Learning from Positive and Unlabeled Data with Arbitrary Positive Shift

Zayd Hammoudeh   **Daniel Lowd**
Department of Computer & Information Science
University of Oregon
Eugene, OR, USA
`{zayd, lowd}@cs.uoregon.edu`

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 rarely holds in practice due to temporal drift, domain shift, and/or adversarial manipulation. This paper shows that PU learning is possible even with arbitrarily non-representative positive data given unlabeled data from the source and target distributions. Our key insight is that only the negative class's distribution need be fixed. We integrate this into two statistically consistent methods to address arbitrary positive bias – one approach combines *negative-unlabeled learning* with *unlabeled-unlabeled learning* while the other uses a novel, recursive risk estimator. Experimental results demonstrate our methods' effectiveness across numerous real-world datasets and forms of positive bias, including disjoint positive class-conditional supports. Additionally, we propose a general, simplified approach to address PU risk estimation overfitting.

1   Introduction

Positive-negative (PN) *learning* (i.e., ordinary supervised classification) trains a binary classifier using positive and negative labeled datasets. In practice, good labeled data are often unavailable for one class. High negative-class diversity may make constructing a representative labeled set prohibitively difficult [1], or negative data may not be systematically recorded in some domains [2].

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: opinion spam detection [3], disease-gene identification [4], land-cover classification [5], and protein similarity prediction [6]. The related task of *negative-unlabeled* (NU) learning is functionally identical to PU learning but with labeled data drawn from the negative class.

Most PU learning methods assume the labeled set is *selected completely at random* (SCAR) from the target distribution [1, 6, 7, 8, 9, 10, 11]. External factors like temporal drift, domain shift, and adversarial concept drift often cause the labeled-positive and target distributions to diverge.

Biased-positive, unlabeled (bPU) *learning* algorithms relax SCAR by modeling *sample selection bias* for the labeled data [12, 13] or a *covariate shift* between the training and target distributions [14].

This paper generalizes bPU learning to the more challenging *arbitrary-positive, unlabeled* (aPU) learning setting, where the labeled (positive) data may be *arbitrarily different* from the target distribution's positive class. Solving this problem would eliminate the need to spend time and money labeling new data whenever the positive class drifts.

Devoid of some assumption, aPU learning is impossible [6]. As a first step to address aPU learning, our key insight is that given a labeled-positive set and two unlabeled sets as proposed by Sakai and Shimizu [14], aPU learning is possible when *all negative examples are generated from a single distribution*. The labeled and target-positive distributions' *supports* (sets of examples with non-zero probability) may even be disjoint. Many real-world PU learning tasks feature a shifting positive class but (largely) fixed negative class including:

1. **Land-Cover Classification**: Cross-border land-cover datasets often do not exist due to differing national technological standards or insufficient financial resources by one country [15]. This limits research into natural processes at broad geographic scales. However, cross-border geographic terrains often follow a similar distribution differing primarily in man-made objects (e.g., roads) due to local construction materials and regulations [5].
2. **Adversarial aPU Learning**: Malicious adversaries (email spammers, malware authors) rapidly adapt their attacks to bypass automated detection. The benign class changes much more slowly but may be too diverse to construct a representative labeled set [3, 16, 17, 18].

Our paper's four primary contributions are enumerated below. Note that most experiments and all proofs are in the supplemental materials.

1. We propose abs-PU – a simplified, statistically *consistent* approach to correct general PU risk estimation overfitting. Our aPU methods leverage abs-PU to streamline their optimization.
2. 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.
3. We separately propose PURR – a novel, recursive, consistent aPU risk estimator.
4. We evaluate our methods on a wide range of benchmarks, demonstrating our algorithms' effectiveness over the state of the art in PU and bPU learning. Our empirical evaluation includes an adversarial aPU learning case study using public spam email datasets.

2 Ordinary Positive-Unlabeled Learning

We begin with an overview of PU learning without distributional shifts, including definitions and notation. Consider two random variables, covariate $X \in \mathbb{R}^d$ and label $Y \in \{\pm 1\}$, with joint distribution $p(x, y)$. Marginal distribution $p_u(x)$ is composed from the positive prior $\pi := p(Y = +1)$, positive class-conditional $p_p(x) := p(x|Y = +1)$, and negative class-conditional $p_n(x) := p(x|Y = -1)$.

Risk 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 $R(g) := \mathbb{E}_{(X,Y) \sim p(x,y)}[\ell(Yg(X))]$ quantifies g's expected loss over $p(x, y)$. It decomposes via the product rule to $R(g) = \pi R_p^+(g) + (1 - \pi)R_n^-(g)$, where the *labeled risk* is

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

for predicted label $\hat{y} \in \{\pm 1\}$ and $\mathcal{D} \in \{p, n, u\}$ denoting the positive class-conditional, negative class-conditional or marginal distribution respectively, as defined above.

Since $p(x, y)$ is unknown, *empirical risk* is used in practice. We consider the *case-control scenario* [19] where each dataset is i.i.d. sampled from its associated distribution. PN learning has two labeled datasets: positive set $\mathcal{X}_p := \{x_i^p\}_{i=1}^{n_p} \overset{\text{i.i.d.}}{\sim} p_p(x)$ and negative set $\mathcal{X}_n := \{x_i^n\}_{i=1}^{n_n} \overset{\text{i.i.d.}}{\sim} p_n(x)$. These are used to calculate empirical labeled risks $\widehat{R}_p^+(g) = \frac{1}{n_p} \sum_{i=1}^{n_p} \ell(g(x_i^p))$ and $\widehat{R}_n^-(g) = \frac{1}{n_n} \sum_{i=1}^{n_n} \ell(-g(x_i^n))$. We denote the empirical positive-negative risk

$$\widehat{R}_{\text{PN}}(g) := \pi \widehat{R}_p^+(g) + (1 - \pi)\widehat{R}_n^-(g). \tag{2}$$

PU learning cannot directly estimate $R_n^{\hat{y}}(g)$ since there is no negative (labeled) data (i.e., $\mathcal{X}_n = \emptyset$). Let $\mathcal{X}_u := \{x_i^u\}_{i=1}^{n_u} \overset{\text{i.i.d.}}{\sim} p_u(x)$ be an unlabeled set with empirical labeled risk $\widehat{R}_u^{\hat{y}}(g) = \frac{1}{n_u} \sum_{i=1}^{n_u} \ell(\hat{y}g(x_i^u))$. du Plessis et al. [20] make a foundational contribution that,

$$(1 - \pi)R_n^{\hat{y}}(g) = R_u^{\hat{y}}(g) - \pi R_p^{\hat{y}}(g). \tag{3}$$

Their unbiased PU (uPU) risk estimator is therefore $\widehat{R}_{\text{uPU}}(g) := \pi \widehat{R}_p^+(g) + \widehat{R}_u^-(g) - \pi \widehat{R}_p^-(g)$. Kiryo et al. [8] observe that highly expressive models (e.g., neural networks) often overfit $\mathcal{X}_p$ causing uPU to estimate that $\widehat{R}_u^-(g) - \pi \widehat{R}_p^-(g) < 0$.

Since negative-valued risk is impossible, Kiryo et al.'s non-negative PU (nnPU) risk estimator ignores negative estimates of risk via a $\max$ term:

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

When Kiryo et al.'s customized empirical risk minimization (ERM) framework detects overfitting (i.e., $\widehat{R}_{\text{u}}^{-}(g) - \pi \widehat{R}_{\text{p}}^{-}(g) < 0$), their framework "defits" g using negated gradient $-\gamma \nabla_{\theta}(\widehat{R}_{\text{u}}^{-}(g) - \pi \widehat{R}_{\text{p}}^{-}(g))$, where hyperparameter $\gamma \in (0, 1]$ attenuates the learning rate to throttle "defitting." Observe that positive-labeled risk, $\widehat{R}_{\text{p}}^{+}(g)$, is excluded from nnPU's negated gradient.

3   Simplifying Non-Negativity Correction

Rather than enforcing the non-negative risk constraint with two combined techniques (a $\max$ term and "defitting") like Kiryo et al., we propose a simpler approach, inspired by Lagrange multipliers, that directly puts the non-negativity constraint into the risk estimator. Our *absolute-value correction*,

$$(1 - \pi)\ddot{R}_{\text{n}}^{\hat{y}}(g) \coloneqq \left|\widehat{R}_{\text{u}}^{\hat{y}}(g) - \pi \widehat{R}_{\text{p}}^{\hat{y}}(g)\right|, \tag{5}$$

replaces nnPU's $\max$ with absolute value to prevent the optimizer overfitting an implausible risk estimate by explicitly penalizing those risk estimates for being negative. This penalty "defits" the learner automatically, eliminating the need for hyperparameter γ and nnPU's custom ERM algorithm.

