Learning from Positive and Unlabeled Data with Arbitrary Positive Shift

Zayd Hammoudeh [1] **Daniel Lowd** [1]

Abstract

Positive-unlabeled (PU) *learning* trains a binary classifier using only positive and unlabeled data. A common simplifying assumption is that the positive data is representative of the target positive class. This assumption is often violated in practice due to time variation, domain shift, or adversarial concept drift. This paper shows that PU learning is possible even with arbitrarily non-representative positive data when provided unlabeled datasets from the source and target distributions. Our key insight is that only the negative class's distribution need be fixed. We propose two methods to learn under such arbitrary positive bias. The first couples *negative-unlabeled* (NU) *learning* with *unlabeled-unlabeled* (UU) *learning* while the other uses a novel recursive risk estimator robust to positive shift. Experimental results demonstrate our methods' effectiveness across numerous real-world datasets and forms of positive data bias, including disjoint positive class-conditional supports.

1. Introduction

Consider binary classification which learns a function that labels each example as either positive or negative. *Positive-negative* (PN) *learning* (i.e., ordinary supervised classification) trains the classifier using positive and negative labeled sets. In practice, good labeled data is often unavailable for one class. For instance, negative class diversity may make construction of a representative, labeled set prohibitively difficult (du Plessis et al., 2015). In other cases, negative data may not be (systematically) recorded like in the medical domain where patient records include diagnosed maladies but may fail to note diseases a person is proven not to have.

Positive-unlabeled (PU) *learning* addresses this problem by constructing classifiers using only labeled-positive and unlabeled data. PU learning has been applied to numerous real-world domains including: disease-gene identification (Yang et al., 2012), land-cover classification (Li et al., 2011), protein similarity prediction (Elkan & Noto, 2008), and outlier detection (Hido et al., 2008; Scott & Blanchard, 2009). The related task of *negative-unlabeled* (NU) learning is functionally identical, but with labeled data from the negative class instead of positive.

Previous PU learning methods generally assume the labeled set is *selected completely at random* (SCAR) from the target distribution (Elkan & Noto, 2008). External factors such as time variation, domain shift, and adversarial concept drift may cause a divergence between the labeled-positive and target distributions. These variations may be unknown or not fully understood, restricting data reuse. For example, looking at the medical domain again, virus strains and their bodily effects often vary temporally (through mutation) and geographically (Neves et al., 2017; Xiao et al., 2017).

This paper proposes *arbitrary-positive, unlabeled* (aPU) classification using only positive and unlabeled data, where the labeled (positive) set may be arbitrarily different from the target distribution's positive class. Solving this problem has obvious benefits such as enabling reuse of existing labeled data, which reduces cost and time-to-market.

aPU learning is distinct from *biased-positive, unlabeled* (bPU) learning where specific positive shift types are assumed. Existing bPU approaches fall into two subcategories. First, *selection bias* methods replace SCAR with new, less-restrictive assumptions. For example, Bekker et al. (2019) posit that the labeling probability is a function of a subset of an example's attributes while Kato et al. (2019) assume examples' labeling probabilities are ordered based on their positive posteriors.

The other subcategory frames bPU learning as a *covariate shift* problem by considering the difference between the labeled and target distributions. Quionero-Candela et al. (2009) integrate the importance function (i.e., ratio of the source and target marginal densities) into a risk minimization framework to improve classifier performance under distributional shifts. Sakai & Shimizu (2019) apply this idea to PU learning by assuming availability of a labeled (positive) set and two unlabeled sets – one whose positive examples are sampled like the labeled set and the other drawn from the target distribution.

[1] Department of Computer & Information Science, University of Oregon, Eugene, OR, USA. Correspondence to: Zayd Hammoudeh <zayd@cs.uoregon.edu>.

bPU methods restrict divergence between the labeled and target distributions' *support*, i.e., the set of all possible examples with positive probability. Selection bias approaches assume that the labeling distribution's support is a subset of the target distribution's support. Covariate shift via importance weighting loses its statistical guarantees whenever the target support is not a subset of the labeled support.

Elkan & Noto (2008) state that devoid of any assumptions aPU learning is impossible. Our key insight is that given a labeled-positive set and two unlabeled sets as proposed by Sakai & Shimizu (2019), aPU learning is possible if *all negative examples are generated from a single distribution*. The labeled and target-positive distributions' supports can even be disjoint. Many real-world PU learning scenarios feature a shifting positive class but fixed negative class including:

1. **Epidemiological Analysis**: PU learning's applicability to disease-related domains and the potential for bias are discussed above. The characteristics of the negative class (healthy population) can be temporally and/or geographically consistent.
2. **Land-Cover Classification**: Adjacent countries' geographic terrain may follow a similar distribution, but man-made object classes (e.g., roads) differ due to local construction methods, materials, and regulations.
3. **Adversarial Image Manipulation**: Fake news spreads manipulated images with new manipulation types evolving adversarially. The unmanipulated image class is nearly static, but creating a representative negative set is both challenging given image diversity and noisy since users manipulate their own images (e.g., via filters).

Our paper's contributions are three-fold:

1. We address our aPU learning task via a two-step formulation; the first step applies standard PU learning and the second uses unlabeled-unlabeled (UU) learning.
2. We separately propose PURR — a novel, recursive aPU risk estimator that works within an empirical risk minimization (ERM) framework.
3. We evaluate our methods on a wide range of benchmark datasets, demonstrating our algorithms' effectiveness over the state-of-the-art in PU and bPU learning.

Additional experiments and all proofs are in the supplemental materials.

2. Standard Positive-Unlabeled Learning

Before exploring aPU learning, it is helpful to be familiar with PU learning without distributional shifts including standard definitions, formulations, and notation.

Define two random variables, covariate $X \in \mathbb{R}^d$ ($d \in \mathbb{N}$) and dependent label $Y \in \{\pm 1\}$. Let $p(x, y)$ be their (unknown) joint distribution. Marginal distribution $p_{\text{u}}(x)$ is composed from positive prior $\pi \coloneqq p(Y = +1)$, positive class-conditional $p_{\text{p}}(x) \coloneqq p(x|Y = +1)$, and negative class-conditional $p_{\text{n}}(x) \coloneqq p(x|Y = -1)$.

Risk Estimation Let $g : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary *decision function* parameterized by θ, and let $\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be the *loss function*. Risk $R(g) \coloneqq \mathbb{E}_{(X,Y)\sim p(x,y)}[\ell(Yg(X))]$ quantifies g's expected loss over $p(x, y)$. It decomposes via the product rule to $R(g) = \pi R_{\text{p}}^{+}(g) + (1 - \pi) R_{\text{n}}^{-}(g)$, where the *labeled risk* is

$$R_{\mathcal{D}}^{\hat{y}}(g) \coloneqq \mathbb{E}_{X\sim p_{\mathcal{D}}(x)}[\ell(\hat{y}g(X))] \tag{1}$$

for predicted label, $\hat{y} \in \{\pm 1\}$ and $\mathcal{D} \in \{\text{p}, \text{n}, \text{u}\}$ denoting the distribution. Since $p(x, y)$ is unknown, *empirical risk* is used in practice; we denote the positive-negative risk

$$\widehat{R}_{\text{PN}}(g) \coloneqq \pi \widehat{R}_{\text{p}}^{+}(g) + (1 - \pi)\widehat{R}_{\text{n}}^{-}(g). \tag{2}$$

We consider the (slightly) more general *case-control scenario* (Niu et al., 2016) where each dataset is i.i.d. sampled from its associated distribution. PN learning has two labeled datasets, positive set $\mathcal{X}_{\text{p}} \coloneqq \{x_i^{\text{p}}\}_{i=1}^{n_{\text{p}}} \overset{\text{i.i.d.}}{\sim} p_{\text{p}}(x)$ and negative set $\mathcal{X}_{\text{n}} \coloneqq \{x_i^{\text{n}}\}_{i=1}^{n_{\text{n}}} \overset{\text{i.i.d.}}{\sim} p_{\text{n}}(x)$. These are used to calculate empirical labeled risks $\widehat{R}_{\text{p}}^{+}(g) = \frac{1}{n_{\text{p}}}\sum_{i=1}^{n_{\text{p}}} \ell\big(g(x_i^{\text{p}})\big)$ and $\widehat{R}_{\text{n}}^{-}(g) = \frac{1}{n_{\text{n}}}\sum_{i=1}^{n_{\text{n}}} \ell(-g(x_i^{\text{n}}))$.

PU learning cannot directly estimate $\widehat{R}_{\text{n}}^{\hat{y}}(g)$ since there is no negative (labeled) data. du Plessis et al. (2014) make the foundational observation that,

$$(1 - \pi)R_{\text{n}}^{\hat{y}}(g) = R_{\text{u}}^{\hat{y}}(g) - \pi R_{\text{p}}^{\hat{y}}(g). \tag{3}$$

Using this insight and provided an unlabeled set $\mathcal{X}_{\text{u}} \coloneqq \{x_i^{\text{u}}\}_{i=1}^{n_{\text{u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{u}}(x)$, their unbiased PU (uPU) estimator is

$$\widehat{R}_{\text{uPU}}(g) \coloneqq \pi \widehat{R}_{\text{p}}^{+}(g) + \widehat{R}_{\text{u}}^{-}(g) - \pi \widehat{R}_{\text{p}}^{-}(g),$$

where $\widehat{R}_{\text{u}}^{\hat{y}}(g) = \frac{1}{n_{\text{u}}}\sum_{i=1}^{n_{\text{u}}} \ell(\hat{y}g(x_i^{\text{u}}))$.

Kiryo et al. (2017) make the important observation that highly expressive models (e.g., neural networks) can overfit $\mathcal{X}_{\text{p}}$ causing uPU to estimate that $\widehat{R}_{\text{u}}^{-}(g) - \pi\widehat{R}_{\text{p}}^{-}(g) < 0$. Since $\forall_t \ell(t) \geq 0$, negative risk is impossible. Kiryo et al.'s non-negative PU (nnPU) risk estimator,

$$\widehat{R}_{\text{nnPU}}(g) \coloneqq \pi \widehat{R}_{\text{p}}^{+}(g) + \max\{0, \widehat{R}_{\text{u}}^{-}(g) - \pi \widehat{R}_{\text{p}}^{-}(g)\}, \tag{4}$$

uses its max term to ignore negative risk estimates. Whenever their training algorithm detects overfitting (i.e., $\widehat{R}_{\text{n}}^{-}(g) < 0$), it "defits" g by using negated gradient $-\gamma \nabla_\theta \left(\widehat{R}_{\text{u}}^{-}(g) - \pi \widehat{R}_{\text{p}}^{-}(g)\right)$, where hyperparameter $\gamma \in (0, 1]$ attenuates the learning rate to throttle "defitting."

3. Arbitrary-Positive Unlabeled Learning

Arbitrary-positive unlabeled (aPU) learning — the focus of this work — is one of three problem settings proposed by Sakai & Shimizu (2019). We generalize their original definition below.

Consider two joint distributions: train $p_{\text{tr}}(x, y)$ and test $p_{\text{te}}(x, y)$. Notation $p_{\text{tr-}\mathcal{D}}(x)$ where $\mathcal{D} \in \{\text{p}, \text{n}, \text{u}\}$ refers to the training positive class-conditional, negative class-conditional, and marginal distributions respectively. $p_{\text{te-}\mathcal{D}}(x)$ denotes the corresponding test distributions.

We make no assumption about the conditional probability of the label, $p_{\text{tr}}(y|x)$ and $p_{\text{te}}(y|x)$, nor the positive class-conditional distributions, $p_{\text{tr-p}}(x)$ and $p_{\text{te-p}}(x)$, only that the negative class-conditional distribution is fixed, i.e.,

$$p_{\text{n}}(x) = p_{\text{tr-n}}(x) = p_{\text{te-n}}(x). \tag{5}$$

As discussed in Section 1, this assumption fits domains like epidemiology, land-cover classification, and media forensics, where it may be difficult to obtain a representative negative sample and where the positive class changes quickly.

Both the train and test positive-class priors, π_{tr} and π_{te} respectively, are assumed known throughout this paper. In practice, they can be estimated from data (Ramaswamy et al., 2016; du Plessis et al., 2017). We propose a technique to estimate π_{te} in the supplemental materials.

The available data is labeled (positive) set $\mathcal{X}_{\text{p}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-p}}(x)$ as before and unlabeled sets $\mathcal{X}_{\text{tr-u}} := \{x_i\}_{i=1}^{n_{\text{tr-u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)$ and $\mathcal{X}_{\text{te-u}} := \{x_i\}_{i=1}^{n_{\text{te-u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{te-u}}(x)$. Their labeled empirical risks are defined like their non-shifted equivalents.

The goal is to select the decision function, g, minimizing the expected *test* loss,

$$R_{\text{te}}(g) := \mathbb{E}_{(X,Y)\sim p_{\text{te}}(x,y)}[\ell(Yg(X))].$$

For brevity, we refer to the positive and negative class-conditional distributions as the *positive distribution* and *negative distribution*, respectively. As needed, we additionally specify whether they are conditionals from the training or test distribution. By the assumption of Eq. (5), the negative train and test distributions are the same so we refer to them jointly as the negative distribution.

Relating aPU and Covariate Shift Adaptation Methods

Covariate shift (Shimodaira, 2000) is a common technique to address differences between the source (train) distribution, $p_{\text{tr}}(x, y)$, and the target (test) distribution of interest, $p_{\text{te}}(x, y)$. Unlike aPU learning, covariate shift restrictively assumes a consistent input-output relation; stated formally, $p_{\text{tr}}(y|x) = p_{\text{te}}(y|x)$. The *importance function* is defined as $w(x) := \frac{p_{\text{te-u}}(x)}{p_{\text{tr-u}}(x)}$. When the conditional distribution $p(y|x)$ is fixed, it is easy to show that $w(x)p_{\text{tr}}(x, y) = p_{\text{te}}(x, y)$.

Algorithm 1 Two-step unlabeled-unlabeled learning

```
Input: Data $(\mathcal{X}_{\text{p}}, \mathcal{X}_{\text{tr-u}}, \mathcal{X}_{\text{te-u}})$
Output: Model parameters $\theta$
1: Train probabilistic classifier $\hat{\sigma}$ using $\mathcal{X}_{\text{p}}$ and $\mathcal{X}_{\text{tr-u}}$
2: Use $\hat{\sigma}$ to transform $\mathcal{X}_{\text{tr-u}}$ into surrogate negative set $\widetilde{\mathcal{X}}_{\text{n}}$
3: Train $g(x)$ using ERM with $\widehat{R}_{\text{wUU}}(g)$ or $\widehat{R}_{\text{aPNU}}(g)$
```

Sakai & Shimizu (2019) exploit this relationship to build their PUc risk estimator. They estimate $w(x)$ using direct density-ratio estimation (Sugiyama et al., 2012), specifically applying the RuLSIF algorithm (Yamada et al., 2013) to $\mathcal{X}_{\text{tr-u}}$ and $\mathcal{X}_{\text{te-u}}$. Their PUc risk is an importance-weighted version of uPU, with each expectation estimated empirically from $\mathcal{X}_{\text{p}}$ or $\mathcal{X}_{\text{tr-u}}$. Sakai & Shimizu's formulation specifies linear-in-parameter models to enforce convexity. For improved tractability, they use a simplified version of du Plessis et al. (2015)'s surrogate squared loss for loss ℓ.

