{\mathrm{p}}^{\mathtt{bias}}[\ell(f(X), -1)] + \mathbb{E}_{\mathrm{u}}[\ell(f(X), -1)],$$

where $\mathbb{E}_{\mathrm{p}}^{\mathtt{bias}}$ is the expectation over $p(\boldsymbol{x}|y = +1, o = +1)$. We call this functional the pseudo classification risk because it is not the true classification risk. An unbiased estimator for the pseudo classification risk can be obtained by replacing the expectations with the corresponding sample averages even if there is a selection bias. For the loss function, we use the logarithmic loss: $\ell(f(\boldsymbol{x}), +1)) = -\log(f(\boldsymbol{x}))$ and $\ell(f(\boldsymbol{x}), -1) = -\log(1 - f(\boldsymbol{x}))$. In this case, the pseudo classification risk of $f \in \mathcal{F}$ becomes

$$R_{\mathrm{PU}}^{\mathrm{bias}}(f) = -\pi \mathbb{E}_{\mathrm{p}}^{\mathtt{bias}}[\log(f(X))] + \pi \mathbb{E}_{p}^{\mathtt{bias}}[\log(1 - f(X))] - \mathbb{E}_{\mathrm{u}}[\log(1 - f(X))]. \tag{4}$$

Justification for minimizing the pseudo classification risk: For the pseudo classification risk with the logarithmic loss function, the following theorem justifies its use. Let us denote a minimizer of (4) by f^*, that is,

$$f^* \in \operatorname*{arg\,min}_{f \in \mathcal{F}} R_{\mathrm{PU}}^{\mathrm{bias}}(f).$$

For the minimizer of (4), we derive the following theorem.

Theorem 3. *f^* is given by*

$$f^*(\boldsymbol{x}) = \begin{cases} \epsilon & (\boldsymbol{x} \notin D^1), \\ \frac{\pi p(\boldsymbol{x}|y=+1, o=+1)}{p(\boldsymbol{x})} & (\boldsymbol{x} \in D^1 \cap D^2), \\ 1 - \epsilon & (\boldsymbol{x} \notin D^2), \end{cases}$$

where $D^1 = \{\boldsymbol{x} | \pi p(\boldsymbol{x}|y = 1, o = +1) \geq \epsilon p(\boldsymbol{x})\}$ and $D^2 = \{\boldsymbol{x} | \pi p(\boldsymbol{x}|y = 1, o = +1) \leq (1 - \epsilon) p(\boldsymbol{x})\}$.

A proof is provided in Appendix C. Theorem 3 implies that the minimization of the empirical version of the pseudo classification risk allows us to estimate r.

Empirical Estimation: When we have training samples, we can naively replace the expectations by the corresponding sample averages. For a hypothesis set $\mathcal{H}$, which is a subset of a set of measurable functions, let us define the following risk minimization problem,

$$\hat{f} = \operatorname*{arg\,min}_{f \in \mathcal{H}} \left[-\pi \hat{\mathbb{E}}_{p}^{\mathtt{bias}}[\log(f(X))] + \pi \hat{\mathbb{E}}_{p}^{\mathtt{bias}}[\log(1 - f(X))] - \hat{\mathbb{E}}_{u}[\log(1 - f(X))] \right], \tag{5}$$

where $\hat{\mathbb{E}}_{p}^{\mathtt{bias}}$ denotes the empirical mean using the positive dataset with a selection bias and $\hat{\mathbb{E}}_{u}$ denotes the empirical mean using the unlabeled dataset. du Plessis et al. (2015) showed that, under SCAR, the empirical version of the risk becomes unbiased toward the classification risk.

However, Kiryo et al. (2017) pointed out that unbiased PU learning does not work with deep learning. Minimizing an empirical risk of (3) with deep learning easily causes over-fitting because the risk is not bounded from below by 0. In order to implement PU learning with deep learning, Kiryo et al. (2017) proposed the following non-negative risk,

$$\hat{f} = \operatorname*{arg\,min}_{f \in \mathcal{H}} \left[-\pi \hat{\mathbb{E}}_{p}^{\mathtt{bias}}[\log(f(X))] + \max\left\{0, \pi \hat{\mathbb{E}}_{p}^{\mathtt{bias}}[\log(1 - f(X))] - \hat{\mathbb{E}}_{u}[\log(1 - f(X))]\right\} \right]. \tag{6}$$

After obtaining $\hat{f}$, we set $\hat{r} = \frac{1}{\pi}\hat{f}$.

Algorithm 1 Conceptual Algorithm in Population

Input: $p(\boldsymbol{x}|y=+1)$, $p(\boldsymbol{x})$ and the class-pror π.
Using $p(\boldsymbol{x}|y=+1)$ and $p(\boldsymbol{x})$, calculate $r(\boldsymbol{x})$ by minimization of either (4) or (7).
Using $r(\boldsymbol{x})$, calculate θ_π in (2).
Using the density ratio $r(\boldsymbol{x})$ and the threshold θ_π, obtain a classifier $h(\boldsymbol{x}) = \mathtt{sign}(r(\boldsymbol{x}) - \theta_\pi)$.

4.2 Estimation of r by Direct Density Ratio Estimation

For another approach, we consider estimating the density ratio $r(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=+1,o=+1)}{p(\boldsymbol{x})}$ directly. In order to estimate the ratio, there are various methods. For example, we can estimate the probability density functions of the numerator and the denominator. However, as Vapnik's principle, we should avoid solving more difficult intermediate problems. Sugiyama et al. (2012) summarized various methods estimating the density ratio directly. Among existing methods of density ratio estimation, we employ *Least-squares importance fitting* (LSIF), which uses the squared loss for density-ratio function fitting. The reason for this choice is that there is an algorithm called *unconstrained Least-Squares Importance Fitting* (uLSIF), which is based on LSIF and allows us to obtain the closed-form solution that can be computed just by solving a system of linear equations. Thus, uLSIF is numerically stable when it is regularized properly. Moreover, the leave-one-out cross-validation score for uLSIF can also be computed analytically, which significantly improves the computational efficiency in model selection.

Here, we introduce the formulation of LSIF. Let S be a class of non-negative measurable functions $s : \mathcal{X} \to \mathbb{R}^+$. We consider minimizing the following squared error between s and r:

$$R_{\mathrm{DR}}(s) = \mathbb{E}_{\mathrm{u}}[(s(X) - r(X))^2] = \mathbb{E}_{\mathrm{u}}[(r(X))^2] - 2\mathbb{E}_{\mathrm{p}}^{\mathtt{bias}}[s(X)] + \mathbb{E}_{\mathrm{u}}[(s(X))^2]. \tag{7}$$

The first term of the last equation does not affect the result of minimization and we can ignore the term, i.e., the density ratio is estimated through the following minimization problem:

$$s^* = \underset{s\in\mathcal{S}}{\arg\min}\, R_{\mathrm{DR}}(s) = \underset{s\in\mathcal{S}}{\arg\min}\left[\frac{1}{2}\mathbb{E}_{\mathrm{u}}[(s(X))^2] - \mathbb{E}_{\mathrm{p}}^{\mathtt{bias}}[s(X)]\right].$$

Empirical Estimation: As mentioned above, to minimize the empirical version of (7), we introduce uLSIF (See Kanamori et al. (2009)). For a hypotheses set $\mathcal{H}$, we obtain $\hat{r}$ by

$$\hat{r} = \underset{s\in\mathcal{H}}{\arg\min}\left[\frac{1}{2}\hat{\mathbb{E}}_{\mathrm{u}}[(s(X))^2] - \hat{\mathbb{E}}_{\mathrm{p}}^{\mathtt{bias}}[s(X)]\right]. \tag{8}$$

4.3 Empirically estimating θ_π

We consider replacing the threshold defined by (2) with a value which can be calculated only from samples. In the case that we have access to the test inputs

$$\{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n^{\mathrm{te}}} \sim p(\boldsymbol{x}),$$

we choose a $\hat{\theta}_\pi$ that satisfies the following equation,

$$\lceil \pi n^{\mathrm{te}} \rceil = \sum_{i=1}^{n^{\mathrm{te}}} \mathbf{1}[\hat{r}(\boldsymbol{x}_i^{\mathrm{te}}) > \hat{\theta}_\pi]. \tag{9}$$

Here, we used the knowledge of π, the class-prior. This choice of $\hat{\theta}_\pi$ amounts to classifying top-π test data as positive after ranking the inputs by $\hat{r}$.