Theorem 1. *Let $g : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary decision function and let $\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be a loss function bounded[1] w.r.t. g then $\ddot{R}_{\text{n}}^{\hat{y}}(g)$ is a consistent estimator of $\widehat{R}_{\text{n}}^{\hat{y}}(g)$.*

We integrate absolute value correction into our *abs-PU risk estimator*,

$$\widehat{R}_{\text{abs-PU}}(g) \coloneqq \pi \widehat{R}_{\text{p}}^{+}(g) + \left|\widehat{R}_{\text{u}}^{-}(g) - \pi \widehat{R}_{\text{p}}^{-}(g)\right|, \tag{6}$$

which by Theorem 1 is consistent like nnPU. When $\widehat{R}_{\text{u}}^{-}(g) - \pi \widehat{R}_{\text{p}}^{-}(g) < 0$, abs-PU's update gradient, $\nabla_{\theta}(\pi \widehat{R}_{\text{p}}^{+}(g) - \widehat{R}_{\text{u}}^{-}(g) + \pi \widehat{R}_{\text{p}}^{-}(g))$, includes $\widehat{R}_{\text{p}}^{+}(g)$. Hence, abs-PU spends comparatively more time optimizing the positive-labeled risk than nnPU. Also, by penalizing implausible risk, abs-PU estimates validation performance (i.e., risk) differently than nnPU.

Empirically we observed that abs-PU yields models of similar or slightly better accuracy than nnPU albeit with a simpler, more efficient optimization. The following builds on abs-PU with a full comparison to nnPU in supplemental Section E.6.

4   Arbitrary-Positive, Unlabeled Learning

Arbitrary-positive unlabeled (aPU) learning — the focus of this work — is one of three problem settings proposed by Sakai and Shimizu [14]. 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.

No assumption is made about the label's conditional probability, i.e., $p_{\text{tr}}(y|x)$ and $p_{\text{te}}(y|x)$, nor about positive class-conditionals $p_{\text{tr-p}}(x)$ and $p_{\text{te-p}}(x)$. We only assume a fixed negative class-conditional

$$p_{\text{n}}(x) = p_{\text{tr-n}}(x) = p_{\text{te-n}}(x). \tag{7}$$

Both the train and test positive-class priors, π_{tr} and π_{te} respectively, are treated as known throughout this work. In practice, they may be known *a priori* through domain-specific knowledge. Techniques also exist to estimate them from data [2, 21, 22, 23]. Theorem 4 in the supplemental materials provides an algorithm to estimate π_{te} by training an additional classifier.

As shown in Figure 1a, the available datasets are: labeled (positive) set $\mathcal{X}_{\text{p}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-p}}(x)$ as well as unlabeled sets $\mathcal{X}_{\text{tr-u}} \coloneqq \{x_i\}_{i=1}^{n_{\text{tr-u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)$ and $\mathcal{X}_{\text{te-u}} \coloneqq \{x_i\}_{i=1}^{n_{\text{te-u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{te-u}}(x)$ with their empirical risks defined as before. An optimal classifier minimizes the *test* risk/expected loss: $\mathbb{E}_{(X,Y) \sim p_{\text{te}}(x,y)}[\ell(Y g(X))]$.

---

[1]Each theorem's definition of "bounded" loss appears in the associated proof. See the supplemental materials.

(a) Example aPU dataset

(b) Weighting $\mathcal{X}_{\text{tr-u}}$ using $\hat{\sigma}(x)$ yields $\widetilde{\mathcal{X}}_{\text{n}}$
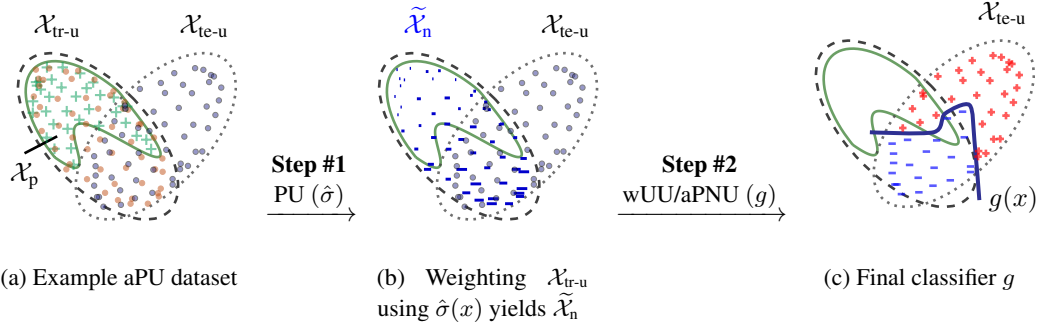
(c) Final classifier g

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.

4.1 Relating aPU Learning and Covariate Shift Adaptation Methods

Covariate shift [24] is a common technique to address differences between $p_{\text{tr}}(x, y)$ and $p_{\text{te}}(x, y)$. Unlike aPU learning, covariate shift restrictively assumes a *consistent input-output relation*, i.e., $p_{\text{tr}}(y|x) = p_{\text{te}}(y|x)$. Define the *importance function* as $w(x) := \frac{p_{\text{te-u}}(x)}{p_{\text{tr-u}}(x)}$. When $p(y|x)$ is fixed, it is easy to show that $w(x)p_{\text{tr}}(x, y) = p_{\text{te}}(x, y)$.

Sakai and Shimizu [14] exploit this relationship in their PUc risk estimator. $w(x)$ is approximated via direct density-ratio estimation [25] – specifically the RuLSIF algorithm [26] over $\mathcal{X}_{\text{tr-u}}$ and $\mathcal{X}_{\text{te-u}}$. Their PUc risk adds importance weighting to uPU, with the labeled risks still estimated from $\mathcal{X}_{\text{p}}$ and $\mathcal{X}_{\text{tr-u}}$. Sakai and Shimizu's formulation specifies linear-in-parameter models to enforce convexity. They improve tractability via a simplified version of du Plessis et al. [1]'s surrogate squared loss for ℓ.

Selection bias bPU methods [12, 13] need the positive-labeled data to meet specific conditions that arbitrary-positive data will not satisfy making a comparison to those methods infeasible. PUc serves as the primary baseline here since as a covariate shift bPU method, it places no requirements on the positive data beyond that the training distribution's support be a superset of the target positive class.

4.2 Comparing Variations of the aPU Learning Problem

Sakai and Shimizu [14] show that PU learning with a fixed positive class and arbitrary negative shift is much simpler than aPU learning. In fact, provided a positive-labeled set and two unlabeled sets as above, they show that arbitrary negative shift is trivially equivalent to ordinary PU learning over $\mathcal{X}_{\text{p}}$ and $\mathcal{X}_{\text{te-u}}$ (since $\mathcal{X}_{\text{p}}$ being drawn from $p_{\text{te-p}}(x)$ renders $\mathcal{X}_{\text{tr-u}}$ unnecessary). When both the positive and negative classes shift arbitrarily, learning is impossible without additional data and/or assumptions. aPU learning's complexity sits between these two extremes.

5 aPU Learning via Unlabeled-Unlabeled 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. (7)'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 g: one a variant of NU learning we call weighted-unlabeled, unlabeled (wUU) learning and the other a semi-supervised method we call arbitrary-positive, negative, unlabeled (aPNU) learning. We refer to the complete algorithms as PU2wUU and PU2aPNU, respectively.

4

Algorithm 1 Two-step unlabeled-unlabeled aPU learning

Input: Labeled-positive set $\mathcal{X}_{\text{p}}$ and unlabeled sets $\mathcal{X}_{\text{tr-u}}, \mathcal{X}_{\text{te-u}}$
Output: g's model parameters θ

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 final classifier, g, using ERM with $\widehat{R}_{\text{wUU}}(g)$ or $\widehat{R}_{\text{aPNU}}(g)$

Figure 1 visualizes our two-step approach, with a formal presentation in Algorithm 1. Below is a detailed description and theoretical analysis.