Among bPU methods, PUc's assumptions about the underlying positive bias mechanism are least restrictive. This makes it most adaptable to arbitrary shifts, which is why we selected it as Section 6's primary experimental baseline.

4. aPU Learning via UU Learning

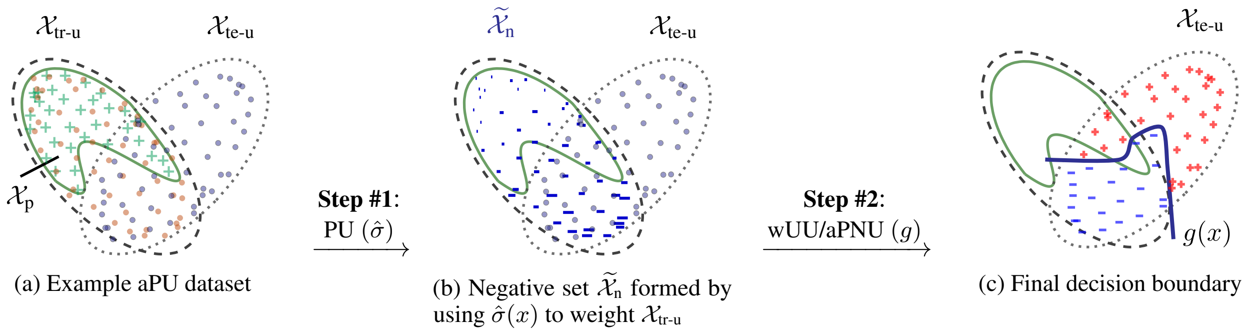
To build an intuition for solving the aPU learning problem, consider the ideal case where a perfect classifier correctly labels $\mathcal{X}_{\text{tr-u}}$. Let $\mathcal{X}_{\text{tr-n}}$ be $\mathcal{X}_{\text{tr-u}}$'s negative examples. $\mathcal{X}_{\text{tr-n}}$ is SCAR w.r.t. $p_{\text{tr-n}}(x)$ and by Eq. (5)'s assumption also $p_{\text{te-n}}(x)$. Multiple options exist to then train the second classifier, g, e.g., NU learning with $\mathcal{X}_{\text{tr-n}}$ and $\mathcal{X}_{\text{te-u}}$.

A perfect classifier is unrealistic. Is there an alternative? Our key insight is that by weighting $\mathcal{X}_{\text{tr-u}}$ (similar to covariate shift's importance function) it can be transformed into a representative negative set. From there, we consider two methods to fit the second classifier: a variant of NU learning we call weighted-unlabeled, unlabeled (wUU) learning and a semi-supervised method we call arbitrary-positive, negative, unlabeled (aPNU) learning. We refer to the complete algorithms as PU2wUU and PU2aPNU, respectively.

Figure 1 provides a visual representation of our two-step method, with a formal presentation in Algorithm 1. Below is a detailed description and theoretical analysis of our method.

Step #1: Create a Representative Negative Set

This step's goal is to learn the training distribution's negative class-posterior, $p_{\text{tr}}(Y = -1|x)$. We achieve this by training PU probabilistic classifier $\hat{\sigma} : \mathbb{R}^d \to [0, 1]$ using $\mathcal{X}_{\text{p}}$ and $\mathcal{X}_{\text{tr-u}}$. In principle, any probabilistic PU method can be


$\mathcal{X}_{\text{tr-u}}$
$\mathcal{X}_{\text{te-u}}$
$\mathcal{X}_{\text{p}}$
Step #1:
PU ($\hat{\sigma}$)
$\widetilde{\mathcal{X}}_{\text{n}}$
$\mathcal{X}_{\text{te-u}}$
Step #2:
wUU/aPNU (g)
$\mathcal{X}_{\text{te-u}}$
$g(x)$

(a) Example aPU dataset

(b) Negative set $\widetilde{\mathcal{X}}_{\text{n}}$ formed by using $\hat{\sigma}(x)$ to weight $\mathcal{X}_{\text{tr-u}}$

(c) Final decision boundary

Figure 1. Two-step aPU learning. Fig. 1a shows a toy aPU dataset with (+) representing a labeled positive example, (●) an unlabeled train sample, and (●) an unlabeled test sample. Borders surround each set for clarity. After learning probabilistic classifier $\hat{\sigma}$ in Step #1, Fig. 1b visualizes $\hat{\sigma}$'s predicted negative-posterior probability using marker (−) size. Fig. 1c shows the final decision boundary with (-) and (+) representing $\mathcal{X}_{\text{te-u}}$ examples classified negative and positive respectively.

used; we focused on ERM-based PU methods so the logistic loss served as surrogate, ℓ. Sigmoid activation is applied to the model's output to bound its range to $(0, 1)$.

Theorem 1. *Let $g : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary decision function and $\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be a loss function bounded w.r.t. g[1]. Let $\hat{y} \in \{\pm 1\}$ be a predicted label. Define $\mathcal{X}_{\text{tr-u}} := \{x_i\}_{i=1}^{n_{\text{tr-u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)$, and restrict $\pi_{\text{tr}} \in [0, 1)$. Define*

$$\widetilde{R}^{\hat{y}}_{\text{n-u}}(g) := \frac{1}{n_{\text{tr-u}}} \sum_{x_i \in \mathcal{X}_{\text{tr-u}}} \frac{\hat{\sigma}(x_i)\ell(\hat{y}g(x_i))}{1 - \pi_{\text{tr}}}.$$

Let $\hat{\sigma} : \mathbb{R}^d \to [0, 1]$ in hypothesis set $\widehat{\Sigma}$. When $\hat{\sigma}(x) = p_{\text{tr}}(Y = -1|x)$, $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ is an unbiased estimator of $R^{\hat{y}}_{\text{n}}(g)$. When the concept class of functions that defines $p_{\text{tr}}(Y = -1|x)$ is probably approximately correct (PAC) learnable by some PAC-learning algorithm $\mathcal{A}$ that selects $\hat{\sigma} \in \widehat{\Sigma}$, then $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ is a consistent estimator of $R^{\hat{y}}_{\text{n}}(g)$.

From Theorem 1, we see that weighting each unlabeled training instance in $\mathcal{X}_{\text{tr-u}}$ by $\hat{\sigma}$ yields a *surrogate negative set* $\widetilde{\mathcal{X}}_{\text{n}}$ that can be used to estimate the train and test negative expected risk.

Implementation $\widetilde{\mathcal{X}}_{\text{n}}$ is formed by transductively applying negative-posterior estimate $\hat{\sigma}(x)$. With large enough $\mathcal{X}_{\text{tr-u}}$, inductive learning is also possible, i.e., split $\mathcal{X}_{\text{tr-u}}$ with part used to train $\hat{\sigma}$ and the rest used to form $\widetilde{\mathcal{X}}_{\text{n}}$.

Since $\mathcal{X}_{\text{tr-u}}$ contains positive examples, it is possible for $\hat{\sigma}(x)$ to overfit and memorize random positive example variation. Tuning model capacity (e.g., hidden layer count) and regularization strength mitigates this phenomenon.

Step #2: Classify $\mathcal{X}_{\text{te-u}}$

As mentioned previously, negative-unlabeled (NU) learning is essentially the same as PU learning. Sakai et al. (2017) formalize the unbiased NU risk estimator; our modified definition below includes a non-negativity (nn) correction:

$$\widehat{R}_{\text{nnNU}}(g) := \max\{0, \widehat{R}^{+}_{\text{u}}(g) - (1-\pi)\widehat{R}^{+}_{\text{n}}(g)\} + (1-\pi)\widehat{R}^{-}_{\text{n}}(g).$$

Our *weighted-unlabeled, unlabeled* (wUU) estimator,

$$\begin{aligned}\widehat{R}_{\text{wUU}}(g) := &\max\{0, \widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g)\} \\ &+ (1-\pi_{\text{te}})\widetilde{R}^{-}_{\text{n-u}}(g),\end{aligned} \tag{6}$$

modifies nnNU to use $\widetilde{\mathcal{X}}_{\text{n}}$. Notice this classifier is trained using only data that was originally unlabeled. When $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ is consistent, wUU is also consistent just like nnPU/nnNU.

Risk Estimation with Positive Data Reuse wUU disregards $\mathcal{X}_{\text{p}}$ due to its arbitrary shift. When $p_{\text{tr-p}}(x)$'s and $p_{\text{te-p}}(x)$'s supports intersect, $\mathcal{X}_{\text{p}}$ may contain useful information about the target distribution, in particular when data is limited. As such, a semi-supervised approach leveraging $\mathcal{X}_{\text{p}}$, surrogate $\widetilde{\mathcal{X}}_{\text{n}}$, and $\mathcal{X}_{\text{te-u}}$ may perform better in practice despite lacking statistical guarantees.

Sakai et al. (2017) propose the PNU risk estimator,

$$\widehat{R}_{\text{PNU}}(g) := (1-\rho)\widehat{R}_{\text{PN}}(g) + \rho\widehat{R}_{\text{NU}}(g),$$

where hyperparameter $\rho \in [0, 1]$ weights the PN and NU estimators. Our arbitrary-positive, negative, unlabeled (aPNU) risk estimator in Eq. (7) modifies PNU to use $\widetilde{\mathcal{X}}_{\text{n}}$ and non-negativity correction. When $\rho = 0$, aPNU ignores the test distribution entirely. When $\rho = 1$, aPNU is simply wUU.

$$\begin{aligned}\widehat{R}_{\text{aPNU}}(g) = &(1-\rho)\pi_{\text{te}}R^{+}_{\text{p}}(g) + (1-\pi_{\text{te}})\widetilde{R}^{-}_{\text{n-u}}(g) \\ &+ \rho\max\{0, R^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g)\}\end{aligned} \tag{7}$$

ERM Framework Both $\widehat{R}_{\text{wUU}}(g)$ and $\widehat{R}_{\text{aPNU}}(g)$ integrate into the ERM framework proposed by Kiryo et al. (2017), which defits the learner whenever $\widehat{R}^{+}_{\text{u}}(g) - \widetilde{R}^{+}_{\text{n-u}}(g)$ is negative. For reference, we include Kiryo et al.'s ERM algorithm in the supplemental material.

[1] The precise definition of bounding appears in the proof.

5. PU Recursive Risk Estimation

Two-step methods — both ours and PUc — solve a challenging problem by decomposing it into sequential (easier) subproblems. Serial decision making's disadvantage is that earlier errors propagate and can be amplified when subsequent decisions are made on top of the error.

Can our aPU problem setting be learned in a single *joint* method? Sakai & Shimizu leave it as an open question. We show in this section the answer is yes, although the resulting risk estimator is challenging to optimize.

To understand why this is possible, it helps to simplify our perspective of unbiased PU and NU learning. When estimating a labeled risk, $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$ (where $\mathcal{D} \in \{\text{p}, \text{n}\}$), the ideal case is to use SCAR data from class-conditional distribution $p_{\mathcal{D}}(x)$. When such labeled data is unavailable, risk *decomposes* via simple linear transformation,

$$(1 - \alpha)\widehat{R}^{\hat{y}}_{A}(g) = \widehat{R}^{\hat{y}}_{\text{u}}(g) - \alpha\widehat{R}^{\hat{y}}_{B}(g) \tag{8}$$

where $A = \text{n}$ and $B = \text{p}$ for PU learning or vice versa for NU learning. α is the positive (negative) prior for PU (NU) learning.

In standard PU and NU learning, either $\widehat{R}^{\hat{y}}_{A}(g)$ or $\widehat{R}^{\hat{y}}_{B}(g)$ can always be estimated directly from labeled data. If that were not true, can this decomposition be applied recursively (i.e., nested)? The answer is again yes. Below we apply recursive risk decomposition to our aPU learning task.

5.1. Applying Recursive Risk to aPU learning

Our positive-unlabeled recursive risk (PURR) estimator quantifies our aPU setting's empirical risk. PURR's top-level definition matches the goal in Eq. (3), i.e.,

$$\widehat{R}_{\text{PURR}}(g) = \pi_{\text{te}}\widehat{R}^{+}_{\text{te-p}}(g) + (1 - \pi_{\text{te}})\widehat{R}^{-}_{\text{te-n}}(g). \tag{9}$$

Since only unlabeled data is drawn from the test distribution, both terms in Eq. (9) require risk decomposition.

Consider first Eq. (9)'s negative labeled risk, $\widehat{R}^{-}_{\text{te-n}}(g)$. We find its more general form $\widehat{R}^{\hat{y}}_{\text{te-n}}(g)$ below since $\widehat{R}^{+}_{\text{te-n}}(g)$ will be needed as well. Using Eq. (5)'s assumption, $\widehat{R}^{\hat{y}}_{\text{te-n}}(g)$ can be estimated directly from the training distribution. Combining Eq. (3) with non-negativity correction, we see that

$$\widehat{R}^{\hat{y}}_{\text{te-n}}(g) = \widehat{R}^{\hat{y}}_{\text{tr-n}}(g) = \max\left\{0, \frac{\widehat{R}^{\hat{y}}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{\hat{y}}_{\text{tr-p}}(g)}{1 - \pi_{\text{tr}}}\right\}. \tag{10}$$

Next consider $\widehat{R}^{+}_{\text{te-p}}(g)$. Since it concerns the positive class, NU decomposition (with non-negativity) is applied, so

$$\pi_{\text{te}}\widehat{R}^{+}_{\text{te-p}}(g) = \max\left\{0, \widehat{R}^{+}_{\text{te-u}}(g) - (1 - \pi_{\text{te}})\widehat{R}^{+}_{\text{te-n}}(g)\right\}. \tag{11}$$

Eq. (10) with $\hat{y} = +1$ substitutes for $\widehat{R}^{+}_{\text{te-n}}(g)$ completing $\widehat{R}_{\text{PURR}}(g)$'s recursive definition.