In the case we do not have access to the test inputs, we use held-out training data instead of test data to estimate the threshold analogically to (9).

5 Experiments

In this section, we report experimental results which were conducted using synthetic data and real-world datasets[1)]. We used six classification datasets, `MNIST`, `shuttle`, `pageblocks`, `usps`,

[1)] All codes of experiments can be downloaded from github

Algorithm 2 PUSB

Input: A positive dataset $\{\boldsymbol{x}_i\}_{i=1}^{n}$, an unlabeled dataset $\{\boldsymbol{x}'_i\}_{i=1}^{n'}$, a test dataset $\{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n^{\mathrm{te}}}$ and the class-pror π.
Using the positive dataset and the unlabeled dataset, estimate $r(\boldsymbol{x})$ by any of (5), (6) or (8) and obtain $\hat{r}(\boldsymbol{x})$.
Using $\hat{r}(\boldsymbol{x})$, estimate θ_π by (9) and obtain $\hat{\theta}_\pi$.
Using an estimator $\hat{r}(\boldsymbol{x})$ and $\hat{\theta}$, obtain a classifier $h(\boldsymbol{x}) = \mathtt{sign}(\hat{r}(\boldsymbol{x}) - \hat{\theta}_\pi)$.

Table 1: Dataset statistics (Pos.frac.: Positive fraction, Dim: Dimension).

Dataset	# of samples	Pos. frac.	Dim.
waveform	5000	0.492	21
mushroom	8124	0.517	112
spambase	4601	0.394	57
MNIST	70000	0.511	784
usps	9298	0.524	256
connect-4	67557	0.658	126
CIFAR-10	60000	0.400	3,072

connect-4 and spambase, from UCI repository[2)], CIFAR-10[3)] and a document dataset obtained from SwissProt (Boeckmann et al., 2003)[4)]. MNIST and CIFAR-10 have 10 and 10 classes originally, and we constructed the positive and negative datasets from them as follows: MNIST was preprocessed in such a way that 0, 2, 4, 6, 8 constitute the positive class, while 1, 3, 5, 7, 9 constitute the negative class; for CIFAR-10, the positive dataset is formed by 'airplane', 'automobile', 'ship' and 'truck', and the negative dataset is formed by 'bird', 'cat', 'deer', 'dog', 'frog' and 'horse'. Except for the document dataset, we show the details of datasets in Table 1 and made positive data with a selection bias based on estimators of $p(y = +1|\boldsymbol{x})$. For six datasets of the UCI repository and the CIFAR-10, we made positive datasets with a selection bias artificially, but, for the document dataset, we have an unlabeled dataset and a positive dataset, which is gathered for classifying the labels in the unlabeled dataset.

We call unbiased PU learning proposed by du Plessis et al. (2015) "PU", unbiased PU learning with a threshold estimated by (9) "PUSB", uLSIF with a threshold estimated by (9) "DRSB", nonnegative PU learning "nnPU" and nonnegative PU learning with a threshold estimated by (9) "nnPUSB".

Let us denote a real-valued function as $g(\boldsymbol{x})$ and use it for approximating f and s. In the risk minimization of PU learning (4) and the density ratio estimation (7), we assume a generic real-valued function for $g(\boldsymbol{x})$, but we use linear-in-parameter model in the empirical risk of unbiased PU learning (5) and the density ratio estimation (8), and deep neural network in that of nonnegative PU learning (6). When we used linear-in-parameter model, we assumed the following model,

$$g(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{\varphi}(\boldsymbol{x}), \tag{10}$$

where $\boldsymbol{\varphi}(\boldsymbol{x}) = [1, \varphi_1(\boldsymbol{x}), ..., \varphi_m(\boldsymbol{x})]^\top$ is a set of basis functions. For basis functions, we used the Gaussian functions centered around sample points $\varphi_\ell(\boldsymbol{x}) = \exp\left(-\|\boldsymbol{x} - \boldsymbol{c}_\ell\|^2/(2\sigma^2)\right)$, where $\{\boldsymbol{c}_1, ..., \boldsymbol{c}_m\} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{x}'_1, ..., \boldsymbol{x}'_{n'}\}$ and $m = n + n'$. A regularization parameter is given by λ and insert a regularization term $\frac{\lambda}{2}\boldsymbol{\beta}^\top\boldsymbol{\beta}$ to the empirical risk of PU learning (5) and $\lambda \sum_{i=1}^{m} \beta_i$ to that of the density ratio estimation (8). The model for deep neural network were explained in the following sections for each dataset. We mainly used the same structure proposed in Kiryo et al. (2017) in order to compare the performances.

For approximating $s(\boldsymbol{x})$, we used $g(\boldsymbol{x})$ directly.

For approximating f, we used the sigmoid function:

$$f(\boldsymbol{x}) = \frac{1}{1 + \exp(-g(\boldsymbol{x}))}.$$

[2)]The UCI data were downloaded from https://archive.ics.uci.edu/ml/index.php and https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/.

[3)]See https://www.cs.toronto.edu/~kriz/cifar.html.

[4)]The data can be downloaded from http://www.cs.ucsd.edu/users/elkan/posonly.

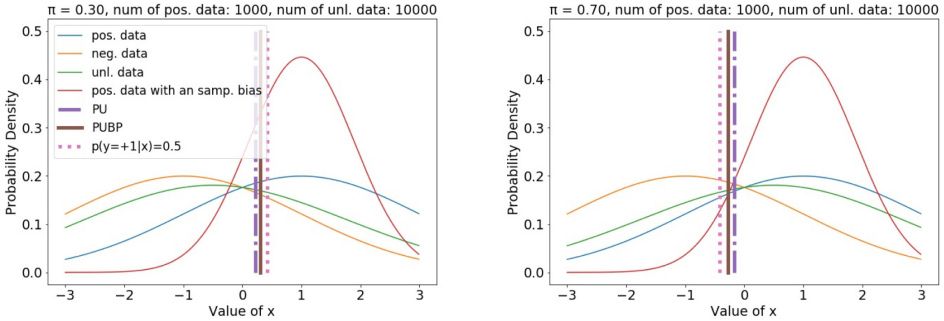

Figure 1: Two Gaussians: The horizontal axis is value of x and the vertical axis is probability density. The vertical line represents the classifier. The distribution of positive data, negative data, unlabeled data and $p(x = +1|o = +1, y = +1)$ is plotted.