Step #1: Create Surrogate Negative Set $\widetilde{\mathcal{X}}_{\text{n}}$ from $\mathcal{X}_{\text{tr-u}}$

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 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 2. *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. 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 $\widetilde{R}_{\text{n-u}}^{\hat{y}}(g) := \frac{1}{n_{\text{tr-u}}} \sum_{x_i \in \mathcal{X}_{\text{tr-u}}} \frac{\hat{\sigma}(x_i)\ell(\hat{y}g(x_i))}{1-\pi_{\text{tr}}}$. Let $\hat{\sigma} : \mathbb{R}^d \to [0,1]$ be in hypothesis set $\widehat{\Sigma}$. When $\hat{\sigma}(x) = p_{\text{tr}}(Y = -1|x)$, $\widetilde{R}_{\text{n-u}}^{\hat{y}}(g)$ is an unbiased estimator of $R_{\text{n}}^{\hat{y}}(g)$. 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 $\widetilde{R}_{\text{n-u}}^{\hat{y}}(g)$ is a consistent estimator of $R_{\text{n}}^{\hat{y}}(g)$.*

From Theorem 2, we see that *soft* weighting each unlabeled 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/test negative labeled risk. We form $\widetilde{\mathcal{X}}_{\text{n}}$ transductively, but inductive learning is an option. Since $\mathcal{X}_{\text{tr-u}}$ contains positive examples, $\hat{\sigma}$ may overfit and memorize random positive example variation. This is usually detectable via an implausible validation loss given π_{tr}, n_{p}, and $n_{\text{tr-u}}$. Care should be shown to tune $\hat{\sigma}$'s capacity and regularization.

Supplemental Section E.7 proposes and empirically evaluates two additional methods to construct $\widetilde{\mathcal{X}}_{\text{n}}$. While these other methods are not statistically consistent, they may outperform soft weighting.

What if $\mathcal{X}_{\text{p}}$ is not SCAR? Our aPU learning setting, detailed in Section 4, specifies that $\mathcal{X}_{\text{p}}$ is representative of $\mathcal{X}_{\text{tr-u}}$'s positive examples. In scenarios where $\mathcal{X}_{\text{p}}$ is biased w.r.t. $\mathcal{X}_{\text{tr-u}}$, any bPU method (e.g., [12, 13]) can be used in step #1 to (hard) label $\mathcal{X}_{\text{tr-u}}$ thereby constructing $\widetilde{\mathcal{X}}_{\text{n}}$.

Step #2: Train the Test Distribution Classifier g

Negative-unlabeled (NU) learning is functionally the same as PU learning. Sakai et al. [27] formalize an unbiased NU risk estimator, $\widehat{R}_{\text{NU}}(g) := \left| \widehat{R}_{\text{u}}^+(g) - (1-\pi)\widehat{R}_{\text{n}}^+(g) \right| + (1-\pi)\widehat{R}_{\text{n}}^-(g)$ (defined here with our absolute-value correction). Our *weighted-unlabeled, unlabeled*[2] (wUU) estimator,

$$\widehat{R}_{\text{wUU}}(g) := \left| \widehat{R}_{\text{te-u}}^+(g) - (1-\pi_{\text{te}})\widetilde{R}_{\text{n-u}}^+(g) \right| + (1-\pi_{\text{te}})\widetilde{R}_{\text{n-u}}^-(g), \tag{8}$$

modifies Sakai et al.'s definition to use $\widetilde{\mathcal{X}}_{\text{n}}$ and $\mathcal{X}_{\text{te-u}}$. Observe that $\widehat{R}_{\text{wUU}}(g)$ uses only data that was originally unlabeled. When $\widetilde{R}_{\text{n-u}}^{\hat{y}}(g)$ is consistent, wUU is also consistent just like nnPU/abs-PU.

Risk Estimation with Positive Data Reuse 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 given limited data. In such settings, a semi-supervised approach leveraging $\mathcal{X}_{\text{p}}$, surrogate $\widetilde{\mathcal{X}}_{\text{n}}$, and $\mathcal{X}_{\text{te-u}}$ may perform better than wUU.

Sakai et al. [27] 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. (9) modifies PNU to use $\widetilde{\mathcal{X}}_{\text{n}}$ and our absolute-value correction.

$$\widehat{R}_{\text{aPNU}}(g) = (1-\rho)\pi_{\text{te}}\widehat{R}_{\text{p}}^+(g) + (1-\pi_{\text{te}})\widetilde{R}_{\text{n-u}}^-(g) + \rho\left| \widehat{R}_{\text{te-u}}^+(g) - (1-\pi_{\text{te}})\widetilde{R}_{\text{n-u}}^+(g) \right| \tag{9}$$

---

[2]"Unlabeled-unlabeled learning" denotes the two unlabeled sets and is different from UU learning in [28, 29].

5

If $\rho = 0$, aPNU ignores the test distribution (i.e., $\mathcal{X}_{\text{te-u}}$) entirely. If $\rho = 1$, aPNU is simply wUU. When a large positive shift is expected (e.g., by domain-specific knowledge), $\mathcal{X}_{\text{p}}$ is of limited value so set ρ closer to 1. For small expected positive shifts, set ρ closer to 0. A midpoint value of $\rho = 0.5$ empirically performed well when no knowledge about the positive shift was assumed.

ERM Framework Both $\widehat{R}_{\text{wUU}}(g)$ and $\widehat{R}_{\text{aPNU}}(g)$ integrate into a standard ERM framework since they use our absolute-value correction. For completeness, supplemental materials Section C.1 details their custom ERM algorithm if Kiryo et al. [8]'s non-negativity correction is used instead.

Heterogeneous Classifiers Two-step learners enable different learner architectures in each step (e.g., random forest for step #1 and a neural network for step #2). Our experiments leverage this flexibility where $\hat{\sigma}$'s neural network may have fewer hidden layers or different hyperparameters than g in step #2.

6   Positive-Unlabeled 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 those errors.

Can our aPU problem setting be learned in a single *joint* method? Sakai and Shimizu leave it as an open question. We show in this section the answer is yes. To understand why this is possible, it helps to simplify our perspective of unbiased PU and NU learning. When estimating a labeled risk, $\widehat{R}_{\mathcal{D}}^{\hat{y}}(g)$ (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, the risk *decomposes* via the simple linear transformation,

$$(1 - \alpha)\widehat{R}_A^{\hat{y}}(g) = \widehat{R}_u^{\hat{y}}(g) - \alpha\widehat{R}_B^{\hat{y}}(g) \tag{10}$$

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 $\widehat{R}_A^{\hat{y}}(g)$ or $\widehat{R}_B^{\hat{y}}(g)$ can always be estimated 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.

Applying Recursive Risk to aPU learning

Our positive-unlabeled recursive risk (PURR) estimator quantifies our aPU setting's empirical risk and integrates into a standard ERM framework. PURR's top-level definition is simply the test risk:

$$\widehat{R}_{\text{PURR}}(g) = \pi_{\text{te}}\widehat{R}_{\text{te-p}}^{+}(g) + (1 - \pi_{\text{te}})\widehat{R}_{\text{te-n}}^{-}(g). \tag{11}$$

Since only unlabeled data is drawn from the test distribution, both terms in Eq. (11) require risk decomposition. First, for $\widehat{R}_{\text{te-n}}^{-}(g)$, we consider its more general form $\widehat{R}_{\text{te-n}}^{\hat{y}}(g)$ below since $\widehat{R}_{\text{te-n}}^{+}(g)$ will be needed as well. Using Eq. (7)'s assumption, $\widehat{R}_{\text{te-n}}^{\hat{y}}(g)$ can be estimated directly from the training distribution. Combining Eq. (3) with absolute-value correction, we see that

$$\widehat{R}_{\text{te-n}}^{\hat{y}}(g) = \widehat{R}_{\text{tr-n}}^{\hat{y}}(g) = \frac{1}{1 - \pi_{\text{tr}}}\left|\widehat{R}_{\text{tr-u}}^{\hat{y}}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{\hat{y}}(g)\right|. \tag{12}$$

Next, $\widehat{R}_{\text{te-p}}^{+}(g)$, as a positive risk, undergoes NU decomposition so (with absolute-value correction):

$$\pi_{\text{te}}\widehat{R}_{\text{te-p}}^{+}(g) = \left|\widehat{R}_{\text{te-u}}^{+}(g) - (1 - \pi_{\text{te}})\widehat{R}_{\text{te-n}}^{+}(g)\right|. \tag{13}$$

Eq. (12) with $\hat{y} = +1$ substitutes for $\widehat{R}_{\text{te-n}}^{+}(g)$ in Eq. (13) yielding $\widehat{R}_{\text{PURR}}(g)$'s complete definition:

$$\widehat{R}_{\text{PURR}}(g) = \underbrace{\left|\widehat{R}_{\text{te-u}}^{+}(g) - (1 - \pi_{\text{te}})\left|\underbrace{\frac{\widehat{R}_{\text{tr-u}}^{+}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{+}(g)}{1 - \pi_{\text{tr}}}}_{\widehat{R}_{\text{te-n}}^{+}(g)}\right|\right|}_{\pi_{\text{te}}\widehat{R}_{\text{te-p}}^{+}(g)} + (1 - \pi_{\text{te}})\left|\underbrace{\frac{\widehat{R}_{\text{tr-u}}^{-}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{-}(g)}{1 - \pi_{\text{tr}}}}_{\widehat{R}_{\text{te-n}}^{-}(g)}\right|. \tag{14}$$

Theorem 3. *Fix decision function $g \in \mathcal{G}$. If ℓ is bounded over $g(x)$'s image and $\widehat{R}_{\text{te-n}}^{\hat{y}}(g), \widehat{R}_{\text{te-p}}^{+}(g) > 0$ 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 $\Pr[\widehat{R}_{\text{tr-u}}^{\hat{y}}(g) - (1 - \pi_{\text{te}})\widehat{R}_{\text{tr-p}}^{\hat{y}}(g) < 0] = 0$ and $\Pr[\widehat{R}_{\text{te-u}}^{+}(g) - (1 - \pi_{\text{te}})\widehat{R}_{\text{te-n}}^{+}(g) < 0] = 0$.*

Optimization   PURR with absolute-value correction integrates into a standard ERM framework. If non-negativity is used instead, PURR's optimization scheme becomes significantly more complicated as it must consider four candidate gradients per update; see suppl. Section C.2 for more details.