Algorithm 2 PURR ERM procedure

Input: Data $(\mathcal{X}_{\text{p}}, \mathcal{X}_{\text{tr-u}}, \mathcal{X}_{\text{te-u}})$ & hyperparameters (γ, η)
Output: Model parameters θ

```
 1: Select SGD-like optimization algorithm $\mathcal{A}$
 2: while Stopping criteria not met do
 3:   Shuffle $(\mathcal{X}_{\text{p}}, \mathcal{X}_{\text{tr-u}}, \mathcal{X}_{\text{te-u}})$ into $N$ batches
 4:   for each minibatch $(\mathcal{X}^{(i)}_{\text{p}}, \mathcal{X}^{(i)}_{\text{tr-u}}, \mathcal{X}^{(i)}_{\text{te-u}})$ do
 5:     if $\widehat{R}^{-}_{\text{te-n}}(g) < 0$ then
 6:       Use $\mathcal{A}$ to update $\theta$ with $-\gamma\eta\nabla_{\theta}\widehat{R}^{-}_{\text{te-n}}(g)$
 7:     else if $\widehat{R}^{+}_{\text{te-n}}(g) < 0$ then
 8:       Use $\mathcal{A}$ to update $\theta$ with $-\gamma\eta\nabla_{\theta}\widehat{R}^{+}_{\text{te-n}}(g)$
 9:     else if $\widehat{R}^{+}_{\text{te-p}}(g) < 0$ then
10:       Use $\mathcal{A}$ to update $\theta$ with $-\gamma\eta\nabla_{\theta}\widehat{R}^{+}_{\text{te-p}}(g)$
11:     else
12:       Use $\mathcal{A}$ to update $\theta$ with $\eta\nabla_{\theta}\widehat{R}_{\text{PURR}}(g)$
13: return $\theta$ minimizing validation loss
```

Theorem 2. *Fix decision function $g \in \mathcal{G}$. If ℓ is bounded over $g(x)$'s image*[2] *and $\widehat{R}^{\hat{y}}_{\text{te-n}}(g), \widehat{R}^{+}_{\text{te-p}}(g) > 0$ for $\hat{y} \in \{\pm 1\}$, then $\widehat{R}_{\text{PURR}}(g)$ is a consistent estimator. $\widehat{R}_{\text{PURR}}(g)$ is a biased estimator unless for all $\mathcal{X}_{\text{tr-u}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)$, $\mathcal{X}_{\text{te-u}} \overset{\text{i.i.d.}}{\sim} p_{\text{te-u}}(x)$, and $\mathcal{X}_{\text{p}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-p}}(x)$ it holds that $\Pr\Big[\widehat{R}^{\hat{y}}_{\text{tr-u}}(g) - (1 - \pi_{\text{te}})\widehat{R}^{\hat{y}}_{\text{tr-p}}(g) < 0\Big] = 0$ and $\Pr\Big[\widehat{R}^{+}_{\text{te-u}}(g) - (1 - \pi_{\text{te}})\widehat{R}^{+}_{\text{te-n}}(g) < 0\Big] = 0.$*

5.2. Optimizing PURR

PURR is an attractive solution since it unifies the disparate training sets into a single objective function. However, each risk decomposition needs its own non-negativity correction so in total, PURR has three $\max$ terms — one of which is nested inside another $\max$. Each $\max$ can overfit and need to be "defit." Their competing interactions can make PURR more challenging to optimize than our two-step methods.

Algorithm 2 details PURR's ERM procedure with learning rate η. Each non-negativity correction is individually checked with the ordering critical. First, overfitting is most likely with labeled (positive) data. When that occurs, $\widehat{R}^{-}_{\text{tr-p}}(g)$ increases significantly making $\widehat{R}^{-}_{\text{te-n}}(g)$ most likely to be negative so it is checked first (line 5). Nested term $\widehat{R}^{+}_{\text{te-n}}(g)$ receives second highest priority since whenever its value is invalid, any term depending on it, e.g., $\widehat{R}^{+}_{\text{te-p}}(g)$, is meaningless. By elimination, $\widehat{R}^{+}_{\text{te-p}}(g)$ has lowest priority.

Algorithm 2 applies non-negativity correction by negating

[2] The precise definition of bounding appears in the proof.

risk $\widehat{R}_{A}^{\hat{y}}(g)$'s gradient. This addresses overfitting by "defitting" g. A large negative gradient can push g into a poor parameter space so hyperparameter $\gamma \in (0,1]$ limits the amount of correction by attenuating gradient magnitude.

6. Experimental Results

This section empirically demonstrates the effectiveness of our algorithms – PURR, PU2wUU, and PU2aPNU – using synthetic and real-world datasets.[3]

We focused on training neural networks using stochastic optimization — specifically the AdamW optimizer (Loshchilov & Hutter, 2017) with AMSGrad (Reddi et al., 2018). Probabilistic classifier, $\hat{\sigma}$, was trained using nnPU and logistic loss as ℓ. All other learners used the sigmoid loss. Like previous work, performance is evaluated using inductive misclassification rate which empirically estimates an unseen example's expected zero-one loss.

Baselines In each experimental trial, all algorithms saw identical random dataset samplings. The primary baseline is Sakai & Shimizu (2019)'s PUc method with a linear model in Section 6.1 and a linear-in-parameter model with Gaussian kernel basis in the remaining sections.

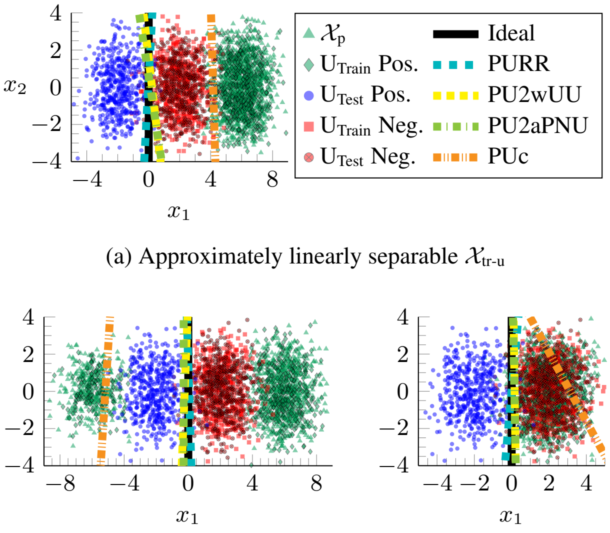
We also report results for PN learning (trained on labeled $\mathcal{X}_{\text{te-u}}$ using Eq. (1)) as well as PU learning via nnPU, which represent the performance ceiling and floor respectively. No single unlabeled set configuration consistently yielded the best nnPU results so we separately trained nnPU using $\mathcal{X}_{\text{te-u}}$ and with combined $\mathcal{X}_{\text{tr-u}} \cup \mathcal{X}_{\text{te-u}}$ (using the true, composite prior). To provide the strongest baseline, we always report the best performing nnPU configuration, which we denote nnPU*.

Hyperparameters Hyperparameters and the best epoch were selected using a validation set, one-fifth the training set size; the validation loss was calculated using the learner's associated risk estimator. As detailed in the supplemental materials, the tuned hyperparameters for each learner were learning rate, weight decay, and (if applicable) γ. As specified by its authors, PUc's hyperparameters were tuned via importance-weighted cross validation (Sugiyama et al., 2007).

We empirically observed that aPNU trade-off parameter $\rho = 0.5$ performed well; we only report that setting's results.

6.1. Illustration using Synthetic Data

This section uses synthetic data to highlight scenarios where our algorithms succeed in spite of challenging conditions.



(a) Approximately linearly separable $\mathcal{X}_{\text{tr-u}}$

(b) Non-linearly separable $\mathcal{X}_{\text{tr-u}}$

(c) $p_{\text{tr-(c)-p}}(x) = p_{\text{n}}(x)$

Figure 2. Predicted linear decision boundaries for three synthetic datasets ($n_{\text{p}} = n_{\text{tr-u}} = n_{\text{te-u}} = 1,000$). Our methods are robust to non-linear or non-existent training class boundaries, but PUc fails in all three cases. The ideal boundary is $x_1 = 0$.

For simplicity, $\hat{\sigma}$ and g are linear models optimized by L-BFGS. PUc also trains linear models for this section so our performance advantage is solely algorithmic.

Synthetic data were generated from multivariate Gaussians $\mathcal{N}(\boldsymbol{\mu}, \mathbf{I}_2)$ with different means $\boldsymbol{\mu}$ and identity covariance $\mathbf{I}_2$. In all experiments, the positive-test and negative distributions were

$$p_{\text{te-p}}(x) = \frac{1}{2}\mathcal{N}(\begin{bmatrix}-2 & -1\end{bmatrix}, \mathbf{I}_2) + \frac{1}{2}\mathcal{N}(\begin{bmatrix}-2 & 1\end{bmatrix}, \mathbf{I}_2)$$

$$p_{\text{n}}(x) = \frac{1}{2}\mathcal{N}(\begin{bmatrix}2 & -1\end{bmatrix}, \mathbf{I}_2) + \frac{1}{2}\mathcal{N}(\begin{bmatrix}2 & 1\end{bmatrix}, \mathbf{I}_2).$$

$\pi_{\text{te}} = \pi_{\text{tr}} = 0.5$ makes the ideal *test* decision boundary $x_1 = 0$. Datasets in Figure 2 vary only in the positive-train distribution, denoted $p_{\text{tr-}(\cdot)\text{-p}}(x)$ where "$\cdot$" is subfigure a to c.

Figure 2a's positive-train distribution is

$$p_{\text{tr-(a)-p}}(x) = \frac{1}{2}\mathcal{N}(\begin{bmatrix}6 & -1\end{bmatrix}, \mathbf{I}_2) + \frac{1}{2}\mathcal{N}(\begin{bmatrix}6 & 1\end{bmatrix}, \mathbf{I}_2),$$

making the training distribution's optimal separator linear. PUc performed poorly for two reasons: covariate shift's assumption $p_{\text{tr}}(y|x) = p_{\text{te}}(y|x)$ does not hold, and the positive-train supports are functionally disjoint so importance function $w(x)$ is practically unbounded. Our methods all performed well, even PU2aPNU where inclusion of $\mathcal{X}_{\text{p}}$'s risk had minimal impact since for most good boundaries, $\mathcal{X}_{\text{p}}$'s risk was an inconsequential penalty.

Figure 2b adds to $p_{\text{tr-(a)-p}}(x)$ a third Gaussian where

$$p_{\text{tr-(b)-p}}(x) = \frac{2}{3}p_{\text{tr-(a)-p}}(x) + \frac{1}{3}\mathcal{N}(\begin{bmatrix}-6 & 0\end{bmatrix}, \mathbf{I}_2),$$

[3] Our implementation is available online at: https://github.com/ZaydH/arbitrary_positive_unlabeled

so the training distribution's optimal separator is non-linear. PUc performs poorly for the same reasons described above. The new centroid does not meaningfully affect PURR. The most important takeaway is that linear $\hat{\sigma}$'s inability to partition $\mathcal{X}_{\text{tr-u}}$ has limited impact on PU2wUU and PU2aPNU; $\mathcal{X}_{\text{tr-u}}$'s misclassified examples act as a fixed penalty that only slightly offsets the two-step decision boundaries.

Figure 2c uses the worst-case positive-train distribution, $p_{\text{tr-(c)-p}}(x) = p_{\text{n}}(x)$, making positive (labeled) data statistically identical to the negative distribution. Its training marginal $p_{\text{tr-u}}(x)$ is not separable — linearly or otherwise. Unlike PUc, our methods learned correct boundaries, which shows their robustness.

6.2. Partially and Fully Disjoint Positive Supports

This section's experiments replicate scenarios where entire positive subclasses exist only in the target (test) distribution. Such phenomena are common in adversarial domains where novel attack types are continuously evolving. Our setup is similar to Hsieh et al. (2019)'s experiments for biased-negative learning. Performance was assessed across three benchmarks: MNIST (LeCun et al., 1998), CIFAR10 (Krizhevsky et al., 2014), and 20 Newsgroups (Lang, 1995). These are multiclass datasets, and binary classes were formed by grouping multiple labels. The supplementary materials detail the experimental setup, which we summarize briefly below.

PUc ensures learning tractability via convexity; it is built upon a linear model with Gaussian kernels. We limited our neural networks to only 1 to 3 fully-connected layers of 300 neurons. For MNIST, networks were trained from scratch. Pretrained deep networks encoded the CIFAR10 and 20 Newsgroups datasets into static representations that all learners used. Specifically, 20 Newsgroups documents were encoded into 9,216 dimensional vectors using ELMo (Peters et al., 2018). This encoding scheme was used by Hsieh et al. (2019) and is based on (Rücklé et al., 2018). DenseNet-121 (Huang et al., 2016) encoded the CIFAR10 images into 1,024 dimensional vectors.

Table 1 lists the negative and positive train/test class definitions. Datasets are sampled uniformly at random from their constituent sublabels. For each dataset, there are four experimental conditions, which ordered by row number are: (1) $\text{P}_{\text{train}} = \text{P}_{\text{test}}$, i.e., no bias, (2 & 3 resp.) partially disjoint positive distribution supports without & with prior shift, and (4) disjoint positive class definitions. π_{te} equals P_{test}'s true prior w.r.t. $\text{P}_{\text{test}} \sqcup \text{N}$. By default $\pi_{\text{tr}} = \pi_{\text{te}}$; in the prior shift and disjoint support experiments (rows 3 & 4), π_{tr} equals P_{train}'s true prior w.r.t. $\text{P}_{\text{train}} \sqcup \text{N}$.

Results As shown numerically in Table 1 and graphically in Figure 3, we outperformed PUc and nnPU* across all bias types. The performance gains are not attributable to our use of neural networks as PUc outperformed nnPU* in all but one biased setup (often by a wide margin).

When the positive supports were only partially disjoint (rows 2 and 3 for each dataset), PU2aPNU was the top performer for five of six setups. This pattern reversed when the supports were fully disjoint on challenging datasets where PU2aPNU lagged both PURR and PU2wUU; we explain why this is expected in Section 4.

nnPU* outperformed both PUc and our methods on the unbiased experiments. This is expected; any time an algorithm searches for non-existent phenomena, the additional patterns found will not generalize. MNIST had the largest performance gap; the gap shrunk for CIFAR10 and 20 Newsgroups due to their transfer learning-derived features.

Reducing π_{tr} always improved our algorithms' performance and degraded PUc's. A smaller prior enables easier identification of $\mathcal{X}_{\text{tr-u}}$'s negative examples and in turn a more accurate estimation of $\mathcal{X}_{\text{te-u}}$'s negative risk. In contrast, importance weighting is most accurate in the absence of bias (see row 1 for each dataset). Any shift increases density estimation's (and by extension PUc's) inaccuracy.

6.3. Identical Positive Supports with Bias

This section's experiments mimic situations where the labeled data are complete but non-representative. We follow the experimental setup described in the PUc paper (Sakai & Shimizu, 2019). LIBSVM (Chang & Lin, 2011) benchmarks are used exclusively to ensure suitability with SVM-like PUc; benchmarks "banana," "susy," "ijcnn1," and "a9a" appear in (Sakai & Shimizu, 2019). Our algorithms trained networks of 1 to 2 fully-connected layers.

Sakai & Shimizu's bias operation is based on the median feature vector. Formally, given dataset $\mathcal{X} \subset \mathbb{R}^d$, define c_{med} as the median of set $\{\lVert x - \bar{x} \rVert_2 : x \in \mathcal{X}\}$ where $\lVert\cdot\rVert_2$ is the L_2 (Euclidean) norm and $\bar{x}$ is $\mathcal{X}$'s mean vector. Partition $\mathcal{X}$ into subsets $\mathcal{X}_{\text{lo}} \coloneqq \{x \in \mathcal{X} : \lVert x - \bar{x} \rVert_2 < c_{\text{med}}\}$ and $\mathcal{X}_{\text{hi}} \coloneqq \mathcal{X} \setminus \mathcal{X}_{\text{lo}}$. Examples in $\mathcal{X}_{\text{p}}$ and $\mathcal{X}_{\text{tr-u}}$ are selected from $\mathcal{X}_{\text{lo}}$ with probability $p = 0.9$ and from $\mathcal{X}_{\text{hi}}$ with probability $1 - p$. $p = 0.1$ is used when constructing $\mathcal{X}_{\text{te-u}}$ and the test set. This bias operation simplifies density-ratio estimation since $\forall_x \; w(x) \in \{\frac{1}{9}, 9\}$. Their setting $\pi_{\text{tr}} = \pi_{\text{te}} = 0.5$ also simplifies density estimation.

