Importance-weighted Positive-unlabeled Learning for Distribution Shift Adaptation

Atsutoshi Kumagai[1] Tomoharu Iwata[1] Hiroshi Takahashi[1]
Taishi Nishiyama[2,1] Yasuhiro Fujiwara[1]
[1]NTT Corporation [2]NTT Security Holdings

Abstract

Positive and unlabeled (PU) learning is a fundamental task in many applications that trains a binary classifier from only PU data. Existing PU learning methods typically assume that training and test distributions are identical. However, this assumption is often violated due to distribution shifts, and shift types such as covariate and concept shifts are generally difficult to identify. In this paper, we propose a distribution shift adaptation method for PU learning without assuming shift types by using a few PU data in the test distribution and PU data in the training distribution. Our method is based on the importance weighting, which can learn the classifier in a principled manner by minimizing the importance-weighted training risk that approximates the test risk. Although existing methods require positive and negative data in both distributions for the importance weighting without assuming shift types, we theoretically show that it can be performed with only PU data in both distributions. On the basis of this finding, our neural network-based classifiers can be effectively trained by iterating the importance weight estimation and classifier learning. We experimentally show that our method outperforms various existing methods with seven real-world datasets.

1 INTRODUCTION

Positive and unlabeled (PU) learning addresses the problem of learning a binary classifier from only PU data without negative data (Bekker and Davis, 2020). It has attracted attention since negative data are often difficult to collect in many applications. For example, in personalized advertising, data liked by the user (positive data) can be collected from clicked ads, but disliked data (negative data) are difficult to collect because unclicked ads contain both liked and disliked ones (Bekker and Davis, 2020). In outlier detection, only normal (positive) data and unlabeled data are often available since anomalous (negative) data are rare (Hido et al., 2011). In addition to these, PU learning can be applied to many other applications such as medical diagnosis (Zuluaga et al., 2011), drug discovery (Liu et al., 2017), text classification (Li and Liu, 2003), and remote sensing (Li et al., 2010).

Although many PU learning methods have been proposed (Fung et al., 2005; Hou et al., 2018; Chen et al., 2020; Luo et al., 2021; Zhao et al., 2022; Du Plessis et al., 2015; Kiryo et al., 2017), they usually assume that training and test distributions are identical. However, in practice, this assumption is often violated. For example, in personalized advertising, users' preferences usually shift over time. In outlier detection, when data are collected in different environments, the data distribution can change (Kumagai et al., 2019). When such a distribution shift occurs, the performance of these methods drastically deteriorates.

To deal with this problem, several PU learning methods for distribution shift adaptation have recently been proposed (Sakai and Shimizu, 2019; Hammoudeh and Lowd, 2020; Mignone and Pio, 2018). Most methods use unlabeled data in the test distribution, which are easy to prepare, and PU data in the training distribution. Since unlabeled data do not have supervision, they assume a particular shift type such as covariate shift (Sakai and Shimizu, 2019). Although they work well when the assumption is satisfied, the shift type is generally difficult or impossible to identify from unlabeled data. One method uses a few PU data in the test distribution, which are often available in practice (Mignone and Pio, 2018). This approach has the

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

potential for improving the performance without any shift assumptions by using PU data in the test distribution. However, the method focuses on a specific task of link prediction on network data and is rather heuristic. For reliable use in a wide variety of tasks, more generic and principled methods are desirable.

In this paper, we propose a distribution shift adaptation method for general PU learning without shift type assumptions that uses a few PU data in the test distribution and PU data in the training distribution. Our method is based on the importance weighting framework, which has been successfully used for distribution shift adaptation on ordinary supervised learning (Sugiyama et al., 2012; Lu et al., 2022). This framework first estimates importance weights that are the (relative) ratio between training and test densities and then learns classifiers by minimizing the importance-weighted empirical training risk. Since it is approximately equivalent to the risk on the test distribution, we can learn classifiers that fit on the test distribution in a principled manner. Although existing importance weighting methods require positive and negative (PN) data in both distributions for adaptation without shift type assumptions (Fang et al., 2020, 2023), we theoretically show that the importance weighting (i.e., the importance weight estimation and classifier learning) can be achieved with only PU data. On the basis of this finding, our loss function is a weighed sum of the empirical risk with PU data in the test distribution and the importance-weighted empirical risk with PU data in the training distribution.

Although we can learn classifiers by minimizing the loss, the importance weight estimation is a difficult task, especially when using complex models such as neural networks or complex data such as image data (Fang et al., 2020; Kato and Teshima, 2021; Rhodes et al., 2020). As a result, the above two-step importance weighting approach often does not work well. To alleviate this problem, following recent works (Fang et al., 2020, 2023), we employ a dynamic learning approach that iterates the importance weight estimation and classifier learning while sharing a neural network for feature extraction. By training the shared feature extractor with simpler classifier learning, the importance weights can be estimated more easily; the classifier learning can be performed without biases by using the estimated importance weights.

Our main contributions are as follows: (1) We formulate the problem of distribution shift adaptation for general PU learning without shift type assumptions. (2) We derive formulas for the importance weighting with only PU data in the training and test distributions. (3) We develop a dynamic approach for the importance weighting based on the derived formulas. (4) We experimentally show that our method outperforms various PU learning methods.

2 RELATED WORK

Many PU learning methods have been proposed (Bekker and Davis, 2020; Fung et al., 2005; Hou et al., 2018; Chen et al., 2020; Luo et al., 2021; Wang et al., 2023). Early methods used some heuristics to identify negative data in unlabeled data (Li and Liu, 2003; Yu et al., 2002). However, the performances of these methods heavily rely on the heuristic strategy and data separability assumption, i.e., PN distributions are separated (Nakajima and Sugiyama, 2023). A representative approach in recent years has been the empirical risk minimization-based approach, which rewrites the risk with PU data and learns classifiers by minimizing the rewritten risk (Du Plessis et al., 2015; Kiryo et al., 2017; Du Plessis et al., 2014; Jiang et al., 2023; Zhao et al., 2022). This approach makes no assumptions about the data separability (Nakajima and Sugiyama, 2023), has good theoretical properties such as consistency (Du Plessis et al., 2015), and has shown the state-of-the-art performance (Jiang et al., 2023). Because of these advantages, the proposed method follows this approach. However, all these methods assume that the training and test distributions are identical. Thus, they are inappropriate for our setting.

Distribution shift adaptation or domain adaptation methods have been proposed to mitigate the gap between the training and test distributions (Pan and Yang, 2009; Wang and Deng, 2018; Farahani et al., 2021). They usually require PN data in the training distribution, which is different to our setting. One representative approach is to learn invariant features to mitigate the distribution gap (Ganin and Lempitsky, 2015; Motiian et al., 2017; Saito et al., 2018). Although this approach is promising, the invariant features often do not improve the performance since they do not explicitly minimize the test risk (Zhao et al., 2019). Another representative approach is the importance weighting that can adapt to the shift by explicitly minimizing the test risk (Lu et al., 2022). Although many two-step methods with importance weighting have been proposed (Kanamori et al., 2009; Yamada et al., 2013; Sugiyama et al., 2012), they often do not work well for complex models and data (Fang et al., 2020). To overcome this difficulty, Fang et al. (2020; 2023) have recently proposed a dynamic approach that iterates the importance weight estimation and classifier learning, which enables the importance weighting to work well even in such difficult cases. Our method is a PU learning extension of this dynamic importance weighting.

Several distribution shift adaptation methods for PU learning have been proposed (Sakai and Shimizu, 2019; Nakajima and Sugiyama, 2023; Hammoudeh and Lowd, 2020). Although most methods that use unlabeled data in the test distribution assume the shift type for the adaptation such as covariate shift (Sakai and Shimizu, 2019), class-prior shift (Nakajima and Sugiyama, 2023), and positive data shift (Hammoudeh and Lowd, 2020), the shift type is generally difficult to identify from unlabeled data in practice. The proposed method does not require such assumptions because it uses a few PU data in the test distribution. Although one invariant feature approach that uses PU data in the test distribution has been proposed (Sonntag et al., 2022), it requires PN data in the training distribution. One method uses PU data in both training and test distributions like this study (Mignone and Pio, 2018). However, it is designed for a specific task on link prediction of network data and is rather heuristic. In contrast, the proposed method is general and based on the principled importance weighting framework.

3 Preliminary

We briefly review PU learning based on the empirical risk minimization (Du Plessis et al., 2015; Kiryo et al., 2017). Let $\mathbf{x} \in \mathcal{X}$ and $y \in \{\pm 1\}$ be the input and output random variables, where $\mathcal{X}$ is the input space. Let $p(\mathbf{x}, y)$ be the joint density, $p^{\rm p}(\mathbf{x}) := p(\mathbf{x}|y=+1)$ and $p^{\rm n}(\mathbf{x}) := p(\mathbf{x}|y=-1)$ be the positive and negative-conditional densities, $p(\mathbf{x}) = \pi p^{\rm p}(\mathbf{x}) + (1-\pi)p^{\rm n}(\mathbf{x})$ be the marginal density, and $\pi := p(y=+1)$ be the positive class-prior. $f : \mathcal{X} \to \mathbb{R}$ is a decision function and the predicted label is obtained by $y = \text{sign}(f(\mathbf{x}))$, where $\text{sign}(\cdot)$ is a sign function. Let $\ell : \mathbb{R} \times \{\pm 1\} \to \mathbb{R}_{\geq 0}$ be the loss function, such that value $\ell(t, y)$ means the discrepancy between predicting output t and ground truth label y.