7   Experimental Results

We empirically studied the effectiveness of our methods – PURR, PU2wUU, and PU2aPNU – using synthetic and real-world data.[3] Limited space allows us to discuss only two experiment sets here. Suppl. Section E details experiments on: synthetic data, 10 LIBSVM datasets [30] under a totally different positive-bias condition, and a study of our methods' robustness to negative-class shift.

7.1   Experimental Setup

Supplemental Section D enumerates our complete experimental setup with a brief summary below.

Baselines   PUc [14] with a linear-in-parameter model and Gaussian kernel basis is the primary baseline.[4] Ordinary nnPU is the performance floor. To ensure the strongest baseline, we separately trained nnPU with unlabeled set $\mathcal{X}_{\text{te-u}}$ as well as with the combined $\mathcal{X}_{\text{tr-u}} \cup \mathcal{X}_{\text{te-u}}$ (using the true, composite prior) and report each experiment's best performing configuration, denoted nnPU*. PN-test (trained on labeled $\mathcal{X}_{\text{te-u}}$) provides a reference for the performance ceiling. All methods saw identical training/test data splits and where applicable used the same initial weights.

Datasets   Section 7.2 considers the MNIST [31], CIFAR10 [32], and 20 Newsgroups [33] datasets with binary classes formed by partitioning each dataset's labels. Section 7.3 uses two different TREC [34] spam email datasets to demonstrate our methods' performance under real-world adversarial concept drift. Further details on all datasets are in the supplemental materials.

Learner Architecture   We focus on training neural networks (NNs) via stochastic optimization (i.e., AdamW [35] with AMSGrad [36]). Probabilistic classifier, $\hat{\sigma}$, used our abs-PU risk estimator with logistic loss. All other learners used sigmoid loss for ℓ. Since PUc is limited to linear models with Gaussian kernels, we limited our NNs to at most three fully-connected layers of 300 neurons. For MNIST, our NNs were trained from scratch. Pretrained deep networks encoded the CIFAR10, 20 Newsgroups, and TREC spam datasets into static representations all learners used. Specifically, the 20 Newsgroups documents and TREC emails were encoded into 9,216 dimensional vectors using ELMo [37]. This encoding scheme was used by Hsieh et al. [11] and is based on [38]. DenseNet-121 [39] encoded each CIFAR10 image into a 1,024 dimensional vector.

Hyperparameters   Our only individually tuned hyperparameters are learning rate and weight decay. We assume the worst case of no *a priori* knowledge about the positive shift so midpoint value $\rho = 0.5$ was used. PUc's hyperparameters were tuned via importance-weighted cross validation [40]. For the complete hyperparameter details, see supplemental materials Section D.8.

7.2   Partially and Fully Disjoint Positive Class-Conditional Supports

Here we replicate scenarios where positive subclasses exist only in the test distribution (e.g., adversarial zero-day attacks). These experiments are modeled after Hsieh et al. [11]'s experiments for positive, unlabeled, biased-negative (PUbN) learning.

Table 1 lists the experiments' positive train/test and negative class definitions. Datasets are sampled u.a.r. from their constituent sublabels. Each dataset has four experimental conditions (ordered by row number): (1) $P_{\text{train}} = P_{\text{test}}$, i.e., no bias, (2 & 3 resp.) partially disjoint positive supports without and with prior shift, and (4) disjoint positive class definitions. π_{te} equals P_{test}'s true prior w.r.t. $P_{\text{test}} \sqcup N$.

---

[4]The PUc implementation was provided by Sakai and Shimizu [14] via personal correspondence.

Table 1: Mean inductive misclassification rate (%) over 100 trials for MNIST, 20 Newsgroups, & CIFAR10 for different positive & negative class definitions. Bold denotes a *shifted task*'s best performing method. For *all* shifted tasks, our three methods – denoted with [†] – statistically outperformed PUc and nnPU* based on a paired t-test ($p < 0.01$). Each dataset's first three experiments have identical negative (N) & positive-test (P_{test}) class definitions. Positive train (P_{train}) specified as "P_{test}" denotes no bias. Additional shifted tasks (with result standard deviations) are in the supplemental materials.

|	N	P_{test}	P_{train}	π_{tr}	π_{te}	$PURR^{\dagger}$	Two-Step (PU2)		Baselines		Ref.
							$aPNU^{\dagger}$	$wUU^{\dagger}$	PUc	nnPU*	PN_{te}
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	P_{test}	0.5	0.5	10.0	10.0	11.6	8.6	5.5	↑
			7, 9	0.5	0.5	9.4	**7.1**	8.3	26.8	35.1	2.8
				0.29	0.5	6.8	**5.3**	6.0	29.2	36.7	↓
	0, 2	5, 7	1, 3	0.5	0.5	4.0	3.6	**3.1**	17.1	30.9	1.1
20 News.	sci, soc, talk	alt, comp, misc, rec	P_{test}	0.56	0.56	15.4	14.9	16.7	14.9	14.1	↑
			misc, rec	0.56	0.56	17.5	**13.5**	15.1	23.9	28.8	10.5
				0.37	0.56	13.9	**12.8**	14.3	28.9	28.8	↓
	misc, rec	soc, talk	alt, comp	0.55	0.46	5.9	7.1	**5.6**	18.5	35.3	2.1
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	P_{test}	0.4	0.4	14.1	14.2	15.5	13.8	12.3	↑
			Plane	0.4	0.4	**13.8**	14.5	15.1	20.6	27.4	9.8
				0.14	0.4	12.1	**11.9**	12.4	26.7	26.7	↓
	Deer, Horse	Plane, Auto	Cat, Dog	0.5	0.5	14.1	14.9	**11.2**	33.1	47.5	7.7
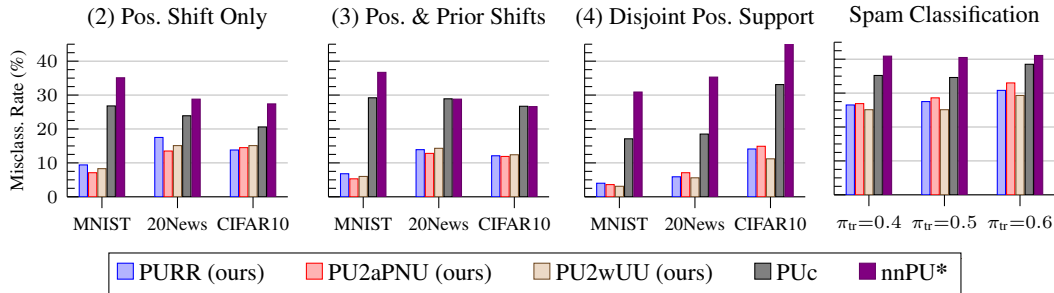
Figure 2: Mean inductive misclassification rate over 100 trials on the MNIST, 20 News., CIFAR10, & TREC spam datasets for our methods & baselines. Each numbered plot (i.e., 2–4) corresponds to one experimental shift task in Table 1. Spam classification experiments are detailed in Section 7.3.

By default $\pi_{tr} = \pi_{te}$; in the prior shift and disjoint support experiments (rows 3 and 4), π_{tr} equals P_{train}'s true prior w.r.t. $P_{train} \sqcup N$.

Analysis Results are shown in Table 1 and Figure 2. On *unshifted* data (row 1 for each dataset), baselines PUc and nnPU* slightly outperformed our methods, which shows that PUc's architecture is sufficiently expressive. In contrast, on *shifted* data (rows 2–4 for each dataset), our methods' performance generally improved while both PUc's and nnPU*'s performance always degraded. This performance divergence demonstrates our methods' algorithmic advantage. In fact for all shifted tasks, our methods always outperformed PUc and nnPU* according to a paired t-test ($p < 0.01$). For partially disjoint positive supports (rows 2 and 3 for each dataset), PU2aPNU was the top performer for five of six setups (PURR was top on the other). This pattern reversed for fully disjoint supports (row 4) where PU2aPNU always lagged PU2wUU; this is expected as explained in Section 5.