We modified Sakai & Shimizu's setup such that $\mathcal{X}$ was exclusively the original dataset's positive-valued examples. Negative examples were sampled uniformly at random.

Results Table 2's experiments used the bias procedure described above. PURR outperformed PUc and nnPU* on all experiments and was the best performer on six of ten benchmarks. PU2aPNU outperformed PUc on eight of ten bench-

Table 1. Inductive misclassification rate mean & standard deviation over 100 trials for MNIST, 20 Newsgroups, and CIFAR10 with different positive & negative class definitions. Underlining denotes an improvement versus PUc according to the 5% paired t-test. Boldface indicates a shifted task's best performing method. Negative (N) & positive-test (P_{test}) class definitions are identical for each dataset's first three experiments. Positive train (P_{train}) specified as P_{test} denotes no bias. Additional results are in the supplemental materials.

	N	P_{test}	P_{train}	π_{tr}	π_{te}		Two-Step (PU2)		Baseline	Reference	
						PURR	aPNU	wUU	PUc	nnPU*	PN_{te}
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	P_{test}	0.5	0.5	9.7 (1.5)	10.6 (1.2)	13.7 (1.8)	8.6 (0.7)	5.5 (0.6)	↑
			7, 9	0.5	0.5	9.5 (1.4)	**7.3 (0.8)**	10.2 (1.6)	26.6 (2.4)	36.9 (3.6)	2.7 (0.2)
				0.29	0.5	6.5 (0.8)	**5.0 (0.5)**	6.3 (0.8)	29.2 (2.0)	37.7 (4.0)	↓
	0, 2	1, 3	5, 7	0.5	0.5	3.7 (0.6)	**2.7 (0.5)**	3.3 (0.7)	17.4 (4.8)	34.3 (2.6)	1.2 (0.2)
20 News.	sci, soc, talk	alt, comp, misc, rec	P_{test}	0.56	0.56	15.9 (1.3)	14.7 (1.0)	16.9 (1.7)	14.9 (0.9)	14.6 (1.8)	↑
			misc, rec	0.56	0.56	18.5 (1.9)	**13.9 (1.0)**	17.3 (2.0)	24.1 (2.4)	29.1 (1.2)	10.4 (0.5)
				0.37	0.56	15.1 (1.2)	**12.3 (0.6)**	14.0 (1.0)	28.6 (1.9)	28.4 (1.0)	↓
	misc, rec	soc, talk	alt, comp	0.55	0.46	5.8 (1.2)	7.0 (1.4)	**5.1 (1.8)**	18.3 (4.5)	38.4 (3.4)	1.8 (0.5)
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	P_{test}	0.4	0.4	14.0 (1.0)	14.0 (1.0)	16.0 (1.5)	13.8 (0.7)	11.9 (0.7)	↑
			Plane	0.4	0.4	**13.6 (0.9)**	13.8 (1.1)	15.3 (1.9)	20.4 (1.4)	27.4 (1.1)	9.2 (0.5)
				0.14	0.4	11.6 (0.8)	**11.3 (0.7)**	11.6 (0.8)	26.6 (1.3)	27.5 (1.0)	↓
	Deer, Horse	Plane, Auto	Cat, Dog	0.5	0.5	13.3 (0.9)	14.4 (1.5)	**11.4 (1.4)**	33.2 (2.5)	52.6 (2.1)	7.1 (0.4)

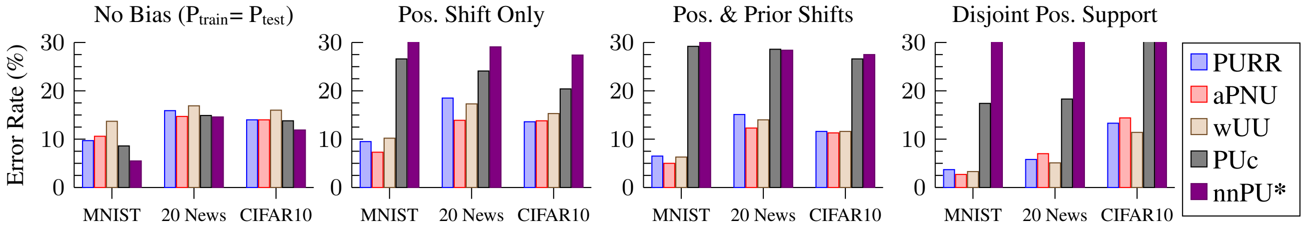


Figure 3. Inductive misclassification rates on MNIST, 20 Newsgroups, and CIFAR10 for our algorithms and the PU baselines. Each graph corresponds to one of the four experimental conditions in Table 1.

marks and was comparable on one other; PU2aPNU was the best performer on four benchmarks. Our methods achieve substantial performance gains over PUc even in an experimental setting designed to work well for PUc. Again, since PUc generally outperforms nnPU*, the gain is not just due to the use of neural networks.

PU2wUU underperformed PU2aPNU and/or PURR on each dataset and was excluded from Table 2. The complete results are in the supplemental materials.

7. Conclusions

We examined arbitrary-positive, unlabeled (aPU) learning, where the labeled (positive) data and target positive class can be arbitrarily different. A (nearly) fixed negative class-distribution allows us to train accurate classifiers without any labeled data from the target distribution (i.e., disjoint positive supports). It is an open question whether the fixed negative-class distribution requirement can be confidently loosened. Empirical results (in the supplemental materials) indicate that even without statistical guarantees, our methods are still robust to some shift. Future works seeks a less restrictive yet statistically sound replacement assumption.

Table 2. Inductive misclassification rate mean & standard deviation over 100 trials with positive-only Sakai & Shimizu (2019) biasing. Underlining denotes improvement vs. PUc based on the 5% paired t-test. Boldface indicates the best performing method. $n_{\text{p}} = 300$ & $n_{\text{tr-u}} = n_{\text{te-u}} = 700$. Complete table in supplemental materials.

Dataset	PURR	PU2aPNU	PUc	nnPU*
banana	13.3 (2.4)	**12.2 (1.6)**	17.5 (3.9)	29.8 (4.1)
cod-rna	12.9 (2.9)	**12.6 (4.2)**	26.1 (5.2)	28.6 (3.9)
susy	**23.7 (1.8)**	27.2 (2.6)	27.2 (4.0)	46.9 (3.8)
ijcnn1	**21.5 (2.6)**	28.6 (3.6)	23.4 (3.5)	31.3 (3.4)
covtype.b	**28.9 (2.7)**	31.7 (3.3)	39.1 (3.6)	54.4 (3.2)
phishing	11.6 (2.1)	**9.3 (1.0)**	13.3 (3.5)	22.9 (3.8)
a9a	26.1 (1.8)	**25.4 (1.7)**	32.9 (2.5)	33.4 (1.9)
connect4	**33.0 (3.1)**	33.2 (2.8)	37.0 (2.9)	45.7 (3.0)
w8a	**15.2 (1.9)**	19.8 (3.0)	31.0 (7.5)	41.3 (4.3)
epsilon	**32.1 (3.0)**	38.7 (7.5)	63.5 (6.5)	64.5 (1.5)

References

Bekker, J., Robberechts, P., and Davis, J. Beyond the selected completely at random assumption for learning from positive and unlabeled data. *Proceedings of the 2019 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD)*, 2019.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

du Plessis, M., Niu, G., and Sugiyama, M. Convex formulation for learning from positive and unlabeled data. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1386–1394, Lille, France, 07–09 Jul 2015.

du Plessis, M. C., Niu, G., and Sugiyama, M. Analysis of learning from positive and unlabeled data. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NeurIPS'14, 2014.

du Plessis, M. C., Niu, G., and Sugiyama, M. Class-prior estimation for learning from positive and unlabeled data. *Machine Learning*, 106(4):463–492, 2017.

Elkan, C. and Noto, K. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pp. 213–220, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4.

Hido, S., Tsuboi, Y., Kashima, H., Sugiyama, M., and Kanamori, T. Inlier-based outlier detection via direct density ratio estimation. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pp. 223–232, 2008. ISBN 978-0-7695-3502-9.

Hsieh, Y.-G., Niu, G., and Sugiyama, M. Classification from positive, unlabeled and biased negative data. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2820–2829, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Huang, G., Liu, Z., and Weinberger, K. Q. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2016.

Kato, M., Teshima, T., and Honda, J. Learning from positive and unlabeled data with a selection bias. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Kiryo, R., Niu, G., du Plessis, M. C., and Sugiyama, M. Positive-unlabeled learning with non-negative risk estimator. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NeurIPS'17, pp. 1674–1684, 2017. ISBN 978-1-5108-6096-4.

Krizhevsky, A., Nair, V., and Hinton, G. The CIFAR-10 dataset, 2014.

Lang, K. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 331–339, 1995.

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

Li, W., Guo, Q., and Elkan, C. A positive and unlabeled learning algorithm for one-class classification of remote-sensing data. *Geoscience and Remote Sensing, IEEE Transactions on*, 49:717 – 725, 2011 2011.

Loshchilov, I. and Hutter, F. Fixing weight decay regularization in Adam. *CoRR*, abs/1711.05101, 2017. URL http://arxiv.org/abs/1711.05101.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X.

Neves, M., Marinho-Dias, J., Ribeiro, J., and Sousa, H. Epstein-barr virus strains and variations: Geographic or disease-specific variants? *Journal of Medical Virology*, 89(3):373–387, 2017.

Niu, G., du Plessis, M. C., Sakai, T., Ma, Y., and Sugiyama, M. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NeurIPS'16, pp. 1207–1215, 2016.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of NAACL*, 2018.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN 0262170051.

Ramaswamy, H. G., Scott, C., and Tewari, A. Mixture proportion estimation via kernel embedding of distributions. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 2052–2060. JMLR.org, 2016.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of Adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018*, 2018.

Rennie, J. 20 newsgroups. http://qwone.com/~jason/20Newsgroups/, 2001.

Rücklé, A., Eger, S., Peyrard, M., and Gurevych, I. Concatenated power mean embeddings as universal cross-lingual sentence representations. *arXiv*, 2018. URL https://arxiv.org/abs/1803.01400.

Sakai, T. and Shimizu, N. Covariate shift adaptation on learning from positive and unlabeled data. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pp. 4838–4845, 2019.

Sakai, T., du Plessis, M. C., Niu, G., and Sugiyama, M. Semi-supervised classification based on classification from positive and unlabeled data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2998–3006, 2017.

Scott, C. and Blanchard, G. Novelty detection: Unlabeled data definitely help. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 464–471, Clearwater Beach, Florida, 16–18 Apr 2009.

Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244, Oct 2000.

Sugiyama, M., Krauledat, M., and Müller, K.-R. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, Dec. 2007.

Sugiyama, M., Suzuki, T., and Kanamori, T. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, USA, 1st edition, 2012.

Xiao, P., Li, J., Fu, G., Zhou, Y., Huan, X., and Yang, H. Geographic distribution and temporal trends of HIV-1 subtypes through heterosexual transmission in China: A systematic review and meta-analysis. *International Journal of Environmental Research and Public Health*, pp. 830, July 2017.

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

Yang, P., Li, X.-L., Mei, J.-P., Kwoh, C.-K., and Ng, S.-K. Positive-unlabeled learning for disease gene identification. *Bioinformatics*, 28(20):2640–2647, 08 2012.

Learning from Positive and Unlabeled Data with Arbitrary Positive Shift: Supplementary Material

A. Nomenclature

Table 3: aPU nomenclature

PN	Positive-negative learning, i.e., ordinary supervised classification
uPU	Unbiased Positive-Unlabeled risk estimator from (du Plessis et al., 2014). See Section 2
nnPU	Non-negative Positive-Unlabeled learner from (Kiryo et al., 2017). See Section 2
aPU	Arbitrary-positive, unlabeled learning where the positive training data may be arbitrarily different from the target application's positive-class distribution
bPU	Biased-positive, unlabeled learning is a constrained version of aPU learning where the labeled data shift is assumed to take a specific form
PUc	Positive-Unlabeled Covariate shift algorithm from (Sakai & Shimizu, 2019). See Section 3
PU2wUU	Positive-Unlabeled to Weighted Unlabeled-Unlabeled (two-step) aPU learner. See Section 4
PU2aPNU	Positive-Unlabeled to Arbitrary-Positive, Negative, Unlabeled (two-step) aPU learner. See Section 4
PURR	Positive-Unlabeled Recursive Risk (one-step) aPU estimator. See Section 5
X	Covariate where $X \in \mathbb{R}^d$
Y	Dependent random variable, i.e., label, where $Y \in \{\pm 1\}$
$\hat{y}$	Predicted label $\hat{y} \in \{\pm 1\}$
g	Decision function, $g : \mathbb{R}^d \to \mathbb{R}$
θ	Parameter(s) of decision function g
$\mathcal{G}$	Real-valued decision function hypothesis class, i.e., $g \in \mathcal{G}$
ℓ	Loss function, $\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$
$p_{\mathcal{D}}(x, y)$	Joint distribution, where $\mathcal{D} \in \{\text{tr}, \text{te}\}$ for train and test resp.
$\pi_{\mathcal{D}}$	Positive-class prior probability, $\pi_{\mathcal{D}} \coloneqq p_{\mathcal{D}}(Y = +1)$ where $\mathcal{D} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{D}\text{-p}}(x)$	Positive class-conditional $p_{\mathcal{D}\text{-p}}(x) = p_{\mathcal{D}}(x \mid Y = +1)$ where $\mathcal{D} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{D}\text{-n}}(x)$	Negative class-conditional $p_{\mathcal{D}\text{-n}}(x) = p_{\mathcal{D}}(x \mid Y = -1)$ where $\mathcal{D} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{D}\text{-u}}(x)$	Marginal distribution where $p_{\mathcal{D}\text{-u}}(x) = p_{\mathcal{D}}(x)$ where $\mathcal{D} \in \{\text{tr}, \text{te}\}$ for train and test resp.
$\mathcal{X}_{\text{p}}$	Labeled (positive) dataset, i.e., $\mathcal{X}_{\text{p}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-p}}(x)$
$\mathcal{X}_{\text{tr-u}}$	Unlabeled dataset sampled from *training* marginal distribution, i.e., $\mathcal{X}_{\text{tr-u}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)$
$\mathcal{X}_{\text{te-u}}$	Unlabeled dataset sampled from *test* marginal distribution, i.e., $\mathcal{X}_{\text{te-u}} \overset{\text{i.i.d.}}{\sim} p_{\text{te-u}}(x)$
$\hat{\sigma}$	Probabilistic classifier, $\hat{\sigma} : \mathbb{R}^d \to [0, 1]$ that approximates $p_{\text{tr}}(Y = -1 \mid x)$
$\widehat{\Sigma}$	Function class containing $\hat{\sigma}$
$\widetilde{\mathcal{X}}_{\text{n}}$	Surrogate negative set formed by reweighting $\mathcal{X}_{\text{tr-u}}$ by $\hat{\sigma}$
$R(g)$	Expected risk for decision function g and loss ℓ, i.e., $R(g) \coloneqq \mathbb{E}_{(X,Y) \sim p(x,y)}[\ell(Yg(X))]$
$\widehat{R}(g)$	Empirical estimate of expected risk $R(g)$
$\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$	Empirical risk when predicting label $\hat{y} \in \{\pm 1\}$ on data sampled from some distribution, $p_{\mathcal{D}}(x)$
$\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$	Surrogate negative risk formed by weighting unlabeled set $\mathcal{X}_{\text{tr-u}}$ by probabilistic classifier $\hat{\sigma}$, where $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g) \coloneqq \frac{1}{n_{\text{tr-u}}} \sum_{x_i \in \mathcal{X}_{\text{tr-u}}} \frac{\hat{\sigma}(x_i)\ell(\hat{y}g(x_i))}{1 - \pi_{\text{tr}}}$
$w(x)$	Covariate shift importance function based on density-ratio estimation, where $w(x) \coloneqq \frac{p_{\text{te-u}}(x)}{p_{\text{tr-u}}(x)}$
n_{p}	Size of the labeled (positive) dataset, i.e., $n_{\text{p}} \coloneqq \lvert \mathcal{X}_{\text{p}} \rvert$
$n_{\text{tr-u}}$	Size of the unlabeled *training* dataset, i.e., $n_{\text{tr-u}} \coloneqq \lvert \mathcal{X}_{\text{tr-u}} \rvert$
$n_{\text{te-u}}$	Size of the unlabeled *test* dataset, i.e., $n_{\text{te-u}} \coloneqq \lvert \mathcal{X}_{\text{te-u}} \rvert$
n_{Test}	Size of the (inductive) test set
$\mathcal{A}$	Learning or optimization algorithm