In this case, the loss is the same as the logistic loss and unbiased PU learning becomes convex.

In order to made a selection biases for the datasets, we estimated $p(y = +1|\boldsymbol{x})$ firstly. For the estimation, we used all samples with the true positive and negative labels and logistic regression with the same model as that we used for PU learning.

5.1 Test with Synthetic Data

This experiment shows a classifier given by PUSB. We used samples from a mixture distribution of the following two class-conditional distributions:

$$p(x|y = +1) = \mathcal{N}_x(1, 2^2) \text{ and } p(x|y = -1) = \mathcal{N}_x(-1, 2^2),$$

where $\mathcal{N}(\mu, \sigma^2)$ denotes the univariate normal distribution with mean μ and variance σ^2. A positive dataset with a selection bias is sampled from

$$p(o = +1|x = +1, y = +1) \propto (p(y = +1|x))^{10}.$$

We generated 1000 positive samples and 10000 unlabeled samples. We made two datasets with the different class-priors $\pi = 0.3$, $\pi = 0.7$. We plotted three classifiers estimated by PU, PUSB and $p(y = +1|\boldsymbol{x})$. The results are shown in Figure 1. The classifiers obtained by our algorithm nearer to the Bayesian optimal classifier for the classification error rate, $\texttt{sign}(p(y = +1|x) - 1/2)$.

5.2 Benchmark Test

In this subsection, we investigated the experimental performance in detail.

Linear-in-parameter model: We used the `mushrooms`, `shuttle`, `pageblocks`, `usps`, `connect-4` and `spambase` datasets.

Firstly, we estimated $p(y = +1|\boldsymbol{x})$ using the logistic regression with the same linear model. Then, from positive data, we sampled positive dataset with a selection bias as

$$p(o = +1|\boldsymbol{x}, y = +1) \propto (p(y = +1|\boldsymbol{x}))^{20}.$$

Then, we trained a classifier by minimizing the empirical risk of PU learning (4) and the density ratio estimation (7).

For each binary labeled dataset, we made 12 different pairs of positive and unlabeled data with 4 different class-priors, $\{0.2, 0.4, 0.6, 0.8\}$, and 3 different numbers of unlabeled data, $\{800, 1600, 3200\}$. The number of positive data was fixed at 400. We used 1000 test data sampled from the same distribution as the unlabeled data. We ran the experiments 100 times and calculated the mean and standard deviation for the test dataset with PU, PUSB, and DRSB. The results are shown in Table 2. The classifiers obtained by our algorithm always show preferable performance to existing methods.

Table 2: The error rate of classification in test data (%) are shown for the different class-priors and the different number of samples. For all experiments, linear-in-parameter model was used. Best and equivalent methods (under 5% t-test) are bold.

		PU			PUSB			DRSB		
Dataset	π	800	1600	3200	800	1600	3200	800	1600	3,200
mushrooms	0.2	5.1 (.076)	5.0 (.055)	5.5 (.083)	**4.1** (.007)	**3.8** (.007)	**4.0** (.007)	10.7 (.010)	9.9 (.011)	9.9 (.012)
	0.4	**7.0** (.012)	**6.6** (.010)	**6.8** (.025)	**6.9** (.012)	**6.7** (.010)	**6.8** (.010)	15.8 (.014)	15.0 (.016)	15.2 (.012)
	0.6	10.8 (.020)	11.3 (.015)	11.3 (.016)	**8.2** (.013)	**8.5** (.010)	**8.3** (.011)	18.5 (.014)	18.8 (.012)	18.5 (.013)
	0.8	21.1 (.022)	21.6 (.021)	21.7 (.016)	**8.1** (.015)	**8.0** (.012)	**7.6** (.014)	14.2 (.013)	14.3 (.014)	14.4 (.011)
shuttle	0.2	23.8 (.006)	23.8 (.007)	23.8 (.007)	5.2 (.008)	**5.0** (.007)	**5.0** (.007)	**5.1** (.008)	**5.1** (.008)	5.1 (.007)
	0.4	15.5 (.023)	15.5 (.019)	15.0 (.016)	7.8 (.011)	**7.7** (.011)	7.6 (.012)	**6.7** (.025)	**7.6** (.029)	**7.2** (.027)
	0.6	26.1 (.018)	26.4 (.017)	26.4 (.016)	11.1 (.015)	10.8 (.012)	11.0 (.015)	**7.8** (.028)	**7.4** (.028)	**7.1** (.028)
	0.8	14.6 (.006)	14.6 (.005)	14.8 (.007)	12.3 (.014)	12.4 (.014)	12.3 (.012)	**7.9** (.021)	**7.6** (.023)	**7.4** (.022)
pageblocks	0.2	39.5 (.009)	39.6 (.010)	39.6 (.008)	41.6 (.021)	42.5 (.022)	43.7 (.019)	**23.1** (.016)	**23.1** (.015)	**22.2** (.012)
	0.4	56.5 (.011)	56.7 (.011)	56.6 (.008)	33.7 (.020)	33.7 (.019)	33.5 (.024)	**23.6** (.012)	**23.7** (.014)	**23.6** (.014)
	0.6	22.4 (.023)	23.4 (.033)	28.1 (.028)	20.9 (.016)	21.1 (.016)	21.5 (.020)	**18.7** (.011)	**18.6** (.012)	**18.1** (.017)
	0.8	19.8 (.003)	20.0 (.001)	20.0 (.001)	**13.9** (.022)	**13.6** (.023)	16.7 (.043)	15.4 (.011)	14.8 (.014)	**14.3** (.019)
usps	0.2	9.0 (.011)	8.5 (.009)	8.2 (.010)	**8.0** (.009)	**7.7** (.007)	**7.4** (.009)	18.2 (.016)	19.6 (.013)	19.7 (.014)
	0.4	**10.5** (.012)	**10.3** (.013)	10.0 (.010)	**10.5** (.013)	**10.2** (.012)	**10.0** (.010)	30.5 (.029)	30.2 (.022)	29.9 (.023)
	0.6	12.5 (.015)	12.3 (.015)	12.1 (.013)	**11.2** (.016)	**10.9** (.015)	**10.6** (.013)	32.9 (.026)	33.2 (.031)	33.1 (.030)
	0.8	19.8 (.020)	19.8 (.016)	19.5 (.017)	**10.3** (.016)	**9.8** (.014)	**9.6** (.014)	25.9 (.028)	25.3 (.027)	25.8 (.030)
connect-4	0.2	22.3 (0.017)	**21.9** (.014)	21.6 (.013)	**21.8** (.013)	**21.5** (.012)	**21.2** (.011)	26.7 (.012)	26.6 (.012)	26.4 (.011)
	0.4	**31.2** (.015)	**31.0** (.015)	**30.7** (.016)	**31.2** (.015)	**31.0** (.015)	**30.7** (.016)	39.4 (.016)	39.6 (.018)	39.2 (.016)
	0.6	32.0 (.017)	**32.1** (.013)	31.9 (.015	**31.7** (.016)	**31.7** (.013)	**31.4** (.016)	40.9 (.015)	41.1 (.017)	41.0 (.017)
	0.8	33.6 (.018)	33.5 (.016)	33.4 (.017)	**24.2** (.013)	**23.9** (.013)	**23.8** (.013)	29.8 (.011)	29.6 (.013)	29.5 (.012)
spambase	0.2	20.1 (0.002)	20.1 (.002)	20.1 (.003)	**13.6** (.013)	**13.8** (.014)	**14.0** (.014)	18.1 (.026)	18.3 (.025)	17.9 (.023)
	0.4	36.2 (.024)	35.9 (.025)	30.7 (.024)	**19.0** (.020)	**18.7** (.021)	**18.9** (.018)	27.4 (.042)	27.8 (.043)	27.7 (.039)
	0.6	40.0 (.001)	39.9 (.001)	31.9 (.001	**20.8** (.019)	**20.7** (.018)	**20.0** (.017)	31.2 (.037)	30.3 (.035)	31.2 (.035)
	0.8	20.0 (.000)	20.0 (.000)	20.0 (.000)	**18.0** (.015)	**17.7** (.013)	**17.3** (.013)	24.5 (.058)	23.9 (.017)	24.2 (.017)