Reducing π_{tr} always improved our algorithms' performance and degraded PUc's. A smaller prior enables easier identification of $\mathcal{X}_{tr-u}$'s negative examples and in turn a more accurate estimation of $\mathcal{X}_{te-u}$'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.

Table 2: Mean inductive misclassification rate (%) over 100 trials for spam adversarial drift. Our methods – PURR, PU2wUU, and PU2aPNU – outperformed PUc & nnPU* based on a 1% paired t-test. Each result's standard deviation appears in supplemental Table 14.

Train Set		Test Set		π_{tr}	π_{te}	Two-Step (PU2)			Baselines		Ref.
Pos.	Neg.	Pos.	Neg.			PURR	aPNU	wUU	PUc	nnPU*	PN_{te}
2005 Spam	2005 Ham	2007 Spam	2007 Ham	0.4	0.5	26.5	26.9	**25.1**	35.2	40.9	↑
				0.5	0.5	27.5	28.6	**25.1**	34.6	40.5	0.6
				0.6	0.5	30.8	33.0	**29.3**	38.5	41.1	↓

nnPU* outperformed both PUc and our methods when there was no bias. This is expected. If an algorithm searches for non-existent phenomena, any additional patterns found will not generalize.

7.3 Case Study: Arbitrary Adversarial Concept Drift

PU learning has been applied to multiple adversarial domains including opinion spam [3, 16, 17, 18]. We use spam classification as a vehicle to test our methods in an adversarial setting by considering two different TREC email spam datasets – training on TREC05 and evaluating on TREC07. Spam – the positive class – evolves quickly over time, but the two datasets' ham emails are also quite different: TREC05 relies on Enron emails while TREC07 contains mostly emails from a university server. Thus, this represents a more challenging, realistic setting where Eq. (7)'s assumption does not hold.

Table 2 and Figure 2 show that our methods outperformed PUc and nnPU* according to a 1% paired t-test across three training priors (π_{tr}). PU2wUU was the top-performer as $\hat{\sigma}$ accurately labeled $\mathcal{X}_{tr-u}$, yielding a strong surrogate negative set. PU2aPNU performed slightly worse than PU2wUU as the significant adversarial concept drift greatly limited $\mathcal{X}_p$'s value. Overall, these experiments show that our aPU setting arises in real-world domains. All of our methods handled large positive shifts better than prior work, even in realistic cases where the negative class also shifts.

7.4 Discussion

Our two-step methods assume asymptotic consistency for $\widetilde{\mathcal{X}}_n$ in step #1, but finite training data ensures a non-consistent evaluation setting. Nonetheless, either PU2aPNU or PU2wUU was the top performer in all but one experiment in this section.[5] Supplemental Section E.7 includes additional experiments where we further stress our two-step methods by forcing $\hat{\sigma}$ away from our posterior estimate. Even under those deleterious step #1 conditions, our two-step learners are robust.

Conventional wisdom suggests that joint method PURR should outperform pipeline approaches. This intuition breaks down in our case because PURR, with its three risk decompositions, is strictly harder to optimize than wUU, aPNU, abs-PU, and nnPU – all of which have a single decomposition. This harder optimization can lead to worse accuracy compared to the two-step methods, especially on easier problems (e.g., MNIST), where each step can be solved accurately on its own.

For completeness, suppl. Section E.5 compares our methods to bPU selection bias method PUSB [13]. Our algorithms generally outperformed PUSB on data specifically tuned for their method even after accounting for the differing unlabeled sets. Those experiments indicate that PUSB's underlying assumption entails only a small data shift and further point to potential PUSB learning brittleness.

8 Conclusions

We examined arbitrary-positive, unlabeled (aPU) learning, where the labeled-positive and target-positive distributions may 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). Empirical results on real-world data above and in the supplementals show that our methods are still robust in the realistic case of some negative shift. Future work seeks a less restrictive yet statistically-sound replacement assumption of a fixed negative class-conditional distribution.

---

[5]Supplemental Sections E.2 and E.4 enumerate multiple empirical setups where PURR is the top performer.

9 Broader Impact

The algorithms proposed in this work are general and could be applied to many different applications. Forecasting the broader impact of work like this is challenging and generally inaccurate. With that caveat, we discuss potential impacts based on possible applications.

The case study on email spam suggests that our methods may be useful in adversarial domains, such as the detection of fraud, malware, network intrusion, distributed denial of service (DDoS) attacks, and many types of spam. In these settings, one class (e.g., spam) evolves quickly as attackers try to evade detection. For many of these domains, improved classifiers would benefit society by reducing spam and fraud. However, for domains such as facial recognition, improved robustness could lead to reduced privacy and other societal harms. See Albert et al. [41] for an extensive discussion of the politics of adversarial machine learning.

In other domains, such as epidemiological analysis and land-cover classification, our work may lead to new or better models by reducing the need for labeled data and relaxing the SCAR assumption. As detailed in Section 1, only recently has the PU SCAR barrier been broken [12, 13, 14]. aPU learning pushes PU learning's positive-shift boundary to a new extreme. We hope this paper will enable PU learning to be applied in domains where existing bPU\PU methods are impractical. This could also benefit society if used responsibly, with experts performing proper model validation and vetting risks. Careful model validation is especially important when labeled data is limited and biased.

References

[1] Marthinus du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *Proceedings of the 32nd International Conference on Machine Learning*, ICML'15, 2015.

[2] Jessa Bekker and Jesse Davis. Estimating the class prior in positive and unlabeled data through decision tree induction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, AAAI'18, 2018.

[3] Donato Hernández Fusilier, Rafael Guzmán Cabrera, Manuel Montes-y Gómez, and Paolo Rosso. Using PU-learning to detect deceptive opinion spam. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 38–45, 6 2013.

[4] Peng Yang, Xiao-Li Li, Jian-Ping Mei, Chee-Keong Kwoh, and See-Kiong Ng. Positive-unlabeled learning for disease gene identification. *Bioinformatics*, 28(20):2640–2647, 08 2012.

[5] Wenkai Li, Qinghua Guo, and Charles Elkan. A positive and unlabeled learning algorithm for one-class classification of remote-sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, 49:717 – 725, 2011.

[6] Charles Elkan and Keith Noto. 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, pages 213–220, 2008.

[7] Ming Hou, Brahim Chaib-Draa, Chao Li, and Qibin Zhao. Generative adversarial positive-unlabeled learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 2255–2261, 2018.

[8] Ryuichi Kiryo, Gang Niu, Marthinus C. du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Proceedings of the 30th Conference on Neural Information Processing Systems*, NeurIPS'17, pages 1674–1684, 2017.

[9] Tieliang Gong, Guangtao Wang, Jieping Ye, Zongben Xu, and Ming Lin. Margin based PU learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, AAAI'18, 2018.

[10] Chuang Zhang, Dexin Ren, Tongliang Liu, Jian Yang, and Chen Gong. Positive and unlabeled learning with label disambiguation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, pages 4250–4256, 2019.

[11] Yu-Guan Hsieh, Gang Niu, and Masashi Sugiyama. Classification from positive, unlabeled and biased negative data. In *Proceedings of the 36th International Conference on Machine Learning*, ICML'19, pages 2820–2829, 2019.

[12] Jessa Bekker, Pieter Robberechts, and Jesse Davis. Beyond the selected completely at random assumption for learning from positive and unlabeled data. In *Proceedings of the 2019 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, ECML-PKDD'19, 2019.

[13] Masahiro Kato, Takeshi Teshima, and Junya Honda. Learning from positive and unlabeled data with a selection bias. In *Proceedings of the 7th International Conference on Learning Representations*, ICLR'19, 2019.

[14] Tomoya Sakai and Nobuyuki Shimizu. Covariate shift adaptation on learning from positive and unlabeled data. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, AAAI'19, pages 4838–4845, 2019.

[15] Galen Maclaurin and Stefan Leyk. Extending the geographic extent of existing land cover data using active machine learning and covariate shift corrective sampling. *International Journal of Remote Sensing*, 37: 5213–5233, 11 2016.

[16] Huayi Li, Zhiyuan Chen, Bing Liu, Xiaokai Wei, and Jidong Shao. Spotting fake reviews via collective positive-unlabeled learning. In *Proceedings of the 14th IEEE International Conference on Data Mining*, ICDM'14, page 899–904, 2014.