Table 3: aPU nomenclature (continued)

γ	Non-negative gradient attenuator hyperparameter $\gamma \in (0, 1]$
η	Learning rate hyperparameter, $\eta > 0$
λ	Weight decay hyperparameter, $\lambda \geq 0$
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{I}_m)$	Multivariate Gaussian (normal) distribution with mean $\boldsymbol{\mu}$ and m-dimensional identity covariance
$[a]_+$	$= \max\{0, a\}$

B. Proofs

B.1. Proof of Theorem 1

Proof. Consider first the case that $\hat{\sigma}(x) = p_{\text{tr}}(Y = -1|x)$:

$$\begin{aligned}
\mathbb{E}_{\mathcal{X}_{\text{tr-u}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)}\left[\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)\right] &= \mathbb{E}_{\mathcal{X}_{\text{tr-u}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)}\left[\frac{1}{n_{\text{tr-u}}} \sum_{X_i \in \mathcal{X}_{\text{tr-u}}} \frac{\ell(\hat{y}g(X_i))\hat{\sigma}(X_i)}{1 - \pi_{\text{tr}}}\right] \\
&= \frac{1}{n_{\text{tr-u}}} \sum_{i=1}^{n_{\text{tr-u}}} \mathbb{E}_{X \sim p_{\text{tr-u}}(x)}\left[\frac{\ell(\hat{y}g(X))\hat{\sigma}(X)}{1 - \pi_{\text{tr}}}\right] && \text{Linearity of expectation} \\
&= \mathbb{E}_{X \sim p_{\text{tr-u}}(x)}\left[\frac{\ell(\hat{y}g(X))\hat{\sigma}(X)}{1 - \pi_{\text{tr}}}\right] \\
&= \mathbb{E}_{X \sim p_{\text{tr-u}}(x)}\left[\frac{\ell(\hat{y}g(X))p_{\text{tr}}(Y = -1|X)}{p_{\text{tr}}(Y = -1)}\right] \\
&= \int_x \ell(\hat{y}g(x))\frac{p_{\text{tr}}(Y = -1|x)p_{\text{tr-u}}(x)}{p_{\text{tr}}(Y = -1)} \\
&= \mathbb{E}_{X \sim p_{\text{tr-n}}(x)}[\ell(\hat{y}g(X))] && \text{Bayes' Rule} \\
&=: R^{\hat{y}}_{\text{tr-n}}(g),
\end{aligned}$$

satisfying the definition of unbiased.

Next we consider whether $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ is a consistent estimator of $R^{\hat{y}}_{\text{n}}(g)$. For the complete definition of PAC learnability that we use here, see (Mohri et al., 2012). We provide a brief sketch of the definition below.

We assume that true posterior distribution, $p_{\text{tr}}(Y = -1|x)$ is in some concept class $\mathcal{C}$ of functions — i.e., *concepts* — mapping $\mathbb{R}^d$ to $[0, 1]$. Let $\hat{\sigma}_{\mathcal{S}} \in \widehat{\Sigma}$ be the hypothesis selected by learning algorithm $\mathcal{A}$ after being provided a training sample $\mathcal{S}$ of size $n = \min\{n_{\text{p}}, n_{\text{tr-u}}\}$.[4] Consider the *realizable* setting so $\mathcal{C}$'s PAC learnability entails that for all $\epsilon, \delta > 0$, there exists an n' such that for all $n > n'$,

$$\Pr\left[\mathbb{E}_{X \sim p_{\text{tr-u}}(x)}[|\hat{\sigma}_{\mathcal{S}}(X) - p_{\text{tr}}(Y = -1|X)|] > \epsilon\right] < \delta. \tag{12}$$

Therefore, as $n \to \infty$, $\hat{\sigma}$'s expected (absolute) error w.r.t. $p_{\text{tr}}(Y = -1|x)$ decreases to 0 making $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ asymptotically unbiased. To demonstrate consistency, it is necessary to show that for all $\epsilon > 0$:

$$\lim_{n \to \infty} \Pr\left[\left|\widetilde{R}^{\hat{y}}_{\text{n-u}}(g) - R^{\hat{y}}_{\text{tr-n}}(g)\right| > \epsilon\right] = 0.$$

Let $\sup_{|t| \leq \|g\|_\infty} \ell(\hat{y}t) \leq C_\ell$, where $\|g\|_\infty$ is the Chebyshev norm of g for $x \in \mathbb{R}^d$. Bounding the loss's magnitude bounds the variance when estimating the surrogate negative risk of $X \sim p_{\text{tr-u}}(x)$ such that $\frac{1}{(1-\pi_{\text{tr}})^2}\text{Var}(\hat{\sigma}(X)\ell(\hat{y}g(X))) \leq C_{\text{var}}$ where $C_{\text{var}} \in \mathbb{R}_{\geq 0}$ and $\pi_{\text{tr}} \in [0, 1)$.

[4] No restrictions are placed on $\mathcal{A}$ other than its existence and that selected hypothesis $\hat{\sigma}_{\mathcal{S}}$ satisfies Eq. (12).

Since $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ is asymptotically unbiased, then from Chebyshev's inequality for $\epsilon > 0$:

$$\begin{aligned}
\lim_{n \to \infty} \Pr\left[\left|\widetilde{R}^{\hat{y}}_{\text{n-u}}(g) - R^{\hat{y}}_{\text{tr-n}}(g)\right| \geq \epsilon\right] &\leq \frac{\text{Var}\Big(\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)\Big)}{\epsilon^2} \\
&= \frac{1}{(1-\pi_{\text{tr}})^2\epsilon^2} \sum_{i=1}^{n_{\text{tr-u}}} \text{Var}\left(\frac{\hat{\sigma}(X)\ell(\hat{y}g(X))}{n_{\text{tr-u}}}\right) && \text{Linearity of independent r.v. variance} \\
&\leq \frac{n_{\text{tr-u}}C_{\text{var}}}{n_{\text{tr-u}}^2\epsilon^2} \\
&= 0 && \text{L'Hôpital's Rule.}
\end{aligned}$$

□

B.2. Proof Regarding Estimating π_{te}

No technique has previously been proposed to directly estimate the test distribution's positive prior π_{te} given only $\mathcal{X}_{\text{p}}$, $\mathcal{X}_{\text{tr-u}}$, and $\mathcal{X}_{\text{te-u}}$.

Theorem 3. *Define $\mathcal{X}_{\text{u}} := \{x_i\}_{i=1}^{n_{\text{u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{u}}(x)$. Let $\mathcal{X}_{\text{n}} = \{x_i \in \mathcal{X}_{\text{u}} : Q_i = 1\}$ be a set where Q_i is a Bernoulli random variable with probability of success $q_i = p(Y = -1|x_i)$. Then $\mathcal{X}_{\text{n}}$ is a SCAR sample w.r.t. negative distribution $p_{\text{n}}(x) = p(x|Y = -1)$.*

Proof. By Bayes' Rule

$$p_{\text{n}}(x) \propto p(Y = -1|x)p_{\text{u}}(x)$$

Each $x_i \in \mathcal{X}_{\text{u}}$ is sampled from $p_{\text{u}}(x)$. By including x_i in $\mathcal{X}_{\text{n}}$ only if $Q_i = 1$, then x_i's effective sampling probability is $p(Y = -1|x_i)p(x)$. Bayes' Rule includes prior inverse $\frac{1}{1-\pi}$, where $\pi = p(Y = +1)$; this constant scalar can be ignored since it does not change whether $\mathcal{X}_{\text{n}}$ is unbiased, i.e., it does not affect relative probability. □

Commentary Theorem 3 states the property generally, but consider it over aPU's training distribution. Probabilistic classifier $\hat{\sigma}$ is used as a surrogate for $p_{\text{tr}}(Y = -1|x)$. Rather than *soft* weighting the samples like in Theorem 1's proof, sample inclusion in the negative set is a *hard* "in-or-out" decision. This does not change the sample's statistical properties, but it allows us to create an unweighted negative set, we denote $\mathcal{X}_{\text{tr-n}}$.

By Eq. (5)'s assumption, $\mathcal{X}_{\text{tr-n}}$ is representative of samples from the negative distribution $p_{\text{n}}(x) = p_{\text{tr-n}}(x) = p_{\text{te-n}}(x)$. Given a representative labeled set from the *test distribution*, well-known positive-unlabeled prior estimation techniques (du Plessis et al., 2017; Ramaswamy et al., 2016) can be used without modification using $\mathcal{X}_{\text{tr-n}}$ and $\mathcal{X}_{\text{te-u}}$. Be aware that these PU prior estimation methods would return the negative-class's prior, $p_{\text{te}}(Y = -1)$, while our risk estimators use the positive class's prior, $\pi_{\text{te}} = 1 - p_{\text{te}}(Y = -1)$.

We provide empirical results regarding the effect of inaccurate prior estimation's in Section E.4.

B.3. Proof of Theorem 2

Proof. Consider first whether PURR is unbiased. du Plessis et al. (2014) observe that the negative labeled risk can be found via decomposition where

$$(1-\pi)R^{\hat{y}}_{\text{n}}(g) = R^{\hat{y}}_{\text{u}}(g) - \pi R^{\hat{y}}_{\text{p}}(g). \tag{13}$$

The positive labeled risk similarly decomposes as

$$\pi R^{\hat{y}}_{\text{p}}(g) = R^{\hat{y}}_{\text{u}}(g) - (1-\pi)R^{\hat{y}}_{\text{n}}(g). \tag{14}$$

Applying these decompositions along with Eq. (5)'s assumption yields an unbiased version of PURR:

$$\widehat{R}_{\text{uPURR}}(g) = \underbrace{\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\underbrace{\frac{\widehat{R}^{+}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{+}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}}_{\widehat{R}^{+}_{\text{te-n}}(g)}}_{\pi_{\text{te}}\widehat{R}^{+}_{\text{te-p}}(g)} + (1-\pi_{\text{te}})\underbrace{\frac{\widehat{R}^{-}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{-}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}}_{\widehat{R}^{-}_{\text{te-n}}(g)}. \tag{15}$$

Since $\forall_t \ell(t) \geq 0$, it always holds that labeled risk $R^{\hat{y}}_{\mathcal{D}}(g) \geq 0$. When using risk decomposition (i.e., Eqs. (13) and (14)) to empirically estimate a labeled risk, it can occur that $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g) < 0$. Non-negativity correction prevents these demonstrably invalid risk estimates using a simple $\max$ operation. For brevity, denote $\max\{0, \cdot\}$ as $[\cdot]_{+}$. The unrolled definition of the PURR risk estimator with non-negativity correction is:

$$\widehat{R}_{\text{PURR}}(g) = \underbrace{\left[\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\left[\underbrace{\frac{\widehat{R}^{+}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{+}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}}_{\widehat{R}^{+}_{\text{te-n}}(g)}\right]_{+}\right]_{+}}_{\pi_{\text{te}}\widehat{R}^{+}_{\text{te-p}}(g)} + (1-\pi_{\text{te}})\left[\underbrace{\frac{\widehat{R}^{-}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{-}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}}_{\widehat{R}^{-}_{\text{te-n}}(g)}\right]_{+}. \tag{16}$$

Clearly, $\widehat{R}_{\text{PURR}}(g) \geq \widehat{R}_{\text{uPURR}}(g)$. For $\widehat{R}_{\text{PURR}}(g)$ to be unbiased, equality must strictly hold, i.e., $[\cdot]_{+}$ has no effect. This only occurs if non-negativity correction is never needed, i.e., has probability 0 of occurring.

Next consider whether PURR is consistent. Formally, an estimator, $\hat{\theta}_n$, over n samples is consistent w.r.t. parameter θ if for all $\epsilon > 0$ it holds that

$$\lim_{n\to\infty} \Pr\left[\left|\hat{\theta}_n - \theta\right| \geq \epsilon\right] = 0.$$

For consistency to hold, we make mild assumptions about the behavior of the loss and decision functions; the following conditions are identical to those assumed by Kiryo et al. (2017). Define loss function ℓ as *bounded* over some class of real-valued functions $\mathcal{G}$ (where $g \in \mathcal{G}$) when the following conditions both hold:

1. $\exists C_g > 0$ such that $\sup_{g\in\mathcal{G}} \|g\|_{\infty} \leq C_g$
2. $\exists C_\ell > 0$ such that $\sup_{|t|\leq C_g} \max_{\hat{y}\in\{\pm 1\}} \ell(\hat{y}t) \leq C_\ell$.

Let estimator $\hat{Y} = \sum_{i=1}^{k} \beta_i \hat{\theta}_{(i)}$ be the weighted sum of k consistent estimators with each constant $|\beta_i| \in \mathbb{R}_{>0}$. Let $\epsilon > 0$ be an arbitrary positive constant. If each $\hat{\theta}_{(i)}$ converges to within $\frac{\epsilon}{k|\beta_i|} > 0$ of $\theta_{(i)} \geq 0$, then $\hat{Y}$ converges to within ϵ of $\sum_{i=1}^{k} \beta_i \theta_{(i)}$. Therefore, to prove the consistency of $\widehat{R}_{\text{PURR}}(g)$, it suffices to show that each of its individual terms is consistent. $\widehat{R}_{\text{PURR}}(g)$'s terms partition into one of two categories: those estimated directly from training data and those that require non-negativity correction. We examine each category separately.