We can learn decision function f by minimizing the following expected test error, known as the risk:

$$R(f) := \mathbb{E}_{p(\mathbf{x},y)}[\ell(f(\mathbf{x}), y)] \\ = \pi \mathbb{E}_{p^{\rm p}(\mathbf{x})}[\ell(f(\mathbf{x}), +1)] + (1-\pi)\mathbb{E}_{p^{\rm n}(\mathbf{x})}[\ell(f(\mathbf{x}), -1)], \quad (1)$$

where $\mathbb{E}_{p(z)}$ is an expectation over $p(z)$. This risk depends on $p^{\rm n}(\mathbf{x})$, which is inaccessible in PU learning. However, since $(1-\pi)p^{\rm n}(\mathbf{x}) = p(\mathbf{x}) - \pi p^{\rm p}(\mathbf{x})$ from the definition of $p(\mathbf{x})$, we have

$$(1-\pi)\mathbb{E}_{p^{\rm n}(\mathbf{x})}[\ell(f(\mathbf{x}), -1)] \\ = \mathbb{E}_{p(\mathbf{x})}[\ell(f(\mathbf{x}), -1)] - \pi \mathbb{E}_{p^{\rm p}(\mathbf{x})}[\ell(f(\mathbf{x}), -1)]. \quad (2)$$

Plugging it into the second term of Eq. (1), we obtain

$$R(f) = \pi \mathbb{E}_{p^{\rm p}(\mathbf{x})}[\ell(f(\mathbf{x}), +1)] \\ + \mathbb{E}_{p(\mathbf{x})}[\ell(f(\mathbf{x}), -1)] - \pi \mathbb{E}_{p^{\rm p}(\mathbf{x})}[\ell(f(\mathbf{x}), -1)]. \quad (3)$$

Since this risk does not depend on $p^{\rm n}(\mathbf{x})$, we approximate it with PU data. We call this risk a PU risk.

Although Eq. (2) is non-negative, when highly expressive models such as neural networks are used, its empirical estimate often takes negative values, and it causes serious overfitting (Kiryo et al., 2017). To avoid this, the non-negative risk correction such as the absolute value correction (Hammoudeh and Lowd, 2020; Lu et al., 2020) is commonly applied to the empirical estimate in PU learning. Specifically, given $N^{\rm p}$ positive instances $\{\mathbf{x}_n^{\rm p}\}_{n=1}^{N^{\rm p}}$ and N unlabeled instances $\{\mathbf{x}_n\}_{n=1}^{N}$, the corrected empirical estimate of the PU risk in Eq. (3) is described as

$$\hat{R}(f) = \frac{\pi}{N^{\rm p}} \sum_{n=1}^{N^{\rm p}} \ell(f(\mathbf{x}_n^{\rm p}), +1) \\ + \left| \frac{1}{N} \sum_{n=1}^{N} \ell(f(\mathbf{x}_n), -1) - \frac{\pi}{N^{\rm p}} \sum_{n=1}^{N^{\rm p}} \ell(f(\mathbf{x}_n^{\rm p}), -1) \right|. \quad (4)$$

The absolute value function $|\cdot|$ in the second term can prevent over-fitting by penalizing the risk for being negative. When large PU data are available (i.e., $N^{\rm p}$ and N are large), we can obtain an accurate binary classifier by minimizing this empirical PU risk.

4 PROPOSED METHOD

In this section, we formulate our problem setting (subsection 4.1). Then, we derive an importance-weighted empirical PU risk (subsection 4.2) and an importance weight estimation method with PU data (subsection 4.3). Lastly, we explain our loss function for classifier learning with the derived importance-weighted PU risk and its training procedure (subsection 4.4).

4.1 Problem Setting

Suppose that we are given a set of positive instances $X_{\rm tr}^{\rm p}$ and a set of unlabeled instances $X_{\rm tr}$ drawn from the training distribution:

$$X_{\rm tr}^{\rm p} = \{\mathbf{x}_{{\rm tr},n}^{\rm p}\}_{n=1}^{N_{\rm tr}^{\rm p}} \sim p_{\rm tr}^{\rm p}(\mathbf{x}) := p_{\rm tr}(\mathbf{x}|y=+1), \quad (5)$$

$$X_{\rm tr} = \{\mathbf{x}_{{\rm tr},n}\}_{n=1}^{N_{\rm tr}} \sim p_{\rm tr}(\mathbf{x}) = \pi_{\rm tr} p_{\rm tr}^{\rm p}(\mathbf{x}) + (1-\pi_{\rm tr}) p_{\rm tr}^{\rm n}(\mathbf{x}), \quad (6)$$

where $p_{\rm tr}(\mathbf{x})$ is the marginal density of the training distribution, $p_{\rm tr}^{\rm p}(\mathbf{x})$ and $p_{\rm tr}^{\rm n}(\mathbf{x}) := p_{\rm tr}(\mathbf{x}|y=-1)$ are positive and negative-conditional densities of the training distribution, respectively, and $\pi_{\rm tr} := p_{\rm tr}(y=+1)$ is the positive class-prior. Similarly, suppose that we are also given a set of positive instances $X_{\rm te}^{\rm p}$ and a set of unlabeled instances $X_{\rm te}$ drawn from the test distribution:

$$X_{\rm te}^{\rm p} = \{\mathbf{x}_{{\rm te},n}^{\rm p}\}_{n=1}^{N_{\rm te}^{\rm p}} \sim p_{\rm te}^{\rm p}(\mathbf{x}) := p_{\rm te}(\mathbf{x}|y=+1), \quad (7)$$

$$X_{\rm te} = \{\mathbf{x}_{{\rm te},n}\}_{n=1}^{N_{\rm te}} \sim p_{\rm te}(\mathbf{x}) = \pi_{\rm te} p_{\rm te}^{\rm p}(\mathbf{x}) + (1-\pi_{\rm te}) p_{\rm te}^{\rm n}(\mathbf{x}), \quad (8)$$

where $p_{\rm te}^{\rm n}(\mathbf{x}) := p_{\rm te}(\mathbf{x}|y=-1)$ and $\pi_{\rm te} := p_{\rm te}(y=+1)$. We assume that PU data in the test distribution are much smaller than those in the training distribution, i.e., $N_{\rm te}^{\rm p} + N_{\rm te} \ll N_{\rm tr}^{\rm p} + N_{\rm tr}$. Although we assume that class-priors $\pi_{\rm tr}$ and $\pi_{\rm te}$ are known in this paper as in (Du Plessis et al., 2015; Kiryo et al., 2017), they can be also estimated from PU data when they are unknown (Ramaswamy et al., 2016; Yao et al., 2021).

We consider a situation in which training and test joint distributions are different, $p_{\rm tr}(\mathbf{x}, y) \neq p_{\rm te}(\mathbf{x}, y)$, which is the most general shift form. Our aim is to learn decision function f that accurately classifies test instance $\mathbf{x}$ drawn from $p_{\rm te}(\mathbf{x})$ by using $X_{\rm tr}^{\rm p} \cup X_{\rm tr} \cup X_{\rm te}^{\rm p} \cup X_{\rm te}$.

4.2 Importance-weighted PU Risk

In this subsection, to exploit useful knowledge in the training distribution, we consider approximating the test risk by the importance-weighted empirical risk with PU training data. We first consider the risk on the test distribution:

$$R_{\rm te}(f) := \mathbb{E}_{p_{\rm te}(\mathbf{x},y)}\left[\ell(f(\mathbf{x}), y)\right], \quad (9)$$

which is the target to be minimized for learning f that fits on the test distribution. As shown in previous studies (Fang et al., 2020, 2023), this risk can be rewritten as follows,

$$\begin{aligned} R_{\rm te}(f) &= \mathbb{E}_{p_{\rm te}(\mathbf{x},y)}\left[\ell(f(\mathbf{x}), y)\right] \\ &= \mathbb{E}_{p_{\rm tr}(\mathbf{x},y)}\left[w(\mathbf{x}, y)\ell(f(\mathbf{x}), y)\right] \\ &= \pi_{\rm tr}\mathbb{E}_{p_{\rm tr}^{\rm p}(\mathbf{x})}\left[\ell_w(f(\mathbf{x}), +1)\right] \\ &+ (1-\pi_{\rm tr})\mathbb{E}_{p_{\rm tr}^{\rm n}(\mathbf{x})}\left[\ell_w(f(\mathbf{x}), -1)\right] =: R_{\rm tr}^{\rm w}(f), \end{aligned} \quad (10)$$

where $w(\mathbf{x}, y) := \frac{p_{\rm te}(\mathbf{x},y)}{p_{\rm tr}(\mathbf{x},y)}$ is referred to as the *importance weight* and we set

$$\ell_w(\mathbf{x}, y) := w(\mathbf{x}, y)\ell(f(\mathbf{x}), y). \quad (11)$$

This equation implies that the risk on the test distribution can be represented as the importance-weighted risk on the training distribution. Although this risk depends on negative density $p_{\rm tr}^{\rm n}(\mathbf{x})$, by applying the same procedure in Eqs. (1, 2, 3), it can be rewritten without the negative density:

$$\begin{aligned} R_{\rm tr}^{\rm w}(f) &= \pi_{\rm tr}\mathbb{E}_{p_{\rm tr}^{\rm p}(\mathbf{x})}\left[\ell_w(f(\mathbf{x}), +1)\right] \\ &+ \mathbb{E}_{p_{\rm tr}(\mathbf{x})}\left[\ell_w(f(\mathbf{x}), -1)\right] - \pi_{\rm tr}\mathbb{E}_{p_{\rm tr}^{\rm p}(\mathbf{x})}\left[\ell_w(f(\mathbf{x}), -1)\right]. \end{aligned} \quad (12)$$