[17] Ya-Lin Zhang, Longfei Li, Jun Zhou, Xiaolong Li, Yujiang Liu, Yuanchao Zhang, and Zhi-Hua Zhou. POSTER: A PU learning based system for potential malicious URL detection. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS'17, page 2599–2601, 2017.

[18] J. Zhang, M. F. Khan, X. Lin, and Z. Qin. An optimized positive-unlabeled learning method for detecting a large scale of malware variants. In *Proceedings of the 2019 IEEE Conference on Dependable and Secure Computing*, DSC'19, 2019.

[19] Gang Niu, Marthinus C. du Plessis, Tomoya Sakai, Yao Ma, and Masashi Sugiyama. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *Proceedings of the 29th Conference on Neural Information Processing Systems*, NeurIPS'16, page 1207–1215, 2016.

[20] Marthinus C du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Proceedings of the 27th Conference on Neural Information Processing Systems*, NeurIPS'14, 2014.

[21] Harish G. Ramaswamy, Clayton Scott, and Ambuj Tewari. Mixture proportion estimation via kernel embedding of distributions. In *Proceedings of the 33rd International Conference on Machine Learning*, ICML'16, page 2052–2060, 2016.

[22] Marthinus C. du Plessis, Gang Niu, and Masashi Sugiyama. Class-prior estimation for learning from positive and unlabeled data. *Machine Learning*, 106(4):463–492, 2017.

[23] Daniel Zeiberg, Shantanu Jain, and Predrag Radivojac. Fast nonparametric estimation of class proportions in the positive-unlabeled classification setting. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, AAAI'20, 2020.

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

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

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

[27] Tomoya Sakai, Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. Semi-supervised classification based on classification from positive and unlabeled data. In *Proceedings of the 34th International Conference on Machine Learning*, ICML'17, pages 2998–3006, 2017.

[28] Aditya Menon, Brendan Van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *Proceedings of the 32nd International Conference on Machine Learning*, ICML'15, pages 125–134, 2015.

[29] Nan Lu, Gang Niu, Aditya Krishna Menon, and Masashi Sugiyama. On the minimal supervision for training any binary classifier from only unlabeled data. In *Proceedings of the 7th International Conference on Learning Representations*, ICLR'19, 2019.

[30] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[32] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset, 2014.

[33] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, ICML'95, pages 331–339, 1995.

[34] TREC. Text REtrieval Conference (TREC) overview. `https://trec.nist.gov/overview.html`, 2019 (accessed May 19, 2020).

[35] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in Adam. *CoRR*, abs/1711.05101, 2017. URL `http://arxiv.org/abs/1711.05101`.

[36] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *Proceedings of the 6th International Conference on Learning Representations*, ICLR'18, 2018.

[37] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL'18, 2018.

[38] Andreas Rücklé, Steffen Eger, Maxime Peyrard, and Iryna Gurevych. Concatenated power mean embeddings as universal cross-lingual sentence representations. *CoRR*, abs/1803.01400, 2018. URL `http://arxiv.org/abs/1803.01400`.

[39] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'17, pages 2261–2269, 2017.

[40] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, Dec. 2007.

[41] Kendra Albert, Jonathon Penney, Bruce Schneier, and Ram Shankar Siva Kumar. Politics of adversarial machine learning. In *ICLR Workshop Towards Trustworthy ML: Rethinking Security and Privacy for ML*, 2020.

[42] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, NeurIPS'19, 2019.

[44] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv*, 2017.

[45] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical Japanese literature. *NeurIPS'18 Workshop on Machine Learning for Creativity and Design*, 2018.

[46] Jason Rennie. 20 newsgroups. `http://qwone.com/~jason/20Newsgroups/`, 2001.

[47] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, ICML'15, pages 448–456, 2015.

[48] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of the 29th Conference on Neural Information Processing Systems*, 2015.

[49] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR'15, 2015.

[50] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, ICML'17, 2017.

Learning from Positive and Unlabeled Data
with Arbitrary Positive Shift

Supplemental Materials

A Nomenclature

Table 3: aPU nomenclature reference

PN	Positive-negative learning, i.e., ordinary supervised classification
PU	Positive-unlabeled learning
NU	Negative-unlabeled learning
uPU	Unbiased Positive-Unlabeled risk estimator from [20]. See Section 2
nnPU	Non-negative Positive-Unlabeled risk estimator from [8]. See Section 2
abs-PU	Our Absolute-value Positive-Unlabeled risk estimator. See Section 3
bPU	Biased-positive, unlabeled learning where the labeled-positive set is not representative of the target positive class. bPU algorithm categories include sample selection bias [12, 13] and covariate shift methods [14]
aPU	Proposed in this work, arbitrary-positive, unlabeled learning generalizes bPU learning where the positive training data may be arbitrarily different from the target application's positive-class distribution
PUc	Positive-Unlabeled Covariate shift algorithm from [14]. See Section 4.1
PU2wUU	Our Positive-Unlabeled to Weighted Unlabeled-Unlabeled (two-step) aPU learner. See Section 5
PU2aPNU	Our Positive-Unlabeled to Arbitrary-Positive, Negative, Unlabeled (two-step) aPU learner. See Section 5
PURR	Our Positive-Unlabeled Recursive Risk (one-step) aPU estimator. See Section 6
abs-PU	Our Absolute-value Positive-Unlabeled risk estimator. See Section 3
nnPU*	Version of nnPU used as an empirical baseline. nnPU* considers two classifiers – one trained with $\mathcal{X}_{\text{te-u}}$ as the unlabeled set and the other trained with $\mathcal{X}_{\text{tr-u}} \cup \mathcal{X}_{\text{te-u}}$ as the unlabeled set – and reports whichever configuration performed better. See Section 7
abs-PU*	Baseline equivalent of nnPU* except risk estimator $\widehat{R}_{\text{abs-PU}}(g)$ is used instead of $\widehat{R}_{\text{nnPU}}(g)$. See Section E.6.2
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, $\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$
$p_{\mathcal{T}}(x, y)$	Joint distribution, where $\mathcal{T} \in \{\text{tr}, \text{te}\}$ for train and test resp.
$\pi_{\mathcal{T}}$	Positive-class prior probability, $\pi_{\mathcal{T}} := p_{\mathcal{T}}(Y = +1)$ where $\mathcal{T} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{T}\text{-p}}(x)$	Positive class-conditional $p_{\mathcal{T}\text{-p}}(x) := p_{\mathcal{T}}(x \mid Y = +1)$ where $\mathcal{T} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{T}\text{-n}}(x)$	Negative class-conditional $p_{\mathcal{T}\text{-n}}(x) := p_{\mathcal{T}}(x \mid Y = -1)$ where $\mathcal{T} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{T}\text{-u}}(x)$	Marginal distribution where $p_{\mathcal{T}\text{-u}}(x) := p_{\mathcal{T}}(x)$ where $\mathcal{T} \in \{\text{tr}, \text{te}\}$ 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 the *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 the *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 $\hat{\sigma}$
$\mathcal{X}_{\text{n}}$	Labeled negative dataset. In PU learning, $\mathcal{X}_{\text{n}} = \emptyset$
$\widetilde{\mathcal{X}}_{\text{n}}$	Surrogate negative set formed by reweighting $\mathcal{X}_{\text{tr-u}}$ by $\hat{\sigma}$
$R(g)$	Risk, i.e., expected loss, for decision function g and loss ℓ, i.e., $R(g) := \mathbb{E}_{(X,Y) \sim p(x,y)}[\ell(Y g(X))]$
$\widehat{R}(g)$	Empirical estimate of risk $R(g)$