First, consider when labeled risk $R^{\hat{y}}_{\mathcal{D}}(g)$ is empirically estimated directly from training data set, $\mathcal{X} \overset{\text{i.i.d.}}{\sim} p_{\mathcal{D}}(x)$. For each (independent) $X \sim p_{\mathcal{D}}(x)$, $\ell(\hat{y}g(X))$ is an unbiased estimate of $R^{\hat{y}}_{\mathcal{D}}(g)$. In addition, $\ell(\hat{y}g(X)) < C_\ell < \infty$ implies that $\text{Var}(\ell(\hat{y}g(X))) < \infty$. By Chebyshev's Inequality, $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$ is consistent as

$$\lim_{|\mathcal{X}|\to\infty} \Pr\left[\left|\frac{1}{|\mathcal{X}|}\sum_{x_i\in\mathcal{X}}\Big(\ell(\hat{y}g(x_i))\Big) - R^{\hat{y}}_{\mathcal{D}}(g)\right| \geq \epsilon\right] < \frac{\text{Var}(\ell(\hat{y}g(X)))}{|\mathcal{X}|\epsilon^2} = 0.$$

Note that $n_{\text{p}}, n_{\text{tr-u}}, n_{\text{te-u}} \to \infty$ since each is directly used to empirically estimate some labeled risk in Eq. (16).

Last, we examine the consistency of risk estimators to which non-negativity correction is applied. Start with the base case where the correction is applied to two terms estimated directly from training data. We showed above that the sum of such estimators is consistent so it suffices to prove that

$$\lim_{n\to\infty} \Pr\left[\left|\left[\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)\right]_{+} - R^{\hat{y}}_{\mathcal{D}}(g)\right| > \epsilon\right] = 0$$

where $[a]_{+} = \max\{a, 0\}$ and n is the size of the smaller dataset used in the decomposition of $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$. Because $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$ is consistent, then as $n \to \infty$ it holds that $R^{\hat{y}}_{\mathcal{D}}(g) - \epsilon \leq \widehat{R}^{\hat{y}}_{\mathcal{D}}(g) \leq R^{\hat{y}}_{\mathcal{D}}(g) + \epsilon$. When $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g) \geq R^{\hat{y}}_{\mathcal{D}}(g) \geq 0$, non-negativity correction (i.e., $[\cdot]_{+}$) has no effect. If $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g) < R^{\hat{y}}_{\mathcal{D}}(g)$, non-negativity correction either has no effect or it increases $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$ which strictly decreases the estimation error. These two conditions cover all possible values of $R^{\hat{y}}_{\mathcal{D}}(g)$ so the base case is consistent.

Consider the more general case when one (or more) of the terms to which non-negativity is applied was recursively estimated via decomposition. It is easy to see via induction that such nested terms are themselves consistent allowing us to apply the same logic as the non-negative base case above. Therefore, all non-negative terms are consistent. □

C. Two-Step Learning Empirical Risk Minimization Algorithm

Algorithm 3 shows the ERM framework for the wUU and aPNU risk estimators (see Section 4); the algorithm learns parameters θ for decision function g. The non-negativity correction occurs whenever $\widehat{R}^{+}_{\text{te-u}}(g) - (1 - \pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g) < 0$ (see line 7). The basic algorithm is heavily influenced by the stochastic optimization algorithm proposed by Kiryo et al. (2017).

Algorithm 3 wUU and aPNU ERM framework to learn g

Input: Data $(\mathcal{X}_{\text{p}}, \widetilde{\mathcal{X}}_{\text{n}}, \mathcal{X}_{\text{te-u}})$, hyperparameters (γ, η) and risk estimator $\widehat{R}_{\text{TS}}(g) \in \{\widehat{R}_{\text{wUU}}(g), \widehat{R}_{\text{aPNU}}(g)\}$
Output: Decision function g's parameters θ

```
 1: Select SGD-like optimization algorithm $\mathcal{A}$
 2: while Stopping criteria not met do
 3:   Shuffle $(\mathcal{X}_{\text{p}}, \widetilde{\mathcal{X}}_{\text{n}}, \mathcal{X}_{\text{te-u}})$ into $N$ batches
 4:   for each minibatch $(\mathcal{X}^{(i)}_{\text{p}}, \widetilde{\mathcal{X}}^{(i)}_{\text{n}}, \mathcal{X}^{(i)}_{\text{te-u}})$ do
 5:     if $\widehat{R}^{+}_{\text{te-u}}(g) - (1 - \pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g) < 0$ then
 6:       Set gradient $-\nabla_{\theta}\left(\widehat{R}^{+}_{\text{te-u}}(g) - (1 - \pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g)\right)$
 7:       Update $\theta$ by $\mathcal{A}$ with attenuated learning rate $\gamma\eta$
 8:     else
 9:       Set gradient $\nabla_{\theta}\widehat{R}_{\text{TS}}(g)$
10:       Update $\theta$ by $\mathcal{A}$ with default learning rate $\eta$
11: return $\theta$ minimizing validation loss
```

Algorithm 3 terminates after a fixed epoch count (see Table 7 for the number of epochs used for each dataset). Although not shown in Algorithm 3, the validation loss is measured at the end of each epoch. The algorithm returns the model parameters with the lowest validation loss.

D. Detailed Experimental Setup

This section details the experiment setup used to collect the results in Section 6.

D.1. Reproducing our Experiments

Our implementation is written and tested in Python 3.6.5 using the `PyTorch` neural network framework version 1.3.1. The source code is available at: https://github.com/ZaydH/arbitrary_positive_unlabeled. The repository includes file `requirements.txt` that details Python package dependency information.

To run the program, invoke:

```
python driver.py ConfigFile
```

where `ConfigFile` is a `yaml`-format text file specifying the experimental setup. We provide configuration files for Section 6's experiments in repository folder “`src/configs`”. Prior probability shifts can be made by manually modifying the configuration files.

Datasets Our program automatically retrieves all necessary data. Synthetic data is generated by the program itself. Otherwise the dataset is downloaded automatically from the web. If you have trouble downloading any datasets, please verify that your network/firewall ports are properly configured.

D.2. Class Definitions

D.2.1. Partially and Fully Disjoint Positive Distribution Supports

Section 6.2's experimental setups are very similar to Hsieh et al. (2019)'s experiments for biased-negative learning. We even follow Hsieh et al.'s label partitions. The basic rationale motivating the splits are:

- **MNIST**: Odd (positive class) vs. even (negative class) digits. Each digit's frequency in the original dataset is approximately 0.1 making each class's target prior $5 * 0.1 = 0.5$.

- **20 Newsgroups**: As its name indicates, the 20 Newsgroups dataset consists of 20 disjoint labels. Categories are formed by partitioning those 20 labels into 7 groups based on the corresponding text document's general theme. Our classes are formed by splitting the categories into two disjoint sets. Specifically, the positive-test class consists of documents with labels 0 to 10 in the original dataset. The negative class is comprised of documents whose labels in the original dataset are 11-19. This split's actual positive prior probability is approximately 0.56.[5]

- **CIFAR10**: Inanimate objects (positive class) vs. animals (negative class). CIFAR10 is a multiclass dataset with ten labels. Each label is equally common in the training and test set, i.e., has prior 0.1. Since CIFAR10's positive-test class has exactly four labels (e.g., plane, automobile, truck, and ship), the positive-test prior is $4 * 0.1 = 0.4$.

The distribution shift between train and test is premised on new subclasses emerging in the test distribution (e.g., due to novel adversarial attacks or systematic failure to collect data on a positive subpopulation in the original dataset).

D.2.2. Identical Positive Supports with Bias

Table 4 defines the positive and negative classes for each LIBSVM dataset. Label “$+1$” always corresponded to the positive class. In two-class (binary) datasets, the other label was the negative class. For multiclass datasets (e.g., connect4), whichever other class had the most examples was used as the negative class.

Table 4. Positive & negative class definitions for the LIBSVM datasets

Dataset	d	Pos. Class	Neg. Class
banana	2	+1	2
cod-rna	8	+1	-1
susy	18	+1	0
ijcnn1	22	+1	-1
covtype.binary	54	+1	2
phishing	68	+1	0
a9a	123	+1	-1
connect4	126	+1	-1
w8a	300	+1	-1
epsilon	2,000	+1	-1

D.3. Training, Validation, and Test Set Sizes

Table 5 lists the default positive, unlabeled, and inductive test set sizes used for each dataset. All LIBSVM datasets (e.g., susy, a9a, etc.) used the dataset sizes defined by Sakai & Shimizu (2019). The validation set was one-fifth Table 5's training set sizes. Each learner observed identical dataset splits in each trial.

[5] We used the latest version of the dataset with duplicates and cross-posts removed.

Table 5. Default training set sizes for each dataset. LIBSVM denotes all datasets downloaded directly from (Chang & Lin, 2011). These quantities do not include the validation set.

Dataset	n_p	$n_\text{tr-u}$	$n_\text{te-u}$	n_Test
MNIST	1,000	5,000	5,000	5,000
20 Newsgroups	500	3,000	3,000	5,000
CIFAR10	1,000	5,000	5,000	3,000
LIBSVM	250	583	583	2,000

Special inductive test set sizes were needed for two of Section 6.2's disjoint positive-support experiments. To understand why, consider the MNIST disjoint-support experiment (i.e., the fourth MNIST row) where the negative class (N) is comprised of labels $\{0, 2\}$ and the positive-test class (P_test) is composed of labels $\{1, 3\}$. Each label has approximately 1,000 examples in the dedicated test set meaning there are approximately 4,000 total test examples between the negative and positive classes. However, MNIST's default inductive test set size (n_Test) is 5,000 (see Table 5). Rather than duplicating test set examples, we reduced MNIST's n_Test to 1,500 *for the disjoint positive-support experiments only*. 20 Newsgroups has the same issue so its disjoint-positive support n_Test was also reduced as specified in Table 6. To be clear, for all other datasets and experimental setups in Sections 6.2 and 6.3, Table 5 applies.

Table 6. Smaller MNIST and 20 Newsgroups inductive test set sizes, i.e., n_Test, used in the disjoint-support experiments.

Dataset	n_Test
MNIST	3,000
20 Newsgroups	1,500

MNIST, 20 Newsgroups, and CIFAR10 have predefined test sets, which we exclusively used to collect the inductive results. They were not used for training or validation. Only some LIBSVM datasets have dedicated test sets, and for those that do, Sakai & Shimizu (2019) do not specify whether the test set was held out in their experiments. When applicable, we merge the LIBSVM train and test datasets together as if there was only a single monolithic training set. $\mathcal{X}_\text{p}$, $\mathcal{X}_\text{tr-u}$, $\mathcal{X}_\text{te-u}$ and the inductive test set are independently sampled at random from this monolithic set without replacement.

Since the PUc formulation is convex, Sakai & Shimizu train their final model on the combined training and validation set.

D.4. CIFAR10 Image Representation

Each CIFAR10 (Krizhevsky et al., 2014) image is 32 pixels by 32 pixels with three (RGB) color channels (3,072 dimensions total). PUc specifies a convex model so it cannot be used to train (non-convex) deep convolutional networks directly. To ensure a meaningful comparison, we leveraged the DenseNet-121 deep convolutional network architecture pretrained on 1.2 million images from ImageNet (Huang et al., 2016). The network's (linear) classification layer was removed, and the experiments used the 1,024-dimension feature vector output by the convolutional backbone.

D.5. 20 Newsgroups Document Representation

The 20 Newsgroups dataset is a collection of internet discussion board posts. The original dataset consisted of 20,000 documents (Lang, 1995); it was pruned to 18,828 documents in 2007 after removal of duplicates and cross-posts (Rennie, 2001). This latest dataset has a predefined split of 11,314 train and 7,532 test documents. Similar to CIFAR10, we use transfer learning to create a richer representation of each document.

Classic word embedding models like GloVe and Word2Vec yield token representations that are independent of context. Proposed by Peters et al. (2018), ELMo (embeddings for language models) enhances classic word embeddings by making the token representations context dependent. We use ELMo to encode each 20 Newsgroup document as described below.

ELMo's embedder consists of three sequential layers — first a character convolutional neural network (CNN) provides subword information and improves unknown word robustness. The CNN's output is then fed into a two-layer, bidirectional LSTM. The output from each of ELMo's layers is a 1,024-dimension vector. For a token stream of length m, the output of ELMo's embedder would be a tensor of size $\langle \text{\#Layers} \times d_\text{layer} \times \text{\#Tokens} \rangle$ — in this case $\langle 3 \times 1024 \times m \rangle$.

Like Hsieh et al. (2019) who used this encoding scheme for biased-negative learning, we used Rücklé et al. (2018)'s sentence representation encoding scheme, which takes the minimum, maximum, and average value along each ELMo layer's output dimension. The dimension of the resulting document encoding is:

$$|\{\max, \min, \text{avg}\}| \cdot \text{\#Layers} \cdot d_{\text{layer}} = 3 \cdot 3 \cdot 1024 = 9{,}216.$$

When documents are encoded serially, each document implicitly contains information about all proceeding documents. Put simply, the order documents are processed affects each document's final encoding. For consistency, all 20 Newsgroups experiments used a single identical encoding for all learners.

The Allen Institute for Artificial Intelligence has published multiple pretrained ELMo models. We used the ELMo model trained on a 5.5 billion token corpus — 1.9 billion from Wikipedia and 3.6 billion from a news crawl. We chose this version because ELMo's developers report that it was the best performing.

D.6. Models and Hyperparameters

This section reviews our experiments' hyperparameter methodology. PUc's author-supplied implementation includes a built-in hyperparameter tuning architecture, based on importance-weighted cross validation (IWCV) (Sugiyama et al., 2007), that we used without modification.

Our experiments' hyperparameters can be grouped into two categories. First, some hyperparameters (e.g., number of epochs) apply to most/all learners (excluding PUc). The second category's hyperparameters are individualized to each learner and were used for all of that learner's experiments on the corresponding dataset.

Table 7 enumerates the general hyperparameter settings that were applied to most/all learners. Batch sizes were selected based on the dataset sizes (see Tables 5 and 6) while the epoch count was determined after monitoring the typical time required for the best validation loss to stop (meaningfully) changing. A grid search was used to select each dataset's layer count; we specifically searched set $\{2, 3\}$ for g and $\{1, 2\}$ for $\hat{\sigma}$. The selected layer count minimized the median validation loss across all learners.

Tables 8, 9, and 10 enumerate the final hyperparameter settings for our models, nnPU, and the positive-negative (PN) learners respectively. The selected hyperparameter setting had the best average validation loss across 10 independent trials. We also used a grid search for these parameters. The search space was: learning rate $\eta \in \{10^{-5}, 10^{-4}, 10^{-3}\}$, weight decay $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}\}$, and (where applicable) gradient attenuator $\gamma \in \{0.1, 0.5, 1.0\}$.