The empirical estimate of this risk with PU data in the training distribution $X_{\rm tr}^{\rm p} \cup X_{\rm tr}$ is represented as

$$\begin{aligned} \hat{R}_{\rm tr}^{\rm w}(f) &= \frac{\pi_{\rm tr}}{N_{\rm tr}^{\rm p}}\sum_{n=1}^{N_{\rm tr}^{\rm p}} \ell_w(f(\mathbf{x}_{{\rm tr},n}^{\rm p}), +1) \\ &+ \left|\frac{1}{N_{\rm tr}}\sum_{n=1}^{N_{\rm tr}} \ell_w(f(\mathbf{x}_{{\rm tr},n}), -1) - \frac{\pi_{\rm tr}}{N_{\rm tr}^{\rm p}}\sum_{n=1}^{N_{\rm tr}^{\rm p}} \ell_w(f(\mathbf{x}_{{\rm tr},n}^{\rm p}), -1)\right|, \end{aligned} \quad (13)$$

where we used the absolute value function to prevent overfitting.

4.3 Importance Weight Estimation with PU Data

In Eq (13), importance weight $w(\mathbf{x}, y) = \frac{p_{\rm te}(\mathbf{x},y)}{p_{\rm tr}(\mathbf{x},y)}$ remains unknown. In this subsection, we consider estimating it with PU data. A common way for estimating importance weights is to use density-ratio estimation methods that directly estimate the ratio between training and test densities from data without density estimation (Sugiyama et al., 2012; Kanamori et al., 2009; Huang et al., 2006). However, they are often unstable since $w(\mathbf{x}, y)$ is unbounded, i.e., it takes extremely larger values around a low-density region of training data (Yamada et al., 2013; Kumagai et al., 2021). To evade this problem, instead of $w(\mathbf{x}, y)$, we use the following relative density-ratio as importance weight,

$$w_\alpha(\mathbf{x}, y) := \frac{p_{\rm te}(\mathbf{x}, y)}{\alpha p_{\rm te}(\mathbf{x}, y) + (1-\alpha)p_{\rm tr}(\mathbf{x}, y)}, \quad (14)$$

where $\alpha \in [0, 1]$ is a hyperparameter (Yamada et al., 2013). $w_\alpha(\mathbf{x}, y)$ is always bounded above by $1/\alpha$, and it coincides with $w(\mathbf{x}, y)$ when $\alpha = 0$. Thus, $w_\alpha(\mathbf{x}, y)$ is a bounded extension of $w(\mathbf{x}, y)$. The importance weighting with the relative density-ratio has shown excellent performance (Yamada et al., 2013; Sakai and Shimizu, 2019).

Let $m(\mathbf{x}, y) \in [0, 1/\alpha]$ be a model such as a neural network for $w_\alpha(\mathbf{x}, y)$. Parameters of $m(\mathbf{x}, y)$ are determined so that the expected squared error between true importance weight $w_\alpha(\mathbf{x}, y)$ and $m(\mathbf{x}, y)$ is minimized as in previous studies (Yamada et al., 2013):

$$\begin{aligned} J(m) &:= \mathbb{E}_{p_\alpha(\mathbf{x},y)}\left[(w_\alpha(\mathbf{x}, y) - m(\mathbf{x}, y))^2\right] \\ &= \mathbb{E}_{p_{\rm te}(\mathbf{x},y)}\left[\alpha m(\mathbf{x}, y)^2 - 2m(\mathbf{x}, y)\right] \\ &+ (1-\alpha)\mathbb{E}_{p_{\rm tr}(\mathbf{x},y)}\left[m(\mathbf{x}, y)^2\right] + C, \end{aligned} \quad (15)$$

where $p_\alpha(\mathbf{x}, y) := \alpha p_{\rm te}(\mathbf{x}, y) + (1-\alpha)p_{\rm tr}(\mathbf{x}, y)$ and C is a constant term that does not depend on m. Letting

$M(\mathbf{x}, y) := \alpha m(\mathbf{x}, y)^2 - 2m(\mathbf{x}, y)$, we have

$$\begin{aligned} J(m) &= \pi_{\rm te}\mathbb{E}_{p^{\rm p}_{\rm te}(\mathbf{x})}\left[M(\mathbf{x}, +1)\right] \\ &+ (1-\pi_{\rm te})\mathbb{E}_{p^{\rm n}_{\rm te}(\mathbf{x})}\left[M(\mathbf{x}, -1)\right] \\ &+ \pi_{\rm tr}(1-\alpha)\mathbb{E}_{p^{\rm p}_{\rm tr}(\mathbf{x})}\left[m(\mathbf{x}, +1)^2\right] \\ &+ (1-\pi_{\rm tr})(1-\alpha)\mathbb{E}_{p^{\rm n}_{\rm tr}(\mathbf{x})}\left[m(\mathbf{x}, -1)^2\right] + C. \end{aligned} \quad (16)$$

As before, the second and fourth terms that depend on negative densities are rewritten as

$$\begin{aligned} &(1-\pi_{\rm te})\mathbb{E}_{p^{\rm n}_{\rm te}(\mathbf{x})}\left[M(\mathbf{x}, -1)\right] = \\ &\mathbb{E}_{p_{\rm te}(\mathbf{x})}\left[M(\mathbf{x}, -1)\right] - \pi_{\rm te}\mathbb{E}_{p^{\rm p}_{\rm te}(\mathbf{x})}\left[M(\mathbf{x}, -1)\right], \quad (17) \\ &(1-\pi_{\rm tr})\mathbb{E}_{p^{\rm n}_{\rm tr}(\mathbf{x})}\left[m(\mathbf{x}, -1)^2\right] = \\ &\mathbb{E}_{p_{\rm tr}(\mathbf{x})}\left[m(\mathbf{x}, -1)^2\right] - \pi_{\rm tr}\mathbb{E}_{p^{\rm p}_{\rm tr}(\mathbf{x})}\left[m(\mathbf{x}, -1)^2\right], \quad (18) \end{aligned}$$

Since $M(\mathbf{x}, y) = \alpha m(\mathbf{x}, y)^2 - 2m(\mathbf{x}, y)$, $M(\mathbf{x}, y)$ is greater than $-1/\alpha$ for all $\mathbf{x}$ and y; Eq. (17) is greater than $-(1-\pi_{\rm te})/\alpha$. However, as in the risk, the empirical estimate of Eq. (17) can take smaller values and it may causes serious overfitting. To avoid this, we use function $F(z) := |z - c| + c$ $(c, z \in \mathbb{R})$ as the correction function where c is the lower bound of the loss (Ishida et al., 2020). When $z \geq c$, $F(z)$ returns z. When $z < c$, $F(z) = 2c - z > c$. Thus, this function can penalize the loss for being smaller than c. Similarly, although Eq. (18) is non-negative, its empirical estimate can take negative values. We correct this by applying the absolute value function $F(z) = |z|$ as in the risk. As a result, our loss function for the importance weight estimation becomes

$$\begin{aligned} \hat{J}(m) &:= \frac{\pi_{\rm te}}{N^{\rm p}_{\rm te}}\sum_{n=1}^{N^{\rm p}_{\rm te}} M(\mathbf{x}^{\rm p}_{{\rm te},n}, +1) \\ &+ \left|\frac{1}{N_{\rm te}}\sum_{n=1}^{N_{\rm te}} M(\mathbf{x}_{{\rm te},n}, -1) - \frac{\pi_{\rm te}}{N^{\rm p}_{\rm te}}\sum_{n=1}^{N^{\rm p}_{\rm te}} M(\mathbf{x}^{\rm p}_{{\rm te},n}, -1) + \frac{1-\pi_{\rm te}}{\alpha}\right| \\ &+ \frac{\pi_{\rm tr}(1-\alpha)}{N^{\rm p}_{\rm tr}}\sum_{n=1}^{N^{\rm p}_{\rm tr}} m(\mathbf{x}^{\rm p}_{{\rm tr},n}, +1)^2 \\ &+ (1-\alpha)\left|\frac{1}{N_{\rm tr}}\sum_{n=1}^{N_{\rm tr}} m(\mathbf{x}_{{\rm tr},n}, -1)^2 - \frac{\pi_{\rm tr}}{N^{\rm p}_{\rm tr}}\sum_{n=1}^{N^{\rm p}_{\rm tr}} m(\mathbf{x}^{\rm p}_{{\rm tr},n}, -1)^2\right|, \end{aligned} \quad (19)$$

where we omitted constant terms that do not depend m. By minimizing this loss, we can estimate $m(\mathbf{x}, y)$.