$\widehat{R}_{\mathcal{D}}^{\hat{y}}(g)$	Empirical risk when predicting label $\hat{y} \in \{\pm 1\}$ on data sampled from some distribution, $p_{\mathcal{D}}(x)$. See Section 2
$\ddot{R}_{\mathrm{n}}^{\hat{y}}(g)$	Labeled negative risk with absolute-value correction. See Eq. (5) in Section 3
$\widetilde{R}_{\text{n-u}}^{\hat{y}}(g)$	Surrogate negative risk formed by weighting unlabeled set $\mathcal{X}_{\text{tr-u}}$ by probabilistic classifier $\hat{\sigma}$ where $\widetilde{R}_{\text{n-u}}^{\hat{y}}(g) := \frac{1}{n_{\text{tr-u}}} \sum_{x_i \in \mathcal{X}_{\text{tr-u}}} \frac{\hat{\sigma}(x_i)\ell(\hat{y}g(x_i))}{1-\pi_{\text{tr}}}$
$w(x)$	Covariate shift importance function based on density-ratio estimation where $w(x) := \frac{p_{\text{te-u}}(x)}{p_{\text{tr-u}}(x)}$
n_{p}	Size of the labeled (positive) dataset, i.e., $n_{\mathrm{p}} := |\mathcal{X}_{\mathrm{p}}|$
$n_{\text{tr-u}}$	Size of the unlabeled *training* dataset, i.e., $n_{\text{tr-u}} := |\mathcal{X}_{\text{tr-u}}|$
$n_{\text{te-u}}$	Size of the unlabeled *test* dataset, i.e., $n_{\text{te-u}} := |\mathcal{X}_{\text{te-u}}|$
n_{Test}	Size of the inductive test set
$\mathcal{A}$	Learning or optimization algorithm
η	Learning rate hyperparameter, $\eta > 0$
λ	Weight decay hyperparameter, $\lambda \geq 0$
γ	Non-negative gradient attenuator hyperparameter $\gamma \in (0, 1]$. This hyperparameter is ignored when absolute-value correction is used.
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{I}_m)$	Multivariate Gaussian (normal) distribution with mean $\boldsymbol{\mu}$ and m-dimensional identity covariance. See Section E.1
$[a]_+$	$:= \max\{0, a\}$. See Section C.2
$\lfloor a \rceil$	Rounds $a \in \mathbb{R}$ to the nearest integer. See Section E.7

B Proofs

B.1 Proof of Theorem 1

Proof. Mild assumptions are made about the behavior of the loss and decision functions; the following conditions match those assumed by Kiryo et al. [8]. 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 $\sup_{|t| \leq C_g} \max_{\hat{y} \in \{\pm 1\}} \ell(\hat{y}t) \leq C_\ell$.

du Plessis et al. [20] show that

$$(1 - \pi)R_{\mathrm{n}}^{\hat{y}}(g) = R_{\mathrm{u}}^{\hat{y}}(g) - \pi R_{\mathrm{p}}^{\hat{y}}(g). \tag{15}$$

Consider the labeled negative-valued risk estimator with absolute-value correction

$$\ddot{R}_{\mathrm{n}}^{\hat{y}}(g) = \left| \widehat{R}_{\mathrm{n}}^{\hat{y}}(g) \right|. \tag{16}$$

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

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

Let estimator $\hat{Y} = \sum_{i=1}^k \beta_i \hat{\theta}_{(i)}$ be the weighted sum of k consistent estimators with each constant $\beta_i \neq 0$. 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 $\ddot{R}_{\mathrm{n}}^{\hat{y}}(g)$ in Eq. (16), it is sufficient to show that each of its individual terms is consistent.

Both $\widehat{R}_{\mathrm{p}}^{\hat{y}}(g)$ and $\widehat{R}_{\mathrm{u}}^{\hat{y}}(g)$ are empirically estimated directly from a training data set. Let $\mathcal{D} \in \{\mathrm{p}, \mathrm{u}\}$ and $\mathcal{X}_{\mathcal{D}} \overset{\text{i.i.d.}}{\sim} p_{\mathcal{D}}(x)$. For each (independent) $X \sim p_{\mathcal{D}}(x)$, $\ell(\hat{y}g(X))$ is an unbiased estimate of $R_{\mathcal{D}}^{\hat{y}}(g)$. In addition, $\ell(\hat{y}g(X)) < C_\ell < \infty$ implies that $\mathrm{Var}(\ell(\hat{y}g(X))) < \infty$. By Chebyshev's Inequality, $\widehat{R}_{\mathcal{D}}^{\hat{y}}(g)$ is consistent as

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

Since $\widehat{R}_{\mathrm{n}}^{\hat{y}}(g)$ is the weighted sum of consistent estimators, it is consistent as $n = \min\{n_{\mathrm{p}}, n_{\mathrm{u}}\} \to \infty$.

To show $\ddot{R}_{\mathrm{n}}^{\hat{y}}(g)$ is consistent, it suffices to show that

$$\lim_{n \to \infty} \Pr\left[\left|\ddot{R}_{\mathrm{n}}^{\hat{y}}(g) - R_{\mathrm{n}}^{\hat{y}}(g)\right| \geq \epsilon\right] = 0.$$

Because $\widehat{R}_{\mathrm{n}}^{\hat{y}}(g)$ is consistent, then as $n \to \infty$ it holds that $\widehat{R}_{\mathrm{n}}^{\hat{y}}(g) - \epsilon \leq R_{\mathrm{n}}^{\hat{y}}(g) \leq \widehat{R}_{\mathrm{n}}^{\hat{y}}(g) + \epsilon$. When $\widehat{R}_{\mathrm{n}}^{\hat{y}}(g) \geq R_{\mathrm{n}}^{\hat{y}}(g) \geq 0$, then $\ddot{R}_{\mathrm{n}}^{\hat{y}}(g) = \widehat{R}_{\mathrm{n}}^{\hat{y}}(g)$ (i.e., absolute value has no effect) so

$$0 \leq \ddot{R}_{\mathrm{n}}^{\hat{y}}(g) - R_{\mathrm{n}}^{\hat{y}}(g) \leq \epsilon.$$

Consider the alternate possibility where $\widehat{R}_{\mathrm{n}}^{\hat{y}}(g) < R_{\mathrm{n}}^{\hat{y}}(g)$. If $\widehat{R}_{\mathrm{n}}^{\hat{y}}(g) \geq 0$ or $R_{\mathrm{n}}^{\hat{y}}(g) = 0$, then absolute-value correction again has no effect on the estimation error (i.e., remains $\leq \epsilon$). Lastly, when $\widehat{R}_{\mathrm{n}}^{\hat{y}}(g) < 0$ and $R_{\mathrm{n}}^{\hat{y}}(g) > 0$, the estimation error strictly decreases as

$$
\begin{aligned}
\mathrm{err}_{\hat{R}} &= \left|\widehat{R}_{\mathrm{n}}^{\hat{y}}(g) - R_{\mathrm{n}}^{\hat{y}}(g)\right| \\
&= -\widehat{R}_{\mathrm{n}}^{\hat{y}}(g) + R_{\mathrm{n}}^{\hat{y}}(g) && \text{Since } \widehat{R}_{\mathrm{n}}^{\hat{y}}(g) < 0 \text{ and } R_{\mathrm{n}}^{\hat{y}}(g) > 0 \\
&= \left|\widehat{R}_{\mathrm{n}}^{\hat{y}}(g)\right| + R_{\mathrm{n}}^{\hat{y}}(g) && \text{Again since } \widehat{R}_{\mathrm{n}}^{\hat{y}}(g) < 0 \\
&= \ddot{R}_{\mathrm{n}}^{\hat{y}}(g) + R_{\mathrm{n}}^{\hat{y}}(g) < \epsilon
\end{aligned}
$$

so

$$
\begin{aligned}
\mathrm{err}_{\ddot{R}} &= \left|\left|\widehat{R}_{\mathrm{n}}^{\hat{y}}(g)\right| - R_{\mathrm{n}}^{\hat{y}}(g)\right| \\
&=: \left|\ddot{R}_{\mathrm{n}}^{\hat{y}}(g) - R_{\mathrm{n}}^{\hat{y}}(g)\right| \\
&< \ddot{R}_{\mathrm{n}}^{\hat{y}}(g) + R_{\mathrm{n}}^{\hat{y}}(g) < \epsilon && \text{Since } \widehat{R}_{\mathrm{n}}^{\hat{y}}(g) < 0 \text{ and } R_{\mathrm{n}}^{\hat{y}}(g) > 0. && (17)
\end{aligned}
$$

The above shows that as $n \to \infty$, it always holds that $\left|\ddot{R}_{\mathrm{n}}^{\hat{y}}(g) - R_{\mathrm{n}}^{\hat{y}}(g)\right| \leq \epsilon$ for arbitrary $\epsilon > 0$ making $\ddot{R}_{\mathrm{n}}^{\hat{y}}(g)$ consistent. □

B.2 Proof of Theorem 2

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

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

satisfying the definition of unbiased.

Next we consider whether $\widetilde{R}_{\mathrm{n\text{-}u}}^{\hat{y}}(g)$ is a consistent estimator of $R_{\mathrm{n}}^{\hat{y}}(g)$. For the complete definition of PAC learnability that we use here, see [42]. We provide a brief sketch of the definition below.

We assume that true posterior distribution, $p_{\mathrm{tr}}(Y = -1|x)$ 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 $n = \min\{n_{\mathrm{p}}, n_{\mathrm{tr\text{-}u}}\}$.[6] Consider the *realizable* setting so $\mathcal{C}$'s PAC learnability entails that for all

---

[6] No restrictions are placed on $\mathcal{A}$ other than its existence and that selected hypothesis $\hat{\sigma}_\mathcal{S}$ satisfies Eq. (18).