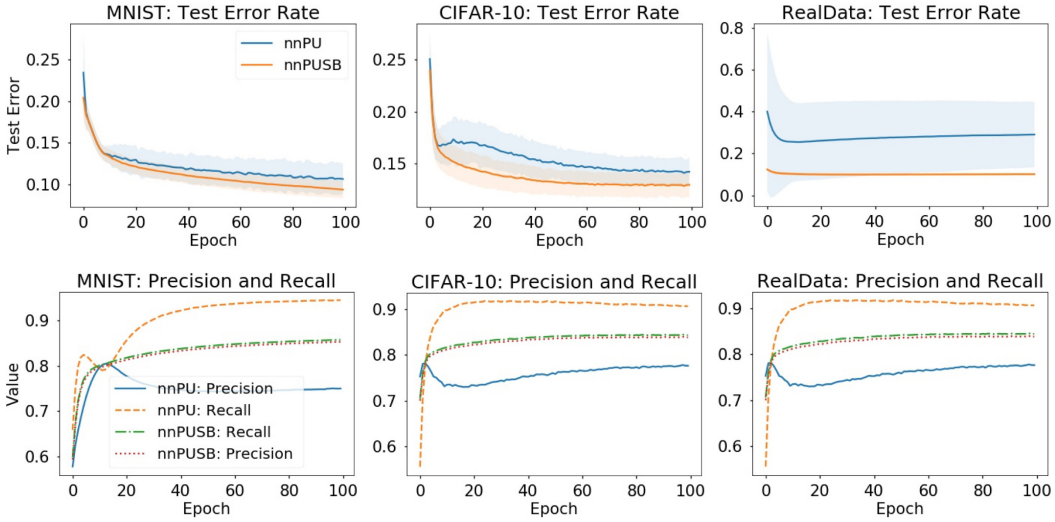

Figure 2: Experimental results of training deep neural networks. Left: `MNIST`; Center: `CIFAR-10`; Right: `RealData`: All measures are calculated for test data sampled from the marginal distribution $p(\boldsymbol{x})$. The horizontal line is epoch of training the network, the vertical line of the upper low is error rate and the vertical line of the lower low is value of the precision and the recall.

Neural network model: We used the `MNIST` and `CIFAR-10` datasets. The model of $g(\boldsymbol{x})$ for the `MNIST` dataset was a 3-layer *multilayer perceptron* (MLP) with ReLU (Nair & Hinton, 2010). The model for the `CIFAR-10` dataset was an all convolutional net (Springenberg et al., 2015). Details of the network structure are shown in Appendix D.

Firstly, we estimated $p(y = +1|\boldsymbol{x})$ using the logistic regression with the same network structure and the positive and unlabeled datasets in the unlabeled dataset. Next, from the positive dataset, we resampled positive dataset with an observation, which follows

$$p(o = +1|\boldsymbol{x}, y = +1) \propto (p(y = +1|\boldsymbol{x}))^{10}.$$

Then, we trained a classifier by minimizing (6) with the model defined above. We used 10000 test data sampled from the same distribution as the unlabeled data. We ran the experiments 100 times and calculated the mean of the error rate, standard deviation of the error rate, the mean of recall and

the mean of precision for each epoch in training with nnPU and nnPUSB. The results are shown on the left side and center of Figure 2. In the upper row, we show the mean and standard deviation of the error rate. In the lower row, we show the mean of recall and the mean of precision. As shown in Figure 2, the mean of the error rate of our algorithm was lower and the variance was also lower than the existing method. As a matter of fact should be noted, for our results, $\mathrm{FP} = \mathrm{FN}$ held as we discussed in Section 4.3.

5.3 Real-world Data Test

We demonstrate the effectiveness of our algorithm in a real-world application, we used a document dataset recorded from the SwissProt database and released by Elkan & Noto (2008). The dataset consists of three datasets, a positive dataset P, another positive dataset Q, and a negative dataset N. We call this real dataset `RealData`. The positive dataset P contains 2453 examples obtained from a specialized database named TCDB (Saier Jr et al., 2015). Meanwhile, there are 4906 records in the unlabeled dataset U, which are randomly selected from SwissProt excluding those in the TCDB. In other words, those 2 datasets P and U are disjoint. Hence, this belongs to TS. Furthermore, Das et al. (2007) also manually labeled the unlabeled dataset. They identified 348 positive members from the whole dataset N, i.e., the class prior is $\pi = 348/4906 = 0.0709$. They called the new positive examples as Q and the leftover was N. We used Bag-of-Words to represent all examples of documents as vectors. We represented each document as a 78894-dimensional real-valued vector. This real-world dataset is often used to evaluate algorithms of PU learning such as Elkan & Noto (2008); He et al. (2018).

The model of $g(\boldsymbol{x})$ for this dataset was a 5-layer *multilayer perceptron* (MLP) with ReLU. Details of the network structure are shown in Appendix D. We trained a classifier using positive and unlabeled data. Then, after finding a threshold estimated by (9), we evaluated the same evaluation measures as the previous experiments of a neural network by classifying U, i.e., we regarded the unlabeled data as test data. The results are shown on the right side of Figure 2. The result obtained from our algorithm outperformed the existing method. In addition, we made positive data with a selection bias by ourselves for Benchmark Test, but FP=FN also held in the real dataset with a selection bias. It means that we cannot ignore a selection bias in real-world applications and our algorithm could be more useful in practice.

5.4 Sensitivity Analysis

In order to empirically measure the sensitivity of our algorithm to the class prior, we conduct two experiments using the `RealData` dataset in Section 5.3, whose class prior is estimated to be 0.0709. In the first experiment, we trained a classifier under misspecified class priors, namely $0.0209(= 0.0709-0.0500)$, $0.1209(= 0.0709+0.0500)$, $0.1709(= 0.0709+0.1000)$, and $0.2709(= 0.0709 + 0.2000)$. In the second experiment, we empirically tested our algorithm with an estimator of the class prior by Ramaswamy et al. (2016), which is considered to be the state-of-the-art method in the case-control scenario under SCAR. Among the two methods proposed by Ramaswamy et al. (2016), we adopted "KM2" for this experiment because "KM2" is reported to be better. The estimated class prior was 0.1095. All results are shown in Appendix E. In so far as this dataset, our method works well under perturbations of the class prior or with an estimated class prior.

6 Conclusion

In this paper, we proposed a novel framework for PU learning with a selection bias in positive data. We put the IO assumption and showed the density ratio of labeled positive data and unlabeled data has the same order as the conditional class distribution for inputs. Based on this result, we proposed an idea of partial identification in which we first estimate the density ratio and then use it as a classifier by setting a threshold. We conducted experiments to confirm the effectiveness of our approach. As we showed in the experiments, our method outperforms previous PU methods on real-world data.