4.4 Classifier Loss Function and Training Procedure

In this subsection, we describe our loss function for learning f. Specifically, our loss function is a weighted sum of the empirical PU risk on the test distribution and the importance-weighted empirical PU risk on the training distribution:

$$\hat{L}(f, m) := \beta\hat{R}_{\rm te}(f) + (1-\beta)\hat{R}^{\rm w}_{\rm tr}(f, m), \quad (20)$$

where $\beta \in (0, 1)$ is a weighting hyperparameter, and $\hat{R}_{\rm te}(f)$ is the empirical PU risk with $X^{\rm p}_{\rm te} \cup X_{\rm te}$,

$$\begin{aligned} \hat{R}_{\rm te}(f) &= \frac{\pi_{\rm te}}{N^{\rm p}_{\rm te}}\sum_{n=1}^{N^{\rm p}_{\rm te}} \ell(f(\mathbf{x}^{\rm p}_{{\rm te},n}), +1) \\ &+ \left|\frac{1}{N_{\rm te}}\sum_{n=1}^{N_{\rm te}} \ell(f(\mathbf{x}_{{\rm te},n}), -1) - \frac{\pi_{\rm te}}{N^{\rm p}_{\rm te}}\sum_{n=1}^{N^{\rm p}_{\rm te}} \ell(f(\mathbf{x}^{\rm p}_{{\rm te},n}), -1)\right|. \end{aligned} \quad (21)$$

Here, we use Eq. (4) to derive Eq. (21), and explicitly describe the dependency of the importance weight model m in $\hat{R}^{\rm w}_{\rm tr}$ for clarity.

To calculate $\hat{R}^{\rm w}_{\rm tr}(f, m)$, the traditional approach is to first estimate the importance weights and then use them to calculate the weighted risk (Yamada et al., 2013; Sugiyama et al., 2012; Kanamori et al., 2009). However, in this two-step approach, errors in the importance weight estimation directly propagates to the subsequent importance-weighted risk calculation, which degrades the performance of the learned classifiers (Fang et al., 2020; Zhang et al., 2020). To alleviate this, Fang et al. (2020; 2023) recently proposed the dynamic approach that iterates between the importance weight estimation and classifier learning for ordinary supervised learning.

The proposed method follows this dynamic approach. Specifically, we assume the following neural networks for modeling classifiers and importance weights,

$$f(\mathbf{x}) := u(h(\mathbf{x})), \quad (22)$$
$$m(\mathbf{x}, y) := v([h(\mathbf{x}), y]), \quad (23)$$

where $h : \mathcal{X} \to \mathbb{R}^K$, $u : \mathbb{R}^K \to \mathbb{R}$, and $v : \mathbb{R}^{K+2} \to \mathbb{R}$ are neural networks for feature extraction, classifiers, and importance weights, respectively. Here, we assume that label $y \in \{-1, +1\}$ is represented as one-hot encoding and $[\cdot, \cdot]$ is a concatenation. By sharing feature extractor h for the weight estimation and classification, we can effectively perform the importance weighting.

Algorithm 1 shows the training procedure of our method with stochastic gradient descent methods. We first randomly sample PU data from the training and test distributions (Lines 2–3). Then, we calculate the loss in Eq. (19) for the importance weight estimation (Line 4) and update parameters of v with the gradient of the loss fixing feature extractor h (Line 5). We fixed h to avoid overfitting as in (Fang et al., 2020). Next,

Algorithm 1 Training procedure of the proposed method

```
Require: PU data in the training and test distributions $X^{\rm p}_{\rm tr} \cup X_{\rm tr} \cup X^{\rm p}_{\rm te} \cup X_{\rm te}$, mini-batch sizes for the training and test distributions $(B_{\rm tr}, B_{\rm te})$, positive class-priors $(\pi_{\rm tr}, \pi_{\rm te})$, relative parameter $\alpha$, and weighting parameter $\beta$.
Ensure: Parameters of neural networks $h$, $u$, and $v$.
1: repeat
2:     Sample PU data with size $B_{\rm tr}$ form $X^{\rm p}_{\rm tr} \cup X_{\rm tr}$
3:     Sample PU data with size $B_{\rm te}$ form $X^{\rm p}_{\rm te} \cup X_{\rm te}$
       {Importance weight estimation}
4:     Calculate loss in Eq. (19) on the sampled data
5:     Update parameters of $v$ with the gradient of the loss fixing feature extractor $h$
       {Classifier learning}
6:     Calculate the loss in Eq. (20) on the sampled data with current importance weights
7:     Update classifier parameters of $u$ and $h$ with the gradient of the loss fixing the importance weights
8: until End condition is satisfied;
```

by using the current estimated importance weights, we calculate the loss in Eq. (20) for classifier learning (Line 6). We update parameters of classifier u and h by using the gradient of the loss (Line 7). In this step, we fixed the importance weights to avoid learning a meaningless model, $m(\mathbf{x}, y) = 0$ for all $\mathbf{x}$ and y.

5 EXPERIMENTS

5.1 Data

We utilized four widely used real-world datasets: MNIST (LeCun et al., 1998), FashionMNIST (Xiao et al., 2017) (FMNIST), CIFAR10 (Krizhevsky et al., 2009), and DIABETES (Gardner et al., 2023). MNIST consists of hand-written images of 10 digits. Each image is represented by grayscale with 28×28 pixels. FMNIST consists of images of 10 fashion categories where each image is represented by gray scale with 28×28 pixels. CIFAR10 consists of 32×32 RGB images of 10 animal and vehicle categories. DIABETES is a tabular dataset used for distribution adaptation studies (Gardner et al., 2023), whose task is to predict whether the respondent has diabetes or not. Each respondent is represented as 142 dimensional features. This dataset has a distribution shift where data from the training and test distributions are "white non-hispanic" and other racial groups, respectively.

For MNIST, FMNIST, and CIFAR10, we consider two types of distribution shifts: The first one is an input-output relation shift where $p(y|\mathbf{x})$ varies, but the support of input $\mathbf{x}$ does not change between the training and testing phases. This simulates a situation where the user's preferences in, for example, personalized adverting change between the training and testing phases. The second one is a support shift (Fang et al., 2023), where the inputs' support varies between the training and testing phases. This simulates a situation where new types of ads presented to the user appear, for example. Note that no methods use the knowledge that such shifts have occurred in our experiments.

We describe how to construct the data in the case of the input-output relation shift. For MNIST, we used even digits as negative and odd digits as positive in the test distribution. We flipped digits (0 and 2) and (1 and 3) in the training distribution (e.g., data with the digits 0, 2, 5, 7, and 9 are positive in the training distribution). For FMNIST, following the study (Xie et al., 2024), we used upper garments (0, 2, 3, 4, and 6) as negative and the others as positive, where numbers in parentheses represent class labels, in the test distribution. We flipped the class labels (0 and 2) and (1 and 5) in the training distribution. For CIFAR10, we used the animal categories (2, 3, 4, 5, 6 and 7) as negative and the vehicles as positive in the test distribution. We flipped the class labels (2 and 3) and (0 and 1) in the training distribution. Next, we describe how to construct the data in the case of the support shift. For MNIST, we used digits 0, 2, and 4 (4, 6, and 8) as negative and digits 1, 3, and 5 (5, 7, and 9) as positive in the training (test) distribution. For FMNIST, we used class labels 0, 2, and 3 (3, 4, and 6) as negative and class labels 1, 5, and 7 (7, 8, and 9) as positive in the training (test) distribution. For CIFAR10, we used class labels 2, 3, 4, and 5 (4, 5, 6, and 7) as negative and class labels 0, 1, and 8 (1, 8, and 9) as positive in the training (test) distribution [1].

For training in each dataset, we used $1,000$ positive and $5,000$ unlabeled data in the training distribution and three types of PU data with different numbers in the test distribution: $(N^{\rm p}_{\rm te}, N_{\rm te}) = (10, 50), (20, 100)$, and $(40, 200)$. In addition, we used 20 positive and 100 unlabeled data in the test distribution for validation. The positive class-priors on both the training and test distributions, $\pi_{\rm tr}$ and $\pi_{\rm te}$, were set to 0.5. We used $2,500$ positive and $2,500$ negative data in the test distribution as test data for evaluation. Training,

[1] These settings are similar to the covariate shift, where the input distribution can vary $p_{\rm te}(\mathbf{x}) \neq p_{\rm tr}(\mathbf{x})$, but the label distribution given input remains unchanged $p_{\rm te}(y|\mathbf{x}) = p_{\rm tr}(y|\mathbf{x})$ (Shimodaira, 2000). In covariate shift studies, it is usually assumed that the input support of the test distribution is contained within that of the training distribution (Zhang et al., 2020; Sugiyama et al., 2008). However, this assumption may not be satisfied in practice (Fang et al., 2023). Thus, we considered a more challenging setting.

Table 1: Average test accuracies over different numbers of PU data in the test distribution: $(N_{\rm te}^{\rm p}, N_{\rm te}) = (10, 50), (20, 100)$, and $(40, 200)$. In the Shift column, 'IO' and 'S' represent the input-output relation shift and support shift, respectively. Values in bold are not statistically different at the 5% level from the best performing method in each row according to a paired t-test.