Table 7. General hyperparameter settings

Dataset	#Epoch	Layer Count		Batch Size			
		$g(x)$	$\hat{\sigma}(x)$	$g(x)$	$\hat{\sigma}(x)$	PN$_{\text{tr}}$	PN$_{\text{te}}$
Synthetic	100	N/A	N/A	2,000	750	500	500
MNIST	200	3	1	5,000	5,000	4,000	4,000
20 Newsgroups	200	2	1	5,000	2,500	2,000	2,000
CIFAR10	200	2	1	5,000	2,500	1,500	1,500
banana	500	3	2	500	750	500	500
cod-rna	500	2	1	500	750	500	500
susy	500	2	2	500	750	500	500
ijcnn1	500	2	2	500	750	500	500
covtype.b	500	3	1	500	750	500	500
phishing	500	2	2	500	750	500	500
a9a	500	2	2	500	750	500	500
connect4	500	2	1	500	750	500	500
w8a	500	2	1	500	750	500	500
epsilon	500	2	1	500	750	500	500

Table 8. Dataset-specific hyperparameter settings for our learners

Dataset	PURR			$\hat{\sigma}$			aPNU			wUU		
	η	λ	γ	η	λ	γ	η	λ	γ	η	λ	γ
Synthetic	1E−2	0	1.0	1E−2	0	1.0	1E−2	0	1.0	1E−2	0	1.0
MNIST	1E−3	1E−3	1.0	1E−3	5E−3	1.0	1E−3	1E−3	1.0	1E−3	1E−2	0.1
20 Newsgroups	1E−4	1E−3	0.5	1E−3	5E−3	1.0	1E−4	1E−4	0.5	1E−4	1E−4	0.5
CIFAR10	1E−3	1E−3	1.0	1E−3	5E−3	1.0	1E−3	1E−4	0.5	1E−3	1E−2	0.5
banana	1E−4	1E−3	0.1	1E−4	5E−3	1.0	1E−5	1E−3	0.5	1E−3	1E−3	0.1
cod_rna	1E−4	1E−3	0.5	1E−3	1E−4	1.0	1E−3	1E−3	0.1	1E−4	1E−3	0.5
susy	1E−5	1E−2	0.5	1E−3	1E−4	1.0	1E−5	1E−3	0.1	1E−5	1E−4	0.5
ijcnn1	1E−4	1E−3	0.5	1E−4	5E−3	1.0	1E−5	1E−4	0.5	1E−3	1E−3	0.5
covtype.b	1E−5	1E−3	1.0	1E−3	1E−4	1.0	1E−5	1E−3	0.1	1E−4	1E−3	1.0
phishing	1E−4	1E−3	0.5	1E−3	1E−4	1.0	1E−5	1E−3	0.5	1E−3	1E−4	0.5
a9a	1E−5	1E−4	1.0	1E−4	5E−3	1.0	1E−5	1E−4	0.5	1E−4	1E−3	0.5
connect4	1E−4	1E−3	0.5	1E−3	1E−4	1.0	1E−4	1E−4	0.5	1E−3	1E−2	0.5
w8a	1E−5	1E−4	0.5	1E−3	1E−4	1.0	1E−5	1E−3	0.5	1E−5	1E−2	0.5
epsilon	1E−3	1E−3	0.1	1E−3	1E−4	1.0	1E−3	1E−3	0.1	1E−3	1E−2	0.1

Table 9. Dataset-specific hyperparameter settings for nnPU

Dataset	nnPU$_{\text{te}\cup\text{ tr}}$			nnPU$_{\text{te}}$		
	η	λ	γ	η	λ	γ
Synthetic	1E−2	0	1.0	1E−2	0	1.0
MNIST	1E−3	1E−3	0.5	1E−3	1E−3	0.5
20 Newsgroups	1E−3	1E−3	0.5	1E−3	1E−2	0.5
CIFAR10	1E−4	1E−3	0.1	1E−4	1E−3	0.1
banana	1E−3	1E−3	1.0	1E−4	1E−3	0.5
cod_rna	1E−3	1E−3	0.5	1E−3	1E−3	0.5
susy	1E−5	1E−2	0.1	1E−3	1E−3	0.5
ijcnn1	1E−3	1E−2	0.5	1E−3	1E−3	0.5
covtype.b	1E−3	1E−2	0.5	1E−3	1E−2	0.5
phishing	1E−3	1E−2	0.5	1E−3	1E−2	0.5
a9a	1E−3	1E−2	1.0	1E−3	1E−3	0.5
connect4	1E−3	1E−3	0.1	1E−3	1E−4	1.0
w8a	1E−3	1E−3	0.5	1E−3	1E−3	0.5
epsilon	1E−3	1E−3	0.5	1E−3	1E−3	0.5

Table 10. Dataset-specific hyperparameter settings for the positive-negative (PN) learners

Dataset	PN$_{\text{te}}$		PN$_{\text{tr}}$	
	η	λ	η	λ
Synthetic	1E−2	0	1E−2	0
MNIST	1E−3	1E−3	1E−3	1E−3
20 Newsgroups	1E−3	1E−3	1E−3	1E−2
CIFAR10	1E−4	1E−3	1E−3	1E−2
banana	1E−4	1E−2	1E−4	1E−3
cod_rna	1E−3	1E−4	1E−3	1E−4
susy	1E−4	1E−2	1E−5	1E−2
ijcnn1	1E−3	1E−3	1E−3	1E−2
covtype.b	1E−3	1E−2	1E−3	1E−2
phishing	1E−3	1E−3	1E−3	1E−2
a9a	1E−5	1E−2	1E−3	1E−3
connect4	1E−3	1E−2	1E−3	1E−3
w8a	1E−4	1E−4	1E−4	1E−3
epsilon	1E−4	1E−3	1E−3	1E−3

E. Additional Experimental Results

This section includes experiments we consider insightful but for which there was insufficient space to include in the paper's main body.

E.1. Expanded MNIST, 20 Newsgroups, and CIFAR10 Experiment Set

Table 11 is an expanded version of Section 6.2's Table 1. We provide these additional results to give the reader further evidence of our methods' superior performance.

In this section, each of the three datasets (i.e., MNIST, 20 Newsgroups, and CIFAR10) now has two positive-training class configurations (P_{train}) that are partially disjoint from the positive test class (P_{test}). For each such configuration, Table 11 contains three experiments (in order):

1. $\pi_{\text{tr}} < \pi_{\text{te}}$
2. $\pi_{\text{tr}} = \pi_{\text{te}}$
3. $\pi_{\text{tr}} > \pi_{\text{te}}$

It is easier to directly compare the effects of increasing/decreasing π_{tr} when the magnitude of the prior increase/decrease are equivalent (e.g., for MNIST $\pi_{\text{te}} = 0.5$ so we tested performance at $\pi_{\text{tr}} = \pi_{\text{te}} \pm 0.1$). We maintained that rule of thumb when possible, but cases did arise where there were insufficient positive example with the labels in P_{train} to support such a high positive prior. In those cases, we clamp that P_{train} class definition's maximum π_{tr}.

The key takeaway from Table 11 is that across these additional (orthogonal) definitions of P_{train}, our methods still outperform PUc — usually by a wide margin.

In all experiments, our methods' performance degraded as π_{tr} increased since a larger prior makes it harder to identify the negative examples in $\mathcal{X}_{\text{tr-u}}$. To gain an intuition about why this is true, consider the extreme case where $\pi_{\text{tr}} = 1$; learning is impossible since the positive-train distribution may be arbitrarily different, and there are no negative samples that can be used to relate the two distributions. In contrast when $\pi_{\text{tr}} = 0$, identifying the negative set is trivial (i.e., all of $\mathcal{X}_{\text{tr-u}}$ is negative), and standard nnNU learning can be applied directly to learn g.

PUc performs best when π_{tr} equals π_{te}. When π_{tr} diverges from that middle point, performance declines. To gain an intuition why that is, consider density-ratio estimation in terms of the component class conditionals. When $\pi_{\text{tr}} = \pi_{\text{te}}$, $w(x) = 1$ for all negative examples; from Table 11's results, we know that PUc performs best when there is no bias, i.e., P_{train} = P_{test}. A static positive prior eliminates one possible source of bias making density-ratio estimation easier and more accurate.

E.2. Complete Results for the "Step #2: Classify $\mathcal{X}_{\text{te-u}}$" Experiments

Limited space required trimming some columns from Section 6.3's Table 2. We provide the complete results in Table 12. In summary, PU2wUU lagged PURR and/or PU2aPNU on all experiments, but outperformed PUc on 7/10 of experiments.

Table 11. Full MNIST, 20 Newsgroups, and CIFAR10 experimental class partition results. Each result is the inductive misclassification rate mean & standard deviation over 100 trials for either MNIST, 20 Newsgroups, and CIFAR10 with different positive & negative class definitions. Underlining denotes an improvement versus PUc according to the 5% paired t-test. Boldface indicates a shifted task's best performing method. Negative (N) & positive-test (P_{test}) class definitions are identical for each dataset's first three experiments. Positive train (P_{train}) specified as P_{test} denotes no bias.

	N	P_{test}	P_{train}	π_{tr}	π_{te}		Two-Step (PU2)		Baseline	Reference	
						PURR	aPNU	wUU	PUc	nnPU*	PN_{te}
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	P_{test}	0.5	0.5	9.7 (1.5)	10.6 (1.2)	13.7 (1.8)	8.6 (0.7)	5.5 (0.6)	↑
			7, 9	0.29	0.5	6.5 (0.8)	**5.0 (0.5)**	6.3 (0.8)	29.2 (2.0)	37.7 (4.0)	
				0.5	0.5	9.5 (1.4)	**7.3 (0.8)**	10.2 (1.6)	26.6 (2.4)	36.9 (3.6)	
				0.71	0.5	14.2 (2.5)	**12.7 (1.5)**	18.1 (3.0)	26.7 (3.0)	35.2 (3.0)	2.7 (0.2)
			1, 3, 5	0.38	0.5	7.9 (1.0)	**6.7 (0.7)**	8.8 (1.1)	20.6 (2.3)	25.4 (1.0)	
				0.5	0.5	9.5 (1.3)	**8.8 (0.9)**	12.0 (1.6)	19.1 (2.9)	26.4 (1.1)	
				0.63	0.5	12.2 (2.0)	**11.9 (1.2)**	16.4 (2.2)	26.7 (3.0)	35.2 (3.0)	↓
	0, 2	1, 3	5, 7	0.5	0.5	3.7 (0.6)	**2.7 (0.5)**	3.3 (0.7)	17.4 (4.8)	34.3 (2.6)	1.2 (0.2)
20 Newsgroups	sci, soc, talk	alt, comp, misc, rec	P_{test}	0.56	0.56	15.9 (1.3)	14.7 (1.0)	16.9 (1.7)	14.9 (0.9)	14.6 (1.8)	↑
			misc, rec	0.37	0.56	15.1 (1.2)	**12.3 (0.6)**	14.0 (1.0)	28.6 (1.9)	28.4 (1.0)	
				0.56	0.56	18.5 (1.9)	**13.9 (1.0)**	17.3 (2.0)	24.1 (2.4)	29.1 (1.2)	
				0.65	0.56	21.4 (1.2)	**15.1 (1.3)**	21.1 (3.2)	22.0 (3.1)	29.3 (1.5)	10.4 (0.5)
			comp	0.37	0.56	14.2 (1.0)	**13.1 (0.6)**	14.1 (1.0)	30.3 (2.0)	31.7 (0.7)	
				0.56	0.56	17.0 (1.8)	**14.2 (0.7)**	16.7 (1.5)	28.5 (2.5)	31.8 (0.7)	
				0.65	0.56	20.4 (2.8)	**15.1 (1.3)**	19.5 (2.5)	27.4 (3.1)	31.7 (0.7)	↓
	misc, rec	soc, talk	alt, comp	0.55	0.46	5.8 (1.2)	7.0 (1.4)	**5.1 (1.8)**	18.3 (4.5)	38.4 (3.4)	1.8 (0.5)
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	P_{test}	0.4	0.4	14.0 (1.0)	14.0 (1.0)	16.0 (1.5)	13.8 (0.7)	11.9 (0.7)	↑
			Plane	0.14	0.4	11.6 (0.8)	**11.3 (0.7)**	11.6 (0.8)	26.6 (1.3)	27.5 (1.0)	
				0.4	0.4	**13.6 (0.9)**	13.8 (1.1)	15.3 (1.9)	20.4 (1.4)	27.4 (1.1)	
				0.6	0.4	**16.2 (1.1)**	17.3 (1.6)	20.7 (2.8)	21.7 (1.7)	28.2 (1.1)	9.2 (0.5)
			Auto, Truck	0.25	0.4	12.6 (0.9)	**12.0 (0.8)**	12.7 (1.0)	19.2 (1.0)	20.5 (0.8)	
				0.4	0.4	14.2 (1.1)	**13.6 (1.0)**	14.6 (1.3)	17.8 (1.0)	20.3 (0.8)	
				0.55	0.4	16.5 (1.4)	**16.1 (1.2)**	18.6 (2.5)	18.3 (1.1)	20.5 (0.9)	↓
	Deer, Horse	Plane, Auto	Cat, Dog	0.5	0.5	13.3 (0.9)	14.4 (1.5)	**11.4 (1.4)**	33.2 (2.5)	52.6 (2.1)	7.1 (0.4)

Table 12. Complete results for Section 6.3 entitled "Step #2: Classify $\mathcal{X}_{\text{te-u}}$". The evaluation metric is the inductive misclassification rate mean & standard deviation over 100 trials with Sakai & Shimizu (2019)'s median feature vector-based bias for 10 LIBSVM datasets. Underlining denotes a performance improvement versus PUc according to the 5% paired t-test. Boldface indicates each dataset's best performing method.

Dataset	d		Two-Step (PU2)		Baseline	Reference	
		PURR	aPNU	wUU	PUc	nnPU*	PN_{te}
banana	2	13.3 (2.4)	**12.2 (1.6)**	15.2 (2.6)	17.5 (3.9)	29.8 (4.1)	8.7 (0.7)
cod-rna	8	12.9 (2.9)	**12.6 (4.2)**	14.1 (4.8)	26.1 (5.2)	28.6 (3.9)	6.6 (1.0)
susy	18	**23.7 (1.8)**	27.2 (2.6)	27.9 (3.0)	27.2 (4.0)	46.9 (3.8)	19.8 (1.1)
ijcnn1	22	**21.5 (2.6)**	28.6 (3.6)	26.1 (3.1)	23.4 (3.5)	31.3 (3.4)	6.2 (0.7)
covtype.binary	54	**28.9 (2.7)**	31.7 (3.3)	30.2 (2.8)	39.1 (3.6)	54.4 (3.2)	21.8 (1.3)
phishing	68	11.6 (2.1)	**9.3 (1.0)**	13.1 (2.8)	13.3 (3.5)	22.9 (3.8)	6.0 (0.7)
a9a	123	26.1 (1.8)	**25.4 (1.7)**	27.4 (2.6)	32.9 (2.5)	33.4 (1.9)	20.6 (1.0)
connect4	126	**33.0 (3.1)**	33.2 (2.8)	35.8 (2.6)	37.0 (2.9)	45.7 (3.0)	20.8 (1.1)
w8a	300	**15.2 (1.9)**	19.8 (3.0)	17.9 (2.9)	31.0 (7.5)	41.3 (4.3)	6.4 (0.7)
epsilon	2,000	**32.1 (3.0)**	38.7 (7.5)	33.2 (4.1)	63.5 (6.5)	64.5 (1.5)	23.6 (1.0)

E.3. Analyzing the Effect of Positive and Negative Train Distribution Shift

The goal of these experiments is to:

1. Demonstrate the effectiveness of our approaches across the entire spectrum of positive train distribution shift.
2. Study how our methods perform when the assumption of a fixed negative distribution is violated.