REFERENCES

Joshua D. Angrist and Jörn-Steffen Pischke. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press, December 2008.

Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data under the selected at random assumption. *arXiv*, 2018a.

Jessa Bekker and Jesse Davis. Beyond the selected completely at random assumption for learning from positive and unlabeled data. *arXiv*, 2018b.

Gilles Blanchard, Gyemin Lee, and Clayton Scott. Semi-supervised novelty detection. *Journal of Machine Learning Research*, 11(Nov):2973–3009, 2010.

Brigitte Boeckmann, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J Martin, Karine Michoud, Claire O'donovan, Isabelle Phan, et al. The SWISS-PROT protein knowledgebase and its supplement trembl in 2003. *Nucleic acids research*, 31(1): 365–370, 2003.

Jakramate Bootkrajang. A generalised label noise model for classification in the presence of annotation errors. *Neurocomputing*, 192(C):61–71, 2016.

Sanmay Das, Milton H Saier, and Charles Elkan. Finding transport proteins in a general protein database. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 54–66. Springer, 2007.

Jun Du and Zhihua Cai. Modelling class noise with symmetric and asymmetric distributions. In *AAAI Conference on Artificial Intelligence*, AAAI'15, pp. 2589–2595. AAAI Press, 2015.

M. C. du Plessis, Gang. Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *NIPS*, pp. 703–711, 2014.

Marthinus Christoffel du Plessis and Masashi Sugiyama. Class prior estimation from positive and unlabeled data. *IEICE Transactions on Information and Systems*, E97-D(5):1358–1362, 2014.

Marthinus Christoffel du Plessis, Gang. Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *ICML*, pp. 1386–1394, 2015.

Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. Class-prior estimation for learning from positive and unlabeled data. In *ACML*, pp. 221–236, 2016.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *ICDM*, pp. 213–220, 2008.

Fengxiang He, Tongliang Liu, Geoffrey I Webb, and Dacheng Tao. Instance-dependent PU learning by bayesian optimal relabeling. *arXiv*, 2018.

James Heckman. Sample selection bias as a specification error. *Econometrica*, 47(1):153–61, 1979.

Shohei Hido, Yuta Tsuboi, Hisashi Kashima, Masashi Sugiyama, and Takafumi Kanamori. Statistical outlier detection using direct density ratio estimation. *Knowledge and Information Systems*, 26(2): 309–336, Feb 2011.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pp. 448–456, 2015.

Shantanu Jain, Martha White, Michael W Trosset, and Predrag Radivojac. Nonparametric semi-supervised learning of class proportions. In *NIPS*, 2016.

Takafumi. Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul.):1391–1445, 2009.

Masahiro Kato, Liyuan Xu, Gang Niu, and Masashi Sugiyama. Alternate estimation of a classifier and the class-prior from positive and unlabeled data. *arXiv*, 2018.

Ryuichi Kiryo, Gang Niu, Marthinus Christoffel du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *NIPS*, pp. 1675–1685, 2017.

Xiao-Li Li, Philip S Yu, Bing Liu, and See-Kiong Ng. Positive unlabeled learning for data stream classification. In *ICDM*, pp. 259–270, 2009.

Zachary C. Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal thresholding of classifiers to maximize F1 measure. In *Machine Learning and Knowledge Discovery in Databases*, pp. 225–239, 2014.

Charles Manski. *Partial Identification in Econometrics*. Palgrave-Macmillan, 2 edition, 2008.

Benjamin M. Marlin and Richard S. Zemel. Collaborative prediction and ranking with non-random missing data. In *ACM Conference on Recommender Systems*, RecSys '09, pp. 5–12, 2009.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Minh Nhut Nguyen, Xiaoli-Li Li, and See-Kiong Ng. Positive unlabeled leaning for time series classification. In *IJCAI*, pp. 1421–1426, 2011.

Gang Niu, Marthinus Christoffel du Plessis, Tomoya. Sakai, Y. Ma, and Masashi Sugiyama. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *NIP*, pp. 1199–1207, 2016.

David M. W. Powers. Visualization of tradeoff in evaluation: from precision-recall & PN to LIFT, ROC & BIRD. *arXiv*, 2015.

Harish Ramaswamy, Clayton Scott, and Ambuj Tewari. Mixture proportion estimation via kernel embeddings of distributions. In *ICML*, pp. 2052–2060, 2016.

Milton H Saier Jr, Vamsee S Reddy, Brian V Tsu, Muhammad Saad Ahmed, Chun Li, and Gabriel Moreno-Hagelsieb. The transporter classification database (TCDB): recent advances. *Nucleic acids research*, 44(D1):D372–D379, 2015.

Claude Sammut and Geoffrey I. Webb (eds.). *Breakeven Point*, pp. 137–138. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_88.

Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *ICML*, volume 48, pp. 1670–1679, New York, New York, USA, 20–22 Jun 2016.

Clayton Scott and Gilles Blanchard. Novelty detection: Unlabeled data definitely help. In *AISTATS*, pp. 464–471, 2009.

J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*, 2015.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, New York, NY, USA, 1st edition, 2012.

Gill Ward, Trevor Hastie, Simon Barry, Jane Elith, and John R Leathwick. Presence-only data and the em algorithm. *Biometrics*, 65(2):554–563, 2009.

A Proof of Theorem 1

Proof. We assumed that no negative data can be labeled. As a result, we have $p(o = +1|\boldsymbol{x}, y = -1) = 0$ for arbitrary $x \in \mathcal{X}$. Therefore,

$$\begin{aligned} p(o = +1|\boldsymbol{x}) &= p(y = +1|\boldsymbol{x})p(o = +1|\boldsymbol{x}, y = +1) + p(y = -1|\boldsymbol{x})p(o = +1|\boldsymbol{x}, y = -1) \\ &= p(y = +1|\boldsymbol{x})p(o = +1|\boldsymbol{x}, y = +1). \end{aligned} \tag{11}$$

By Bayes' theorem, we can expand the density ratio in (1) as follows:

$$\begin{aligned} r(\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|o=+1, y=+1)}{p(\boldsymbol{x})} \\ &= \frac{p(y=+1|\boldsymbol{x}, o=+1)p(\boldsymbol{x}|o=+1)}{p(y=+1|o=+1)} \frac{1}{p(\boldsymbol{x})} \\ &= p(y=+1|\boldsymbol{x}, o=+1) \frac{1}{p(y=+1|o=+1)} \frac{p(\boldsymbol{x}|o=+1)}{p(\boldsymbol{x})} \\ &= \frac{p(y=+1|\boldsymbol{x})p(o=+1|\boldsymbol{x}, y=+1)}{p(o=+1|\boldsymbol{x})} \frac{1}{p(y=+1|o=+1)p(o=+1)} \frac{p(o=+1)p(\boldsymbol{x}|o=+1)}{p(\boldsymbol{x})} \end{aligned}$$