Data	Shift	Ours	tePU	trPU	ftPU	mtPU	mtsPU	DAPU	UDAPU	GIW
MNIST	IO	**0.7493**	0.7371	0.5628	0.6807	0.7115	0.7152	0.7223	0.5499	**0.7369**
	S	0.7339	0.7295	0.5760	0.6808	**0.7482**	0.7383	**0.7536**	0.6782	**0.7459**
FMNIST	IO	**0.9153**	0.8876	0.5851	0.7348	0.8981	0.8972	0.8937	0.7106	0.8728
	S	**0.9656**	**0.9609**	0.8841	0.9175	0.9547	0.9489	0.9480	0.9398	**0.9622**
CIFAR10	IO	**0.7081**	0.6557	0.6603	0.6699	0.6718	**0.7033**	**0.6967**	0.6747	0.6970
	S	**0.8572**	0.6633	0.8434	0.8316	0.8092	0.8388	0.8452	0.8477	0.8331
DIABETES		**0.7242**	0.6016	0.7194	0.7163	**0.7142**	**0.7206**	0.7176	**0.7206**	0.6863

validation, and test datasets did not overlap. We conducted 10 experiments for each type of the number of PU data in the test distribution while changing the random seeds and evaluated mean test accuracy.

5.2 Comparison Methods

We compared the proposed method with eight methods: PU learning method on the test distribution (tePU), PU learning method on the training distribution (trPU), fine-tuning method (ftPU), multi-task learning method (mtPU), multi-task learning method with a single neural network (mtsPU), domain adaptation method for PU learning (DAPU), unsupervised domain adaptation method for PU learning (UDAPU), and generalized dynamic importance weighting method with pseudo PN data (GIW). All methods including the proposed method used neural network-based classifiers and the absolute value correction to prevent overfitting (Hammoudeh and Lowd, 2020; Lu et al., 2020).

tePU and trPU learn the classifier by minimizing the non-negative empirical risk (in Eq. (4)) with PU data in the test and training distribution, respectively (Kiryo et al., 2017). ftPU fine-tunes the classifier pretrained by trPU by minimizing the non-negative empirical risk with PU data in the test distribution. mtPU and mtsPU learn the classifier by minimizing the weighted sum of the non-negative empirical PU risks on the training and test distributions. mtPU used a two-head neural network to deal with the difference between the training and test distributions. mtsPU used the single neural network as in the proposed method. The difference between the proposed method and mtsPU is whether or not the importance weights are used in Eq. (20). DAPU is a domain adaptation method for PU learning, which minimizes the weighted sum of the non-negative empirical PU risks on both distributions while minimizing the feature discrepancy to mitigate the distribution gap as in (Sonntag et al., 2022). We used the maximum mean discrepancy (MMD) loss (Li et al., 2015) to minimize the feature discrepancy, which is commonly used in domain adaptation studies (Farahani et al., 2021). UDAPU uses PU data in the training distribution and unlabeled data in the test distribution. It minimizes the non-negative empirical PU risk on the training distribution while minimizing the MMD loss to obtain invariant features. GIW applies the generalized dynamic importance weighting method, which is designed for ordinary supervised learning (Fang et al., 2020, 2023), to pseudo PN data in both distributions. The pseudo PN data were created by applying tePU and trPU to PU data in both distributions beforehand. The difference between the proposed method and GIW is whether the importance weighting and PU learning are performed simultaneously.

For all methods, the sigmoid loss was used for loss ℓ as in the previous study (Kiryo et al., 2017). The empirical PU risk with validation data on the test distribution was used to select hyperparameters and early-stopping to mitigate overfitting. All methods were implemented using Pytorch (Paszke et al., 2017), and all experiments were conducted on a Linux server with an Intel Xeon CPU and A100 GPU. The details of neural network architectures and hyperparameters are described in Sections C and D.

5.3 Results

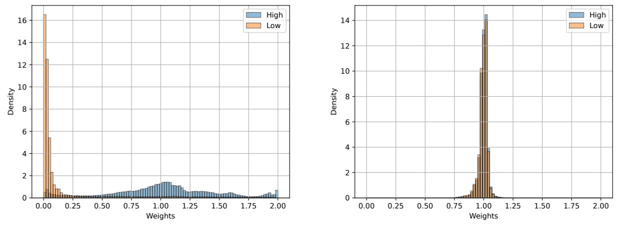
Table 1 shows the average test accuracies over different numbers of PU data in the test distribution (full results including the standard deviations are reported in Section E.4). The proposed method performed the best or comparably to it in almost all cases (6 out of 7 cases). tePU and trPU performed worse than the proposed method because tePU used only a small number of PU data in the test distribution and trPU used only PU data in the training distribution, which is different to the test distribution. ftPU, which is

Table 2: Ablation study: average test accuracies over different numbers of PU data in the test distribution: $(N_{\rm te}^{\rm p}, N_{\rm te}) = (10, 50), (20, 100)$, and $(40, 200)$.

Data	Shift	Ours	2step	w/o IW-C	w/o CL-C	w/o IWCL-C
MNIST	IO	**0.7493**	0.7195	**0.7479**	0.6433	0.6506
	S	0.7339	0.7378	**0.7588**	0.6703	0.6728
FMNIST	IO	**0.9153**	0.8998	**0.9160**	0.8149	0.8130
	S	**0.9656**	**0.9625**	0.9567	0.8786	0.8816
CIFAR10	IO	**0.7081**	**0.7136**	**0.7073**	0.6900	0.6945
	S	**0.8572**	0.8378	0.8420	0.8322	0.8208
DIABETES		**0.7242**	**0.7211**	**0.7230**	**0.7220**	0.7101

the fine-tuning method for PU learning, did not work well. This result suggests the difficulty of fine-tuning with PU data. mtPU and mtsPU outperformed these methods by learning with PU data in both distributions simultaneously. However, since their losses are not designed for the test distribution, they performed worse than the proposed method. Although DAPU and UDAPU aim to mitigate the distribution gap by learning invariant features, they did not work well, which suggests that the invariant features do not necessarily guarantee better performance in the test distribution. GIW, which used pseudo PN data for the dynamic importance weighting, performed worse than the proposed method. Since the pseudo PN data in both distributions were obtained by applying tePU and trPU to the PU data in advance, they contained label noise. This noise negatively impacted the subsequent importance weighting. In contrast, since the proposed method performs the PU learning and importance weighting simultaneously, it robustly worked well. The result with each number of PU data in the test distribution was described in Section E.1. The proposed method worked well on each case.

Table 2 shows the results of ablation studies to investigate the validity of our framework. '2step' is a two-step approach of the proposed method that first performs the importance weight estimation with Eq. (19) and then learns classifiers with the estimated weights by minimizing Eq. (20). 'w/o IW-C' is the proposed method without the absolute value correction for the importance weight estimation in Eq. (19). 'w/o CL-C' is the proposed method without the absolute value correction for classifier learning in Eqs. (21) and (13). 'w/o IWCL-C' is the proposed method without the absolute value correction for both the importance weight estimation and classifier learning. The proposed method performed better than 2step, which indicates the effectiveness of our dynamic importance weighting formulation. Figure 1 shows distributions of importance weights estimated from the proposed method and 2step. Although 2step was not able to estimate the weights correctly, the proposed method was (i.e., the weights of 'High' and 'Low' were large



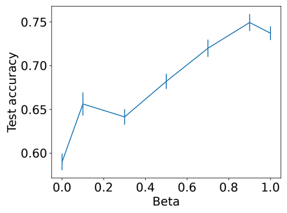
(a) Ours (b) 2step

Figure 1: Importance weight distributions of the proposed method (Ours) and 2step on FMNIST (IO) when $N_{\rm te}^{\rm p} = 40$ and $N_{\rm te} = 200$. 'High' (blue) represents data in the test distribution, and 'Low' (orange) represents data that are not in the test distribution.

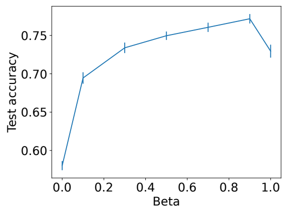
and small, respectively). The proposed method outperformed w/o CL-C and w/o IWCL-C by a large margin, which indicates the effectiveness of the loss correction in classifier learning. In contrast, it slightly outperformed w/o IW-C. Since the importance weights are indirectly related to classification, the effect of the correction for importance weights may be weaker than that for classifier learning. Note that the proposed method without the corrections is also our proposal.

Figure 2 shows the average test accuracies with the standard errors when changing β, which controls the effect of the empirical PU risk on the test distribution and the importance-weighted PU risk on the training distribution in Eq. (20). When $\beta = 1$, it is equivalent to tePU. When $\beta = 0$, it uses only the importance-weighted PU risk for classifier learning. For all datasets, the best values of β were within $(0, 1)$, which indicates the effectiveness of our loss function (the weighted sum of both PU risks). Although the best values are different across datasets, our method was able to select good values using validation data.