We look at these trends across three datasets (as in Section 6.2): MNIST, 20 Newsgroups, and CIFAR10. The positive and negatives classes are formed by combining two labels from the original dataset (the use of two labels per class is necessary for this experimental setup). Table 13 enumerates each dataset's positive and negative class definitions; these definitions apply for both train and test. The dataset sizes are listed in Table 14; note that n_{Test} is the size of the inductive test set used to measure performance. The validation set was one-fifth the training set size. The priors were also fixed such that $\pi_{\text{tr}} = \pi_{\text{te}} = 0.5$.

Table 13. Positive and negative class definitions for the class-conditional bias experiments

Dataset	Positive		Negative	
	C_1	C_2	C_1	C_2
MNIST	8	9	3	4
20 Newsgroups	sci	rec	comp	talk
CIFAR10	Auto	Plane	Ship	Truck

Table 14. Dataset sizes for the class-conditional bias experiments

Dataset	n_{p}	$n_{\text{tr-u}}$	$n_{\text{te-u}}$	n_{Test}
MNIST	250	5,000	5,000	1,500
20 Newsgroups	500	2,000	2,000	1,000
CIFAR10	500	5,000	5,000	1,500

The default rule in this section is that the positive/negative train/test classes are selected uniformly at random without replacement from their respective subclasses. In each experiment, either the positive-train or negative-train distribution is shifted (never both). The test distribution is never biased and is identical for all experiments.

Positive-Train Shift In these experiments, the positive-train distribution (i.e., $p_{\text{tr-p}}(x)$) is shifted. Recall that each positive class is composed of two labels; denote them C_1 and C_2 (e.g., $C_1 = \text{Auto}$ and $C_2 = \text{Plane}$ for CIFAR10). $\Pr[\text{Label}_{\text{tr}}{=}C_1 | Y = +1]$ is the probability that any positive-valued *training* example has original label C_1. Since there are two labels per class,

$$\Pr[\text{Label}_{\text{tr}}{=}C_2 | Y = +1] = 1 - \Pr[\text{Label}_{\text{tr}}{=}C_1 | Y = +1]. \tag{17}$$

The positive-train distribution shift entails sweeping $\Pr[\text{Label}_{\text{tr}}{=}C_1 | Y = +1]$ from 0.5 to 1 (i.e., from unbiased on the left to maximally biased on the right). This setup is more challenging than shifting the positive-test distribution since it entails the learner seeing fewer *labeled* examples from positive subclass C_2.

Figures 4a, 4c, and 4e show the positive-train shift's effect on the MNIST, 20 Newsgroups, and CIFAR10 misclassification rate respectively (where C_1 corresponds to digit 8, document category "rec", and image type "automobile"). PURR's performance was consistent across the entire bias range while the two step methods' (PU2wUU and PU2aPNU) performance improved as bias increased (due to easier identification of negative examples as explained in Section 6.2). In contrast, PUc's performance degrades as bias increases; this degradation is largely due to poor density estimation and demonstrates why covariate shift methods can be non-ideal.

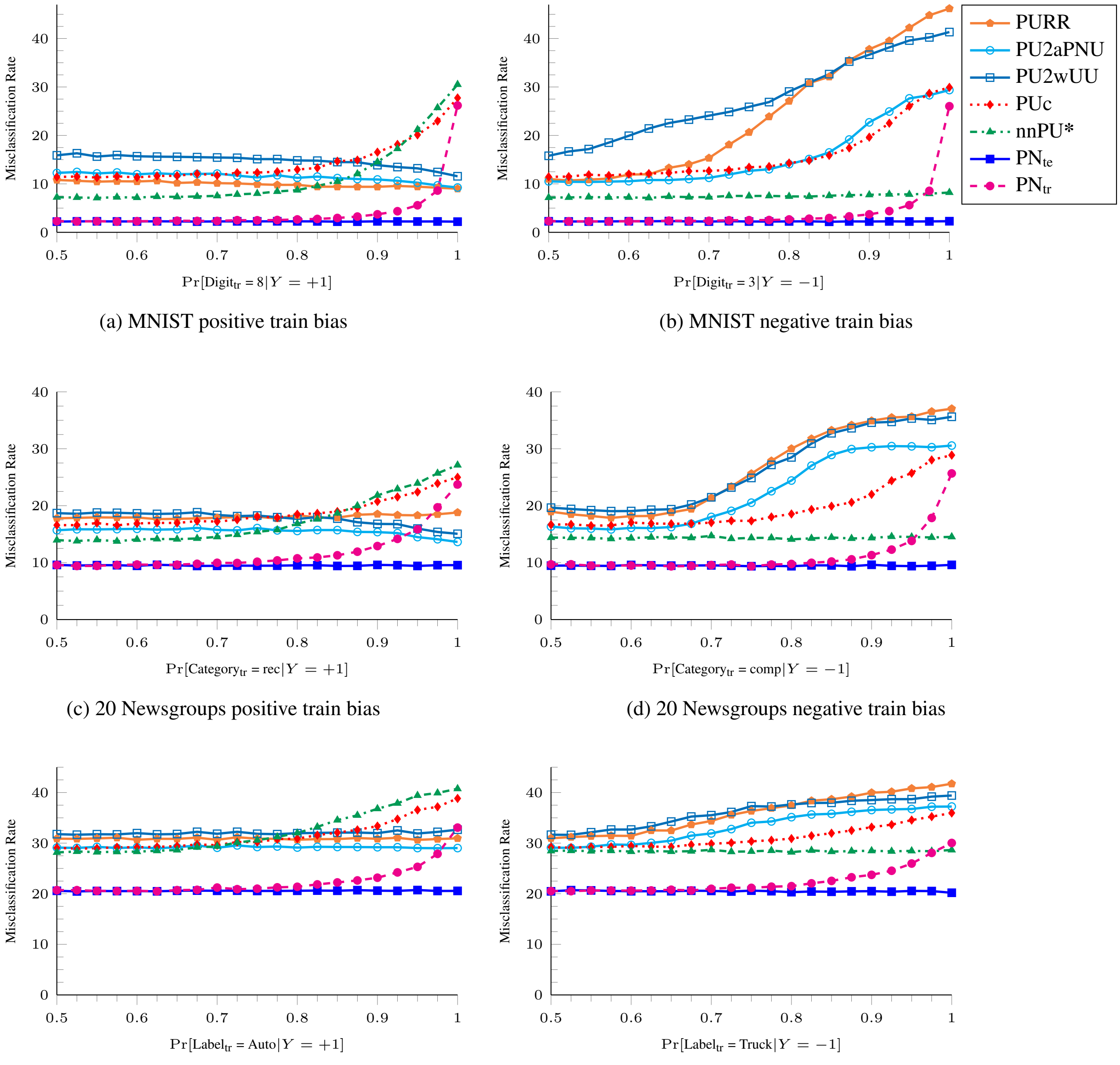
PN$_{\text{tr}}$ and PN$_{\text{te}}$ are trained using (labeled) $\mathcal{X}_{\text{tr-u}}$ and $\mathcal{X}_{\text{te-u}}$. Since the test distributions are never biased, PN$_{\text{te}}$ is unaffected by shift. In contrast, as $\Pr[\text{Label}_{\text{tr}}{=}C_1 | Y = +1]$ increases, there are fewer examples in $\mathcal{X}_{\text{tr-u}}$ with label C_2 causing a degradation in PN$_{\text{tr}}$'s performance.

PUc's and nnPU*'s performance begins to degrade at the same point where PN_{tr}'s and PN_{te}'s performance begins to diverge. For nnPU* in particular, this degradation is primarily attributable to fewer examples labeled C_2 in $\mathcal{X}_{\text{p}}$. PUc is more robust to bias than nnPU* (as shown by the slower rate of degradation) since it considers distributional shifts.

Negative-Train Shift These experiments follow the same basic concept as the positive-train distribution shift described above except that the bias is instead applied to the negative-train distribution, i.e., $p_{\text{tr-n}}(x)$. This bias means that $p_{\text{tr-n}}(x) \neq p_{\text{te-n}}(x)$. To reiterate, *these experiments deliberately violate Eq.* (5)*'s assumption* upon which our methods are predicated. The goal here is to understand our methods' robustness under intentionally deleterious conditions. It is more deleterious to bias the negative class in $\mathcal{X}_{\text{tr-u}}$ since both two-step methods and PURR use $\mathcal{X}_{\text{tr-u}}$'s negative risk in dependent calculations; any error propagates and compounds in these subsequent operations.

Let C_1 and C_2 now be the two labels that make up the negative class (e.g., $C_1 =$ Ship and $C_2 =$ Truck for CIFAR10). $\Pr[\text{Label}_{\text{tr}} = C_1 | Y = +1]$ is again swept along the x-axis from 0.5 to 1 (unbiased to maximally biased). The results for MNIST, 20 Newsgroups, and CIFAR10 are in Figures 4b, 4d, and 4f respectively.

With the exception of PU2wUU on MNIST, all of our methods showed moderate robustness to some negative distribution bias. In particular, PU2aPNU was almost as robust as PUc in some cases. nnPU*'s robustness here is expected since anything not in $\mathcal{X}_{\text{p}}$ is assumed negative; even under bias, sufficient negative examples exist for each label in $\mathcal{X}_{\text{te-u}}$ to allow nnPU* to learn how to classify those examples.



(a) MNIST positive train bias

(b) MNIST negative train bias

(c) 20 Newsgroups positive train bias

(d) 20 Newsgroups negative train bias

(e) CIFAR10 positive train bias

(f) CIFAR10 negative train bias

Figure 4. Effect of positive ($p_{\text{tr-p}}(x)$) or negative ($p_{\text{tr-n}}(x)$) train distribution shift on inductive misclassification rate for the MNIST, 20 Newsgroups, and CIFAR10 datasets. The x-axis corresponds to $\Pr[\text{Label}_{\text{tr}} = C_1|y = \hat{y}]$ where $\hat{y} \in \{\pm1\}$. Each data point is the average of 100 trials.

E.4. Effect of Prior Probability Estimation Error

As explained in Section 3, this work assumes that positive-class priors, π_{tr} and π_{te}, are known. The goal of these experiments is to study our methods' performance when the priors are misspecified.

Experimental Setup These experiments reuse the partially disjoint positive-support experiment setups from Section 6.2's Table 1. Therefore, we are specifically considering the MNIST, 20 Newsgroups, and CIFAR10 datasets with Table 15 summarizing all setups.

π_{tr} and π_{te} in Table 15 are the *actual* prior probabilities used to construct each training and test data set. We tested our methods' performance when each prior was specified correctly and when each prior was misspecified by $\pm 20\%$ for a total of $9 = 3 \times 3$ conditions per learner. π_{tr} was only misspecified when training g. $\hat{\sigma}$ was always provided the correct prior; this decision was made due to constraints in the implementation of our code. It is not an algorithmic limitation. Like all previous experiments, performance was evaluated using the inductive misclassification rate.

PUc estimates π_{te} as part of its density-ratio estimation. As such, we only report three bias conditions for PUc, all over training prior π_{tr}.

Tables 16, 17, and 18 contain the results for MNIST, 20 Newsgroups, and CIFAR10 respectively. Each learner's results are presented in a 3×3 grid with π_{tr} changing row to row while π_{te} changes column to column. Each cell is shaded red, with a darker background denoting worse performance (i.e., a greater misclassification rate). Even under worst case bias where both π_{tr} and π_{te} were shifted, all of our methods outperformed PUc.

Similar to Section E.3's experiments, the MNIST results were most affected by bias. The 20 Newsgroups and CIFAR10 results were more immune due to the richer feature representations generated through transfer learning. In most cases, the worst performance was observed when π_{tr} and π_{te} saw opposite bias, e.g., π_{tr} was overestimated while π_{te} was underestimated or vice versa; these values appear in the upper-right or lower-left corners of each learner's 3×3 grid.

aPNU was least affected by misspecified priors. While this is partially due to $\hat{\sigma}$ not observing the misspecified prior, it is not entirely due to that since aPNU generally shifted less than wUU.

Table 15. Positive train (P_{train}), positive test (P_{test}), and negative (N) class definitions and actual prior probabilities for the experiments examining the effect of misspecified prior(s) on our algorithms' performance.

	N	P_{train}	P_{test}	π_{tr}	π_{te}
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	7, 9	0.5	0.5
20 News.	sci, soc, talk	alt, comp, misc, rec	misc, rec	0.37	0.56
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	Plane	0.4	0.4

Table 16. Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on MNIST's inductive misclassification rate. Each result is the average of 100 trials.

	PURR			aPNU			wUU			PUc
	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	
$0.8\pi_{\text{tr}}$	17.4	16.6	19.8	7.4	9.5	12.2	10.3	13.2	18.7	29.6
π_{tr}	12.9	9.2	13.6	6.6	7.4	10.1	8.5	10.3	13.9	26.7
$1.2\pi_{\text{tr}}$	25.3	15.8	12.7	18.0	7.7	7.5	16.9	8.9	10.3	26.3

Table 17. Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on 20 Newsgroups's inductive misclassification rate. Each result is the average of 100 trials.

	PURR			aPNU			wUU			PUc
	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	
$0.8\pi_{\text{tr}}$	19.9	18.8	18.3	12.5	12.5	13.2	13.9	14.6	16.6	34.2
π_{tr}	16.8	15.3	17.7	12.7	12.3	12.8	14.0	14.2	15.7	28.8
$1.2\pi_{\text{tr}}$	16.4	13.8	16.7	14.8	12.4	12.5	16.4	13.9	15.0	25.3

Table 18. Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on CIFAR10's inductive misclassification rate. Each result is the average of 100 trials.

	PURR			aPNU			wUU			PUc
	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	$0.8\pi_{\text{te}}$	π_{te}	$1.2\pi_{\text{te}}$	
$0.8\pi_{\text{tr}}$	16.5	16.5	18.9	14.0	15.4	16.9	15.5	17.5	20.6	23.9
π_{tr}	15.0	13.7	15.9	13.3	14.1	15.7	14.1	15.5	17.8	20.6
$1.2\pi_{\text{tr}}$	18.1	14.8	14.8	15.7	13.5	14.3	15.7	13.9	15.5	20.0