Because $\frac{p(y=+1|\boldsymbol{x})p(o=+1|\boldsymbol{x},y=+1)}{p(o=+1|\boldsymbol{x})} = 1$ from (11), this is equivalent to

$$r(\boldsymbol{x}) = Cp(o=+1|\boldsymbol{x})$$

where $C = \frac{1}{p(y=+1,o=+1)}$. Hence, if Assumption 1 holds, for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$,

$$p(y=+1|\boldsymbol{x}_i) \leq p(y=+1|\boldsymbol{x}_j) \Leftrightarrow r(\boldsymbol{x}_i) \leq r(\boldsymbol{x}_j).$$

holds. □

B Proof of Theorem 2

Proof. Regarding the probability as a classifier, let us decide a threshold for $p(y=+1|\boldsymbol{x})$ as follows:

$$\pi = \int \mathbf{1}[p(y=+1|\boldsymbol{x}) \geq \gamma] p(\boldsymbol{x}) d\boldsymbol{x}. \tag{12}$$

This is equivalent to

$$\int p(y=+1|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \int \mathbf{1}[p(y=+1|\boldsymbol{x}) \geq \gamma] p(\boldsymbol{x}) d\boldsymbol{x}.$$

The left hand side is equal to

$$\int_{\{\boldsymbol{x}|p(y=+1|\boldsymbol{x})<\gamma\}} p(y=+1|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} + \int_{\{\boldsymbol{x}|p(y=+1|\boldsymbol{x})\geq\gamma\}} p(y=+1|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}.$$

The right hand side is equal to

$$\int_{\{\boldsymbol{x}|p(y=+1|\boldsymbol{x})\geq\gamma\}} p(y=+1|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} + \int_{\{\boldsymbol{x}|p(y=+1|\boldsymbol{x})\geq\gamma\}} p(y=-1|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}.$$

Hence, (12) is equivalent to the following equation,

$$\int_{\{\boldsymbol{x}|p(y=+1|\boldsymbol{x})<\gamma\}} p(y=+1|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \int_{\{\boldsymbol{x}|p(y=+1|\boldsymbol{x})\geq\gamma\}} p(y=-1|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}.$$

From the definition of *true positive* (TP), *false positive* (FP), *true negative* (TN) and *false negative* (FN), the left hand side of the above equation is equal to $\frac{FP}{TP+FP+TN+FN}$ and the right hand side of the above equation is equal to $\frac{FN}{TP+FP+TN+FN}$. Therefore, this means that the threshold classifies data as the number of negative data predicted as positive is equal to the number of positive data predicted as negative, i.e., $FP = FN$.

Then, we state the following lemma about the relationship between γ and θ.

Lemma 1. *For all $\boldsymbol{x} \in \mathcal{X}$, $(r(\boldsymbol{x}) - \theta)(p(y=+1|\boldsymbol{x}) - \gamma) \geq 0$ holds almost surely.*

Proof. By (2) and (12), the following equation also hold,

$$\int \mathbf{1}[r(\boldsymbol{x}) > \theta] p(\boldsymbol{x}) d\boldsymbol{x} = \int \mathbf{1}[p(y=+1|\boldsymbol{x}) > \gamma] p(\boldsymbol{x}) d\boldsymbol{x}.$$

Hence, we can derive

$$\int (\mathbf{1}[r(\boldsymbol{x}) > \theta] - \mathbf{1}[p(y=+1|\boldsymbol{x}) > \gamma])p(\boldsymbol{x})d\boldsymbol{x} = 0.$$

This is equivalent to

$$\int (\mathbf{1}[(r(\boldsymbol{x}) - \theta)(p(y=+1|\boldsymbol{x}) - \gamma) < 0]p(\boldsymbol{x})d\boldsymbol{x} = 0.$$

From this equation, $(r(\boldsymbol{x}) - \theta)(p(y=+1|\boldsymbol{x}) - \gamma) \geq 0$ holds almost surely. □

Therefore, if $p(y=+1|\boldsymbol{x}) < \gamma$, then $r(\boldsymbol{x}) < \theta$; if $p(y=+1|\boldsymbol{x}) \geq \gamma$, then $r(\boldsymbol{x}) \geq \theta$ almost surely. Because $\text{precision} = \left(\frac{TP}{TP+FP}\right)$ and $\text{recall} = \left(\frac{TP}{TP+FN}\right)$, $FP = FN$ means that $\text{precision} = \text{recall}$.Therefore, using a threshold defined by (2), we can obtain a classifier whose precision and the recall are the same. □

C Proof of Theorem 3

Proof. We minimize the functional $R_{\mathtt{PU}}^{\mathtt{bias}}$ in the space of all functions on $\mathcal{X}$ taking values in $[\epsilon, 1-\epsilon]$, instead of $\mathcal{F}$, and we will see that the minimizer matches the f^* stated in Theorem 3 which belongs to $\mathcal{F}$. Because we are optimizing

$$R_{\mathtt{PU}}^{\mathtt{bias}}(f) := \int \left(-\pi p(\boldsymbol{x}|y=+1)(\log f(\boldsymbol{x}) - \log(1-f(\boldsymbol{x}))) - \log(1-f(\boldsymbol{x}))p(\boldsymbol{x})\right) \mathrm{d}\boldsymbol{x}$$

over all functions taking values in $[\epsilon, 1-\epsilon]$, the optimization is reduced to a point-wise minimization of

$$l(f(\boldsymbol{x})) := (-\pi p(\boldsymbol{x}|y=+1)(\log f(\boldsymbol{x}) - \log(1-f(\boldsymbol{x}))) - \log(1-f(\boldsymbol{x}))p(\boldsymbol{x}))$$

by considering $f(\boldsymbol{x})$ as the optimization variable subject to $f(\boldsymbol{x}) \in [\epsilon, 1-\epsilon]$. The Karush-Kuhn-Tucker (KKT) condition of this minimization problem is

$$\begin{aligned}
&\pi p(\boldsymbol{x}|y=+1)\left(\frac{1}{f^*(\boldsymbol{x})} + \frac{1}{1-f^*(\boldsymbol{x})}\right) - \frac{p(\boldsymbol{x})}{1-f^*(\boldsymbol{x})} - \lambda + \mu = 0,\\
&\epsilon \leq f^*(\boldsymbol{x}) \leq 1-\epsilon,\\
&\lambda, \mu \geq 0,\\
&\lambda(f^*(\boldsymbol{x}) - 1 - \epsilon) = 0, \mu f^*(\boldsymbol{x}) = 0,
\end{aligned}$$

where λ and μ are the Lagrange multipliers. The first equation is equivalent to

$$(\mu - \lambda)(f^*(\boldsymbol{x}))^2 + (p(\boldsymbol{x}) + \lambda - \mu)f^*(\boldsymbol{x}) - \pi p(\boldsymbol{x}|y=+1) = 0. \tag{13}$$

From these conditions, the following results:

1. Assume $\lambda = \mu = 0$. $f^*(\boldsymbol{x}) = \frac{\pi p(\boldsymbol{x}|y=+1)}{p(\boldsymbol{x})}$. This solution satisfies the constraint only when $\boldsymbol{x}$ is in the domain D, where $D = \{x|p(\boldsymbol{x}|y=1) \leq (1-\epsilon)p(\boldsymbol{x})\}$.
2. Assume $\lambda > 0$ and $\mu = 0$, $f^*(\boldsymbol{x}) = 1-\epsilon$. It is because $\inf_f \mathcal{L}(f;\alpha,\beta) \to +\infty$ as $\lambda \to +\infty$ if $f^*(\boldsymbol{x}) < 1-\epsilon$. $\lambda > 0$ holds only when $\boldsymbol{x}$ is in the domain $\mathbb{R}\backslash D^2$, where $\mathbb{R}\backslash D^2 = \{x|\pi p(\boldsymbol{x}|y=1) < (1-\epsilon)p(\boldsymbol{x})\}$, because

$$\begin{aligned}
&-\lambda(1-\epsilon)^2 + (p(\boldsymbol{x}) + \lambda)(1-\epsilon) - \pi p(\boldsymbol{x}|y=+1) = 0\\
&\Leftrightarrow\ -\lambda(1-\epsilon)^2 + \lambda(1-\epsilon) + p(\boldsymbol{x})(1-\epsilon) - \pi p(\boldsymbol{x}|y=+1) = 0\\
&\Leftrightarrow\ \lambda = \frac{p(\boldsymbol{x})(1-\epsilon) - \pi p(\boldsymbol{x}|y=+1)}{(1-\epsilon)^2 + (1-\epsilon)}.
\end{aligned}$$

3. Assume $\lambda = 0$ and $\mu > 0$, $f^*(\boldsymbol{x}) = \epsilon$. It is because $\inf_f \mathcal{L}(f; \alpha, \beta) \to +\infty$ as $\mu \to +\infty$ if $f^*(\boldsymbol{x}) > \epsilon$. This solution satisfies (13), only when $\boldsymbol{x}$ is in the domain $\mathbb{R}\backslash D^1$, where $\mathbb{R}\backslash D^1 = \{x | \pi p(\boldsymbol{x}|y=1) > \epsilon p(\boldsymbol{x})\}$, because

$$\mu(\epsilon)^2 + (p(\boldsymbol{x}) - \mu)\epsilon - \pi p(\boldsymbol{x}|y=+1) = 0$$
$$\Leftrightarrow \ \mu = \frac{p(\boldsymbol{x})\epsilon - \pi p(\boldsymbol{x}|y=+1)}{\epsilon^2 - \epsilon}.$$

Otherwise, there is no feasible solution. As a result, the solution for the optimization problem $f^*(\boldsymbol{x}) = \arg\min_{f(\boldsymbol{x}) \in [\epsilon, 1-\epsilon]} R_{\mathtt{PU}}^{\mathtt{bias}}(f)$ is

$$f^*(\boldsymbol{x}) = \begin{cases} \epsilon & (\boldsymbol{x} \notin D^1), \\ \frac{\pi p(\boldsymbol{x}|y=+1, o=+1)}{p(\boldsymbol{x})} & (\boldsymbol{x} \in D^1 \cap D^2), \\ 1 - \epsilon & (\boldsymbol{x} \notin D^2), \end{cases}$$

where $D^1 = \{\boldsymbol{x} | \pi p(\boldsymbol{x}|y=1, o=+1) \geq \epsilon p(\boldsymbol{x})\}$ and $D^2 = \{\boldsymbol{x} | \pi p(\boldsymbol{x}|y=1, o=+1) \leq (1-\epsilon) p(\boldsymbol{x})\}$. Now, because the function $f^*(\boldsymbol{x})$ constructed above is a measurable function if $p(\boldsymbol{x}|y=1, o=+1)$ and $p(\boldsymbol{x})$ are measurable, the solution of the original optimization problem

$$\arg\min_{f \in \mathcal{F}} R_{\mathtt{PU}}^{\mathtt{bias}}(f)$$

is equal to this f^*. □

D The Network Structure used in Section 5.2 and Section 5.3

In Section 5.2, we used the `MNIST` and `CIFAR-10` datasets. The model of $g(\boldsymbol{x})$ for the `MNIST` dataset was a 3-layer *multilayer perceptron* (MLP) with ReLU (Nair & Hinton, 2010) (more specifically, $784 - 100 - 1$). The model for the `CIFAR-10` dataset was an all convolutional net (Springenberg et al., 2015): $(32 \times 32 \times 3)$-$[C(3 \times 3, 96)] \times 2$-$C(3 \times 3, 96, 2)$-$[C(3 \times 3, 192)] \times 2$-$C(3 \times 3, 192, 2)$-$C(3 \times 3, 192)$-$C(1 \times 1, 192)$-$C(1 \times 1, 10)$-1000-1000-1, where the input is a 32×32 RGB image, $C(3 \times 3, 96)$ means 96 channels of 3×3 convolutions followed by ReLU, $[\cdot] \times 2$ means there are two such layers, $C(3 \times 3, 96, 2)$ means a similar layer but with stride 2, etc.; it is one of the best architectures for `CIFAR-10`. Batch normalization (Ioffe & Szegedy, 2015) was applied before hidden layers.

In Section 5.3, the model of $g(\boldsymbol{x})$ for this dataset was a 5-layer *multilayer perceptron* (MLP) with ReLU (more specifically, 78894-300-300-300-300-1).

E Experimental Results of Section 5.4

In order to empirically measure the sensitivity of a classifier to the class prior, we conduct two experiments using the `RealData` dataset in Section 5.3, whose true class prior is 0.0709.

In the first experiment, we trained a classifier under misspecified class priors, $0.0209 (= 0.0709 - 0.0500)$, $0.1209 (= 0.0709 + 0.0500)$, $0.1709 (= 0.0709 + 0.1000)$, and $0.2709 (= 0.0709 + 0.2000)$. The results are shown in Figures 3–5. In Figure 3, the test error in each misspecified class priors are shown. In Figure 4, the precision and recall in each misspecified class prior is shown. In both existing method and our method, misspecified class priors had a bad influence. However, our method was less influenced by the misspecification and showed better performance than the existing method. Besides, the difference between the precision and recall of our algorithm is narrower than that of existing method as expected. In order to discuss how misspecified class prior affects the precision and recall, we subtract the recall from the precision and show the values in Figure 5. This result shows how the difference broadens as the class prior is misspecified worse.

In the second experiment, we empirically tested our algorithm with an estimated class prior. In order to estimate the class prior, we used Ramaswamy et al. (2016), which is considered to be the state-of-the-art method in the case-control scenario under SCAR. Among two methods proposed by Ramaswamy et al. (2016), we adopted "KM2" for this experiment because "KM2" is reported

to be the better one. The mean of the estimated class prior was 0.1095. This experiment assumes the situation that we face in the real world. The result is shown in Figure 5. In so far as the `RealData` dataset, our method still works empirically with an estimated class prior although there is no theoretical guarantee.

There is no theoretical justification for using the existing methods of the class prior estimation in our problem setting. Although we could not propose a universal approach for this problem, we still could apply the existing method to our problem setting for a specific dataset.