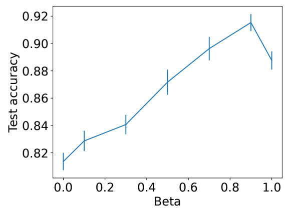
The proposed method needs to know class-priors of the training and test distributions, $\pi_{\rm tr}$ and $\pi_{\rm te}$, for the importance weight estimation and classifier learning. Although we evaluated the proposed method with true class-priors of the datasets in our experiments, there might be some noise in the class-prior in practice. Thus, we investigated how the performance of the proposed method changes when using class-priors that are different to the true ones for training. Figure 3 shows the results. As expected, the method performed better when using the class-priors that were close to true ones $(\pi_{\rm tr}, \pi_{\rm te}) = (0.5, 0.5)$ for all datasets. The difference in the class-prior in the test distribution affected the performance more than that in the class-prior in the training distribution. This is because the class-prior in the test distribution is directly related to classification in the test distribution. In addition, the



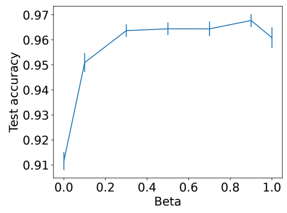
(a) MNIST(IO)



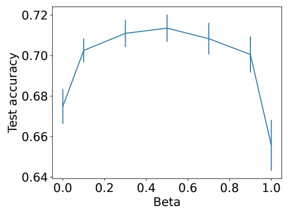
(b) MNIST(S)



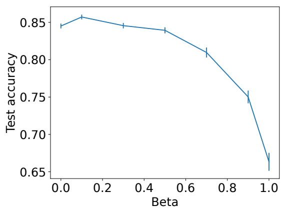
(c) FMNIST(IO)



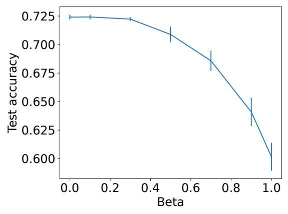
(d) FMNIST(S)



(e) CIFAR10(IO)

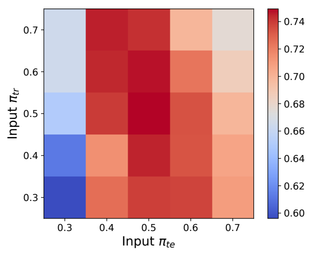


(f) CIFAR10(S)

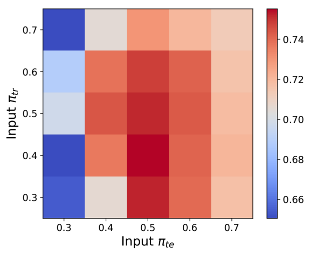


(g) DIABETES

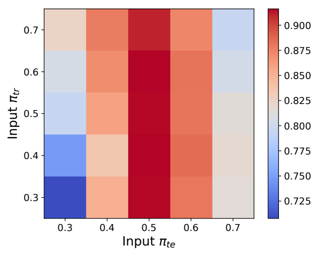
Figure 2: The average test accuracies with the standard errors of the proposed method when changing weighting parameter β.



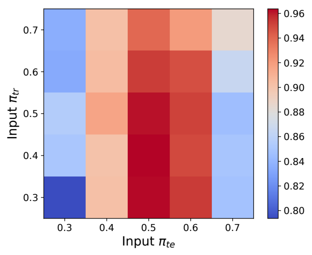
(a) MNIST(IO)



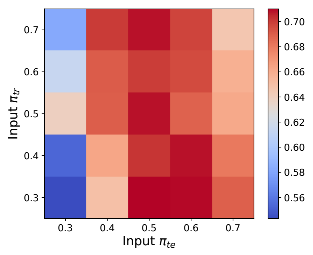
(b) MNIST(S)



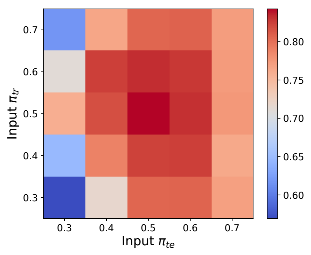
(c) FMNIST(IO)



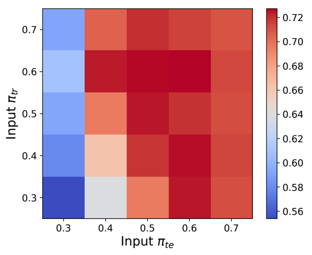
(d) FMNIST(S)



(e) CIFAR10(IO)



(f) CIFAR10(S)



(g) DIABETES

Figure 3: The average test accuracies of the proposed method when changing values of the class-priors used in the importance weighting and classifier learning. The true class-priors of the datasets are $(\pi_{\rm tr}, \pi_{\rm te}) = (0.5, 0.5)$.

proposed method tended to perform better when using large values of the test class prior (e.g., 0.7) than small values (e.g., 0.3). This result is consistent with that in the previous study (Kiryo et al., 2017) and implies that using a large value of the class-prior in the test distribution is safer when it is unknown.

6 CONCLUSION

In this paper, we proposed a distribution shift adaptation method for general PU learning, which uses a few PU data in the test distribution and PU data in the training distribution. Our method can perform adaptation without knowing shift types, which is beneficial in practice. The experiments show that our method outperformed various existing PU learning methods.

References

Bekker, J. and Davis, J. (2020). Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760.

Chen, H., Liu, F., Wang, Y., Zhao, L., and Wu, H. (2020). A variational approach for learning from positive and unlabeled data. *NeurIPS*.

Du Plessis, M., Niu, G., and Sugiyama, M. (2015). Convex formulation for learning from positive and unlabeled data. In *ICML*.

Du Plessis, M. C., Niu, G., and Sugiyama, M. (2014). Analysis of learning from positive and unlabeled data. *NeurIPS*.

Fang, T., Lu, N., Niu, G., and Sugiyama, M. (2020). Rethinking importance weighting for deep learning under distribution shift. *NeurIPS*.

Fang, T., Lu, N., Niu, G., and Sugiyama, M. (2023). Generalizing importance weighting to a universal solver for distribution shift problems. *NeurIPS*.

Farahani, A., Voghoei, S., Rasheed, K., and Arabnia, H. R. (2021). A brief review of domain adaptation. *ICDATA*.

Fung, G. P. C., Yu, J. X., Lu, H., and Yu, P. S. (2005). Text classification without negative examples revisit. *IEEE transactions on Knowledge and Data Engineering*, 18(1):6–20.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *ICML*.

Gardner, J., Popovic, Z., and Schmidt, L. (2023). Benchmarking distribution shift in tabular data with tableshift. *NeurIPS*.

Hammoudeh, Z. and Lowd, D. (2020). Learning from positive and unlabeled data with arbitrary positive shift. *NeurIPS*.

Hido, S., Tsuboi, Y., Kashima, H., Sugiyama, M., and Kanamori, T. (2011). Statistical outlier detection using direct density ratio estimation. *Knowledge and information systems*, 26(2):309–336.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2021). Meta-learning in neural networks: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169.

Hou, M., Chaib-Draa, B., Li, C., and Zhao, Q. (2018). Generative adversarial positive-unlabeled learning. In *IJCAI*.

Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., and Smola, A. (2006). Correcting sample selection bias by unlabeled data. *NeurIPS*.

Ishida, T., Yamane, I., Sakai, T., Niu, G., and Sugiyama, M. (2020). Do we need zero training loss after achieving zero training error? In *ICML*.

Jiang, Y., Xu, Q., Zhao, Y., Yang, Z., Wen, P., Cao, X., and Huang, Q. (2023). Positive-unlabeled learning with label distribution alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kanamori, T., Hido, S., and Sugiyama, M. (2009). A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445.

Kato, M. and Teshima, T. (2021). Non-negative bregman divergence minimization for deep direct density ratio estimation. In *ICML*.

Kingma, D. P. and Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiryo, R., Niu, G., Du Plessis, M. C., and Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. *NeurIPS*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kumagai, A., Iwata, T., and Fujiwara, Y. (2019). Transfer anomaly detection by inferring latent domain representations. *NeurIPS*.

Kumagai, A., Iwata, T., and Fujiwara, Y. (2021). Meta-learning for relative density-ratio estimation. In *NeurIPS*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, W., Guo, Q., and Elkan, C. (2010). A positive and unlabeled learning algorithm for one-class classification of remote-sensing data. *IEEE transactions on geoscience and remote sensing*, 49(2):717–725.

Li, X. and Liu, B. (2003). Learning to classify texts using positive and unlabeled data. In *IJCAI*.

Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *ICML*.

Liu, Y., Qiu, S., Zhang, P., Gong, P., Wang, F., Xue, G., and Ye, J. (2017). Computational drug discovery with dyadic positive-unlabeled learning. In *ICDM*.

Lu, N., Zhang, T., Fang, T., Teshima, T., and Sugiyama, M. (2022). Rethinking importance weighting for transfer learning. In *Federated and Transfer Learning*, pages 185–231. Springer.

Lu, N., Zhang, T., Niu, G., and Sugiyama, M. (2020). Mitigating overfitting in supervised classification from two unlabeled datasets: a consistent risk correction approach. In *AISTATS*.

Luo, C., Zhao, P., Chen, C., Qiao, B., Du, C., Zhang, H., Wu, W., Cai, S., He, B., Rajmohan, S., et al. (2021). Pulns: positive-unlabeled learning with effective negative sample selector. In *AAAI*.

Mignone, P. and Pio, G. (2018). Positive unlabeled link prediction via transfer learning for gene network reconstruction. In *ISMIS*.

Motiian, S., Piccirilli, M., Adjeroh, D. A., and Doretto, G. (2017). Unified deep supervised domain adaptation and generalization. In *ICCV*.