$\epsilon, \delta > 0$, there exists an n' such that for all $n > n'$,

$$\Pr\left[\mathbb{E}_{X \sim p_{\text{tr-u}}(x)}[|\hat{\sigma}_\mathcal{S}(X) - p_{\text{tr}}(Y = -1|X)|] > \epsilon\right] < \delta. \tag{18}$$

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 $\sup_{|t| \le \|g\|_\infty} \ell(\hat{y}t) \le C_\ell$, 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 $\frac{1}{(1-\pi_{\text{tr}})^2}\text{Var}(\hat{\sigma}(X)\ell(\hat{y}g(X))) \le C_{\text{var}}$ where $C_{\text{var}} \in \mathbb{R}_{\ge 0}$ and $\pi_{\text{tr}} \in [0, 1)$.

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

$$\lim_{n \to \infty} \Pr\left[\left|\widetilde{R}^{\hat{y}}_{\text{n-u}}(g) - R^{\hat{y}}_{\text{tr-n}}(g)\right| \ge \epsilon\right] \le \frac{\text{Var}(\widetilde{R}^{\hat{y}}_{\text{n-u}}(g))}{\epsilon^2}$$

$$= \frac{1}{(1-\pi_{\text{tr}})^2 \epsilon^2} \sum_{i=1}^{n_{\text{tr-u}}} \text{Var}\left(\frac{\hat{\sigma}(X)\ell(\hat{y}g(X))}{n_{\text{tr-u}}}\right) \quad \text{Linearity of independent r.v. var.}$$

$$\le \frac{n_{\text{tr-u}} C_{\text{var}}}{n_{\text{tr-u}}^2 \epsilon^2}$$

$$= 0 \qquad\qquad\qquad\qquad\qquad \text{L'Hôpital's Rule.}$$

$\square$

B.3 Proof Regarding Estimating π_{te}

We are not aware of an existing technique to directly estimate the test distribution's positive prior π_{te} given only $\mathcal{X}_{\text{p}}$, $\mathcal{X}_{\text{tr-u}}$, and $\mathcal{X}_{\text{te-u}}$. We propose the following that uses an additional classifier.

Theorem 4. *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 class-conditional 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. $\square$

Commentary Theorem 4 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 2'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. (7)'s assumption, $\mathcal{X}_{\text{tr-n}}$ is representative of samples from the negative class-conditional 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 [21, 22] 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.9.

B.4 Proof of Theorem 3

The definition of "bounded loss" is identical to the proof of Theorem 1.

Proof. Consider first whether PURR is unbiased. du Plessis et al. [20] observe that the negative labeled risk can be found via decomposition where

$$(1 - \pi)R_{\text{n}}^{\hat{y}}(g) = R_{\text{u}}^{\hat{y}}(g) - \pi R_{\text{p}}^{\hat{y}}(g). \tag{19}$$

The positive labeled risk similarly decomposes as

$$\pi R_{\text{p}}^{\hat{y}}(g) = R_{\text{u}}^{\hat{y}}(g) - (1 - \pi)R_{\text{n}}^{\hat{y}}(g). \tag{20}$$

Applying these decompositions along with Eq. (7)'s assumption yields an unbiased version of PURR:

$$\widehat{R}_{\text{uPURR}}(g) = \widehat{R}_{\text{te-u}}^{+}(g) - \underbrace{(1 - \pi_{\text{te}}) \underbrace{\frac{\widehat{R}_{\text{tr-u}}^{+}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{+}(g)}{1 - \pi_{\text{tr}}}}_{\widehat{R}_{\text{te-n}}^{+}(g)}}_{\pi_{\text{te}}\widehat{R}_{\text{te-p}}^{+}(g)} + (1 - \pi_{\text{te}}) \underbrace{\frac{\widehat{R}_{\text{tr-u}}^{-}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{-}(g)}{1 - \pi_{\text{tr}}}}_{\widehat{R}_{\text{te-n}}^{-}(g)}. \tag{21}$$

Since $\forall_t \ell(t) \geq 0$, it always holds that labeled risk $R_{\mathcal{D}}^{\hat{y}}(g) \geq 0$. When using risk decomposition (i.e., Eqs. (19) and (20)) to empirically estimate a labeled risk, it can occur that $\widehat{R}_{\mathcal{D}}^{\hat{y}}(g) < 0$. Absolute-value correction addresses these obviously implausible risk estimates. The unrolled definition of the PURR risk estimator with absolute-value correction is:

$$\widehat{R}_{\text{PURR}}(g) = \left| \underbrace{\widehat{R}_{\text{te-u}}^{+}(g) - (1 - \pi_{\text{te}}) \underbrace{\left| \frac{\widehat{R}_{\text{tr-u}}^{+}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{+}(g)}{1 - \pi_{\text{tr}}} \right|}_{\widehat{R}_{\text{te-n}}^{+}(g)}}_{\pi_{\text{te}}\widehat{R}_{\text{te-p}}^{+}(g)} \right| + (1 - \pi_{\text{te}}) \left| \underbrace{\frac{\widehat{R}_{\text{tr-u}}^{-}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{-}(g)}{1 - \pi_{\text{tr}}}}_{\widehat{R}_{\text{te-n}}^{-}(g)} \right|. \tag{22}$$

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. This only occurs if the absolute-value is never needed, i.e., has probability 0 of occurring.

Next consider whether PURR is consistent. Theorem 1 showed that $\ddot{R}_{\text{n}}^{\hat{y}}(g)$ is consistent. Following the same logic in Theorem 1's proof, it is straightforward to see that when performing decomposition on $R_{\text{p}}^{\hat{y}}(g)$, $\ddot{R}_{\text{p}}^{\hat{y}}(g)$ is also consistent.

It follows by induction that PURR (and any similarly-defined recursive risk estimator) is consistent. Theorem 1 shows the consistency of the base case where both composite terms (e.g., $R_{\text{u}}^{\hat{y}}(g)$ and $R_{B}^{\hat{y}}(g)$ in Eq. (10)) were estimated directly from training data. By induction, it is again straightforward from Theorem 1 that any decomposed term (e.g., $R_{A}^{\hat{y}}(g)$ in Eq. (10)) formed from the sum of consistent estimators must be itself consistent.

Theorem 1 further demonstrated that applying absolute-value correction does not affect the consistency of a risk estimator. Therefore, any recursive risk estimator with absolute-value correction is consistent. PURR's consistency is just a single, specific example of this general property. □

C Non-Negativity Correction Empirical Risk Minimization Algorithms

Kiryo et al. [8]'s non-negativity correction algorithm uses the $\max\{0, \cdot\}$ term to ensure a plausible risk estimate. Unlike our simpler absolute-value correction described in Section 3, Kiryo et al.'s non-negativity correction requires a custom empirical risk minimization (ERM) procedure. This section presents the custom ERM algorithms required if non-negativity correction is used for our two-step methods and PURR.

C.1 Two-Step, Non-Negativity ERM Algorithm

The weighted-unlabeled, unlabeled (wUU) risk estimator with non-negativity correction is defined as:

$$\widehat{R}_{\text{nn-wUU}}(g) := \max\left\{0, \widehat{R}_{\text{te-u}}^{+}(g) - (1 - \pi_{\text{te}})\widetilde{R}_{\text{n-u}}^{+}(g)\right\} + (1 - \pi_{\text{te}})\widetilde{R}_{\text{n-u}}^{-}(g). \tag{23}$$

The arbitrary-positive, negative, unlabeled (aPNU) risk estimator with non-negativity correction is similarly defined as:

$$\widehat{R}_{\text{nn-aPNU}}(g) := (1 - \rho)\pi_{\text{te}}R_{\text{p}}^{+}(g) + (1 - \pi_{\text{te}})\widetilde{R}_{\text{n-u}}^{-}(g) + \rho\max\left\{0, R_{\text{te-u}}^{+}(g) - (1 - \pi_{\text{te}})\widetilde{R}_{\text{n-u}}^{+}(g)\right\}. \tag{24}$$

Like their counterparts with absolute-value correction, both $\widehat{R}_{\text{nn-wUU}}(g)$ and $\widehat{R}_{\text{nn-aPNU}}(g)$ are consistent estimators.

Algorithm 2 shows the custom ERM framework for $\widehat{R}_{\text{nn-wUU}}(g)$ and $\widehat{R}_{\text{nn-aPNU}}(g)$ with integrated "defitting." The algorithm learns parameters θ for decision function g. The non-negativity correction occurs whenever $\widehat{R}_{\text{te-u}}^{+}(g) - (1 - \pi_{\text{te}})\widehat{R}_{\text{n-u}}^{+}(g) < 0$ (see line 7). The basic algorithm is heavily influenced by the