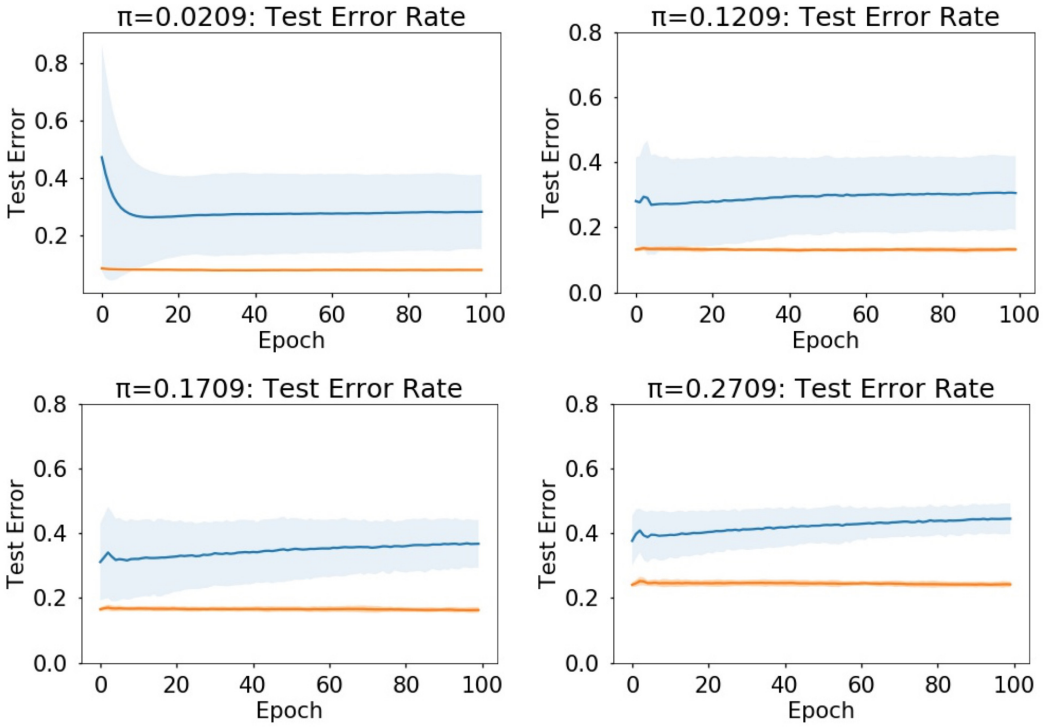

Figure 3: Experimental results of training deep neural networks using the `RealData` dataset with misspecified class priors (the true class prior is 0.0709). Upper Left: $\pi = 0.0209$; Upper Right: $\pi = 0.1209$; Lower Left: $\pi = 0.1709$; Lower Right: $\pi = 0.2709$: All measures are calculated for test data sampled from the marginal distribution $p(\boldsymbol{x})$. The horizontal line is epoch of training the network, the vertical line of the upper is error rate.

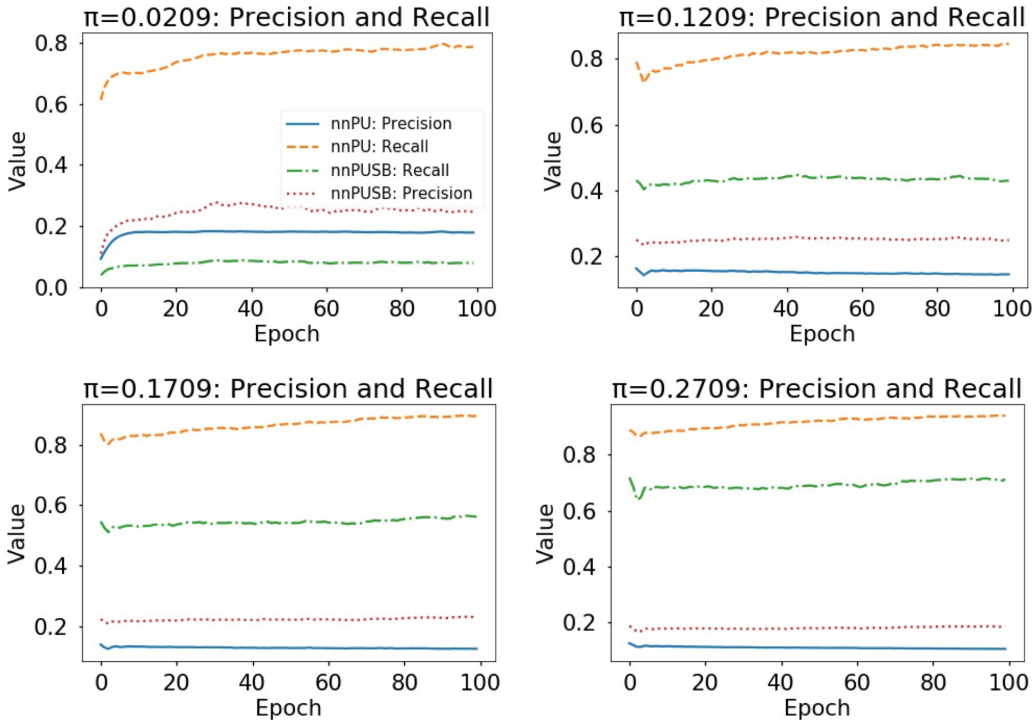

Figure 4: Experimental results of training deep neural networks using the `RealData` dataset with misspecified class priors (the true class prior is 0.0709). Upper Left: $\pi = 0.0209$; Upper Right: $\pi = 0.1209$; Lower Left: $\pi = 0.1709$; Lower Right: $\pi = 0.2709$: All measures are calculated for test data sampled from the marginal distribution $p(\boldsymbol{x})$. The horizontal line is epoch of training the network, the vertical line of the upper is value of the precision and the recall.

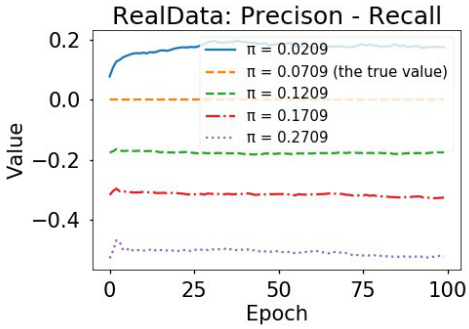

Figure 5: Experimental results of training deep neural networks using the `RealData` dataset, whose true class prior is 0.0709, with misspecified class priors, 0.0209, 0.1209, 0.1709, and 0.2709.: All measures are calculated for test data sampled from the marginal distribution $p(\boldsymbol{x})$. The horizontal line is epoch of training the network, the vertical line of the right graph is value of the $\mathrm{precision} - \mathrm{recall}$.

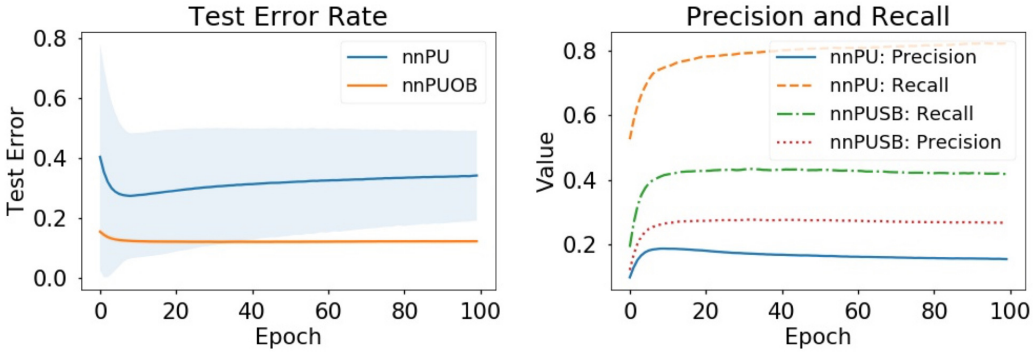

Figure 6: Experimental results of training deep neural networks using the `RealData` dataset with the estimated class prior (the true class prior is 0.0709). The estimated class prior was 0.1095: All measures are calculated for test data sampled from the marginal distribution $p(\boldsymbol{x})$. The horizontal line is epoch of training the network, the vertical line of the right graph is error rate and the vertical line of the left graph is value of the precision and the recall.