Nakajima, S. and Sugiyama, M. (2023). Positive-unlabeled classification under class-prior shift: a prior-invariant approach based on density ratio estimation. *Machine Learning*, 112(3):889–919.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Ramaswamy, H., Scott, C., and Tewari, A. (2016). Mixture proportion estimation via kernel embeddings of distributions. In *ICML*.

Rhodes, B., Xu, K., and Gutmann, M. U. (2020). Telescoping density-ratio estimation. *NeurIPS*.

Saito, K., Watanabe, K., Ushiku, Y., and Harada, T. (2018). Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*.

Sakai, T. and Shimizu, N. (2019). Covariate shift adaptation on learning from positive and unlabeled data. In *AAAI*.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.

Sonntag, J., Behrens, G., and Schmidt-Thieme, L. (2022). Positive-unlabeled domain adaptation. *arXiv preprint arXiv:2202.05695*.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2012). *Density ratio estimation in machine learning*. Cambridge University Press.

Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., Von Bünau, P., and Kawanabe, M. (2008). Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60:699–746.

Wang, M. and Deng, W. (2018). Deep visual domain adaptation: a survey. *Neurocomputing*, 312:135–153.

Wang, X., Chen, H., Guo, T., and Wang, Y. (2023). Pue: biased positive-unlabeled learning enhancement by causal inference. *NeurIPS*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, Z., Liu, Y., He, H.-Y., Li, M., and Zhou, Z.-H. (2024). Weakly supervised auc optimization: a unified partial auc approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., and Sugiyama, M. (2013). Relative density-ratio estimation for robust distribution comparison. *Neural computation*, 25(5):1324–1370.

Yao, Y., Liu, T., Han, B., Gong, M., Niu, G., Sugiyama, M., and Tao, D. (2021). Rethinking class-prior estimation for positive-unlabeled learning. In *ICLR*.

Yu, H., Han, J., and Chang, K. C.-C. (2002). Pebl: positive example based learning for web page classification using svm. In *SIGKDD*, pages 239–248.

Zhang, T., Yamane, I., Lu, N., and Sugiyama, M. (2020). A one-step approach to covariate shift adaptation. In *ACML*.

Zhao, H., Des Combes, R. T., Zhang, K., and Gordon, G. (2019). On learning invariant representations for domain adaptation. In *ICML*.

Zhao, Y., Xu, Q., Jiang, Y., Wen, P., and Huang, Q. (2022). Dist-pu: positive-unlabeled learning from a label distribution perspective. In *CVPR*.

Zuluaga, M. A., Hush, D., Delgado Leyton, E. J., Hoyos, M. H., and Orkisz, M. (2011). Learning from only positive and unlabeled data to detect lesions in vascular ct images. In *MICCAI*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] We described them in Section 4.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] We described them in Section 4.
   (b) Complete proofs of all theoretical results. [Yes] We described them in Section 4.
   (c) Clear explanations of any assumptions. [Yes] We described them in Section 4.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No] The code is proprietary. We described the details of our experiment settings in Sections 5.1, C, and D.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] We described them in Sections 5.1 and D.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] We described them in captions of Tables and Figures.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] We described them in Section 5.2.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] We described them in 5.1.
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A LIMITATIONS

Although the proposed method worked well in our experiments, if data in the test distribution are very small (e.g., two or three instances), the proposed method will have difficulty performing well since the importance weighting becomes difficult. However, this difficulty is common in other distribution shift adaptation methods. To alleviate this problem, one promising research direction is to combine the proposed method with the meta-learning that can learn how to learn from very small data by using data in multiple related tasks (Hospedales et al., 2021).

B DIFFERENCES FROM (Sakai and Shimizu, 2019)

Importance weighting has already been used in a PU learning method for covariate shift adaptation (Sakai and Shimizu, 2019). Here, we explain the difference between this previous method and the proposed method in detail.

First, the previous method assumes the specific shift type, a covariate shift, where $p_{\rm te}(y|\mathbf{x}) = p_{\rm tr}(y|\mathbf{x})$ and $p_{\rm te}(\mathbf{x}) \neq p_{\rm tr}(\mathbf{x})$ and cannot be applied to other shift types. In contrast, the proposed method does not assume shift types and thus can be applied to any shift types, which is the strong advantage in practice since shift types are generally difficult or impossible to identify.

Second, by assuming the covariate shift, importance weights in the previous method are defined by $w(\mathbf{x}) = p_{\rm te}(\mathbf{x})/p_{\rm tr}(\mathbf{x})$. Since they can be estimated from unlabeled data in the training and test distributions, the previous method uses the conventional density-ratio estimation methods, and new density-ratio estimation methods were not proposed in the previous paper. In contrast, in our paper, importance weights are defined by $w(\mathbf{x}, y) = p_{\rm te}(\mathbf{x}, y)/p_{\rm tr}(\mathbf{x}, y)$. To estimate them from only PU data, we proposed a new density-estimation method in Section 4.3, which is one of the main contributions of our paper.

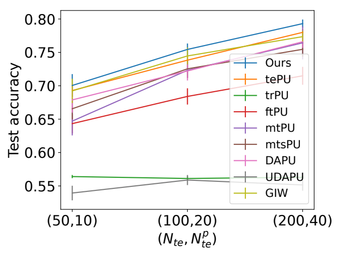
Lastly, the previous method uses the conventional two-step approach for the importance weighting, which is difficult to apply to complex models and data (Fang et al., 2020; Kato and Teshima, 2021; Rhodes et al., 2020). In contrast, the proposed method develops the dynamic approach for the importance weighting to apply to even complex models and data.

C NEURAL NETWORK ARCHITECTURES

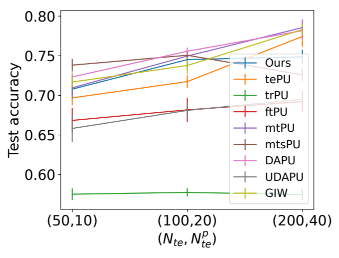
For MNIST, FMNIST, and DIABETES, a three-layered feed-forward neural network with ReLU activation was used for feature extractor h. The number of hidden and output nodes was 128 (32) for MNIST and FMNIST (DIABETES). We used the small node sizes for DIABETES since its feature dimension was small. For CIFAR10, a convolutional neural network, which consisted of two convolutional blocks followed by a two-layered feed-forward neural network, was used for feature extractor h. The first (second) convolutional block comprised a 6 (16) filter 5×5 convolution, the ReLU activation, and a 2×2 max-pooling layer. The numbers of the hidden and output nodes were 120 and 84, respectively. One-layered and two-layered feed-forward neural networks were used for u and v, respectively. For the output activation of v, we used $\frac{1}{\alpha}\sigma(\cdot)$, where σ is a sigmoid function, to match the value range of the relative density-ratio (importance weights). For all comparison methods, the same neural network architecture (i.e., $u(h(\mathbf{x}))$) was used for the classifier. For GIW, the architecture of v was also the same as that of the proposed method. For mtPU, two different classifier heads (i.e., u) were used for the training and test distributions.

D HYPERPARAMETERS

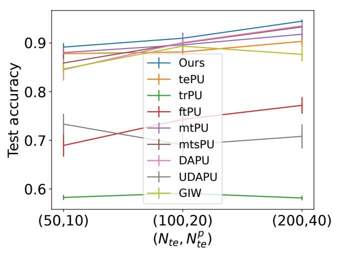
For all methods, the empirical PU risk with validation data on the test distribution was used to select hyperparameters and early-stopping to mitigate overfitting. For the proposed method and GIW, relative parameter α was selected from $\{0.1, 0.5, 0.9\}$. For the proposed method, GIW, mtPU, mtsPU, and DAPU, weighting parameter β was selected from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. Note that the proposed method, GIW, mtPU, and mtsPU are equivalent to tePU when $\beta = 1.0$. For DAPU and UDAPU, the MMD loss was applied to $h(\mathbf{x})$. The number of RBF kernel mixtures was set to 5 as in the previous study (Li et al., 2015). The weighting parameter of the MMD loss was chosen from $\{1, 10^{-1}, 10^{-2}, 10^{-3}\}$. The mini-batch sizes for the training and test distributions, $B_{\rm tr}$ and $B_{\rm te}$, were set to 256. For all methods, we used the Adam optimizer (Kingma and Ba, 2014). We set



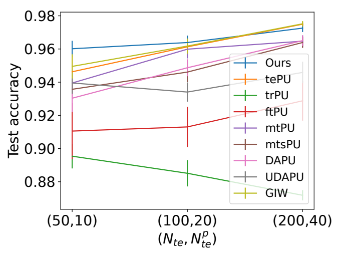
(a) MNIST(IO)



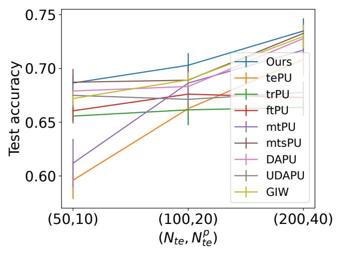
(b) MNIST(S)



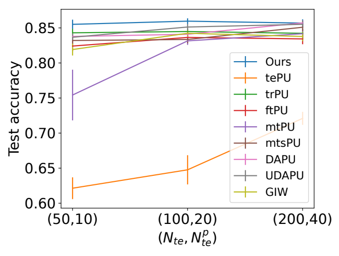
(c) FMNIST(IO)



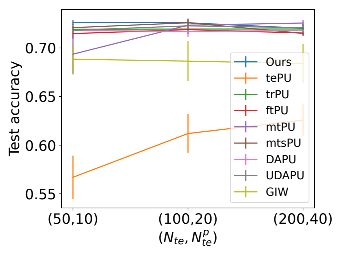
(d) FMNIST(S)



(e) CIFAR10(IO)



(f) CIFAR10(S)



(g) DIABETES

Figure 4: The average test accuracies with the standard errors of the proposed method when changing the number of PU data in the test distribution.

the learning rate to 10^{-4} and 10^{-3} for classifier learning and importance weight estimation, respectively. The maximum number of epochs was 200. All methods were implemented using Pytorch (Paszke et al., 2017) and all experiments were conducted on a Linux server with an Intel Xeon CPU and A100 GPU.

E ADDITIONAL EXPERIMENTAL RESULTS

E.1 Results with Different Numbers of PU Data in the Test Distribution

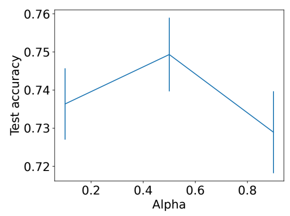
Figure 4 shows the average test accuracies with the standard errors when changing the numbers of PU data in the test distribution. As the number of PU data in the test distribution increased, the performance of the proposed method improved. The proposed method tended to outperform the others across different numbers of PU data in the test distribution.

E.2 Dependency of Relative Parameter α

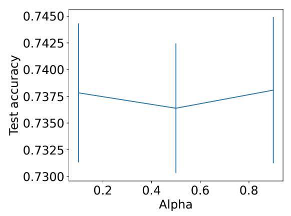
Figure 5 shows the average test accuracies with the standard errors when changing relative parameter α in importance weight estimation in Eq. (14). Although the best values are different across datasets, the performance differences between different values of α were not significant. This result suggests that the proposed method was relatively robust against the value of α.

E.3 Class-prior Estimation Results with PU Data

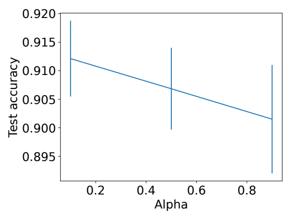
The proposed method assumes the availability of class-prior information $\pi_{\rm tr}$ and $\pi_{\rm te}$. When they are unavailable, we must estimate them from PU data. Thus, we investigated the estimation performance for both $\pi_{\rm tr}$ and $\pi_{\rm te}$. In this experiment, we used a kernel embedding-based class-prior estimation method (Ramaswamy et al., 2016) with FMNIST. In addition, to estimate $\pi_{\rm te}$, we used both training and validation PU data in the test distribution to improve the performance. Note that the number of validation PU data was also small. In this dataset, the true class-prior of $\pi_{\rm tr}$ and $\pi_{\rm te}$ was 0.5. Tables 3 and 4 show the results. Since there are many PU data in the training distribution, the estimated $\pi_{\rm tr}$ was accurate. As for $\pi_{\rm te}$, when the number of PU data in the test distribution is very small, the estimated $\pi_{\rm te}$ was a bit inaccurate compared to the case of $\pi_{\rm tr}$. However, the estimation performance improved as the number of PU data increased, even with a small amount of PU data. Since we used one class-prior estimation method in this experiment, using other methods may lead to more



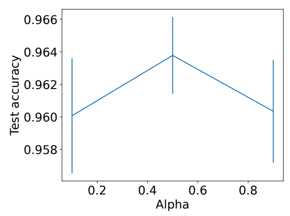
(a) MNIST(IO)



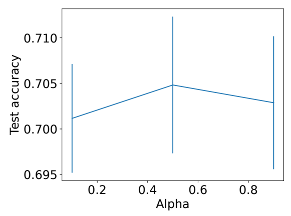
(b) MNIST(S)



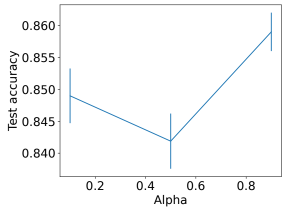
(c) FMNIST(IO)



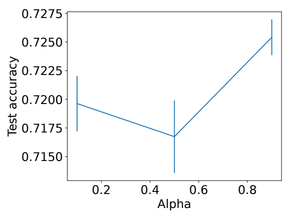
(d) FMNIST(S)



(e) CIFAR10(IO)



(f) CIFAR10(S)



(g) DIABETES

Figure 5: The average test accuracies with the standard errors of the proposed method when changing relative parameter α.

Table 3: Class-prior estimation performance on the training distribution with FMNIST.

Shift	IO	S
Estimated π_{tr}	0.524	0.498

Table 4: Class-prior estimation performance on the test distribution with FMNIST.

Shift	IO	IO	IO	S	S	S
$(N_{\mathrm{te}}^{\mathrm{p}}, N_{\mathrm{te}})$	(10, 50)	(20, 100)	(40, 200)	(10, 50)	(20, 100)	(40, 200)
Estimated π_{te}	0.360	0.414	0.431	0.390	0.419	0.449

accurate estimation performance. Developing an accurate class-prior estimation method from a few data is one of the important research directions.

E.4 Results with Standard Deviations

Table 5 shows the average test accuracies with the standard deviations over different numbers of PU data in the test distribution.

E.5 Computation Cost

Table 6 shows the training time of the distribution shift adaptation methods including the proposed method with MNIST(IO) when $N_{\mathrm{te}}^{\mathrm{p}} = 40$ and $N_{\mathrm{te}} = 200$. We used a Linux server with 2.20Hz CPU. Our method's training time was about the same or slightly longer than the other methods. This result indicates that the proposed method is practical in terms of computation costs.

E.6 Results with the FoodStamp dataset

We additionally evaluated the proposed method with the FoodStamp dataset, which is a real-world tabular dataset used for recent distribution shift adaptation studies (Gardner et al., 2023). The task of this dataset is to predict whether an individual is receiving food stamps. The feature dimension is 239. The distribution shift occurred due to the different geographic regions in which individuals live. In this experiment, weighting parameter β was selected from $\{0.0, 0.1, 0.3, 0.5, 0.7, 0.9\}$ with validation data. Table 7 shows the average test accuracies over different numbers of PU data in the test distribution. The proposed method performed well.

Table 5: Average test accuracies and their standard deviations over different numbers of PU data in the test distribution: $(N_{\mathrm{te}}^{\mathrm{p}}, N_{\mathrm{te}}) = (10, 50), (20, 100)$, and $(40, 200)$. In the Shift column, 'IO' and 'S' represent the input-output relation shift and support shift, respectively.

Data	Shift	Ours	tePU	trPU	ftPU	mtPU
MNIST	IO	0.7493(0.053)	0.7371(0.043)	0.5628(0.009)	0.6807(0.052)	0.7115(0.080)
	S	0.7339(0.038)	0.7295(0.046)	0.5760(0.021)	0.6808(0.048)	0.7482(0.046)
FMNIST	IO	0.9153(0.034)	0.8876(0.037)	0.5851(0.018)	0.7348(0.066)	0.8981(0.038)
	S	0.9656(0.013)	0.9609(0.023)	0.8841(0.023)	0.9175(0.045)	0.9547(0.020)
CIFAR10	IO	0.7081(0.038)	0.6557(0.069)	0.6603(0.032)	0.6699(0.033)	0.6718(0.072)
	S	0.8572(0.017)	0.6633(0.066)	0.8434(0.017)	0.8316(0.022)	0.8092(0.079)
DIABETES		0.7242(0.011)	0.6016(0.067)	0.7194(0.010)	0.7163(0.013)	0.7142(0.041)

Data	Shift	mtsPU	DAPU	GIW	UDAPU
MNIST	IO	0.7152(0.057)	0.7223(0.058)	0.7369(0.055)	0.5499(0.030)
	S	0.7383(0.025)	0.7536(0.034)	0.7459(0.040)	0.6782(0.040)
FMNIST	IO	0.8972(0.053)	0.8937(0.062)	0.8728(0.044)	0.7106(0.079)
	S	0.9489(0.019)	0.9480(0.023)	0.9622(0.019)	0.9398(0.020)
CIFAR10	IO	0.7033(0.042)	0.6967(0.039)	0.6970(0.035)	0.6747(0.037)
	S	0.8388(0.019)	0.8452(9.017)	0.8331(0.023)	0.8477(0.018)
DIABETES		0.7206(0.013)	0.7176(0.013)	0.6863(0.059)	0.7206(0.012)

Table 6: Training time [s] of the proposed method with MNIST(IO) when $N_{\mathrm{te}}^{\mathrm{p}} = 40$ and $N_{\mathrm{te}} = 200$.

Ours	mtPU	mtsPU	DAPU	UDAPU	GIW
244.57	218.36	214.72	232.67	230.90	242.34

Table 7: The average test accuracies over different numbers of PU data in the test distribution on FoodStamp.

Ours	tePU	trPU	ftPU	mtPU	mtsPU	DAPU	UDAPU	GIW
0.7092	0.5757	0.7045	0.6987	0.6370	0.7075	0.7066	0.6930	0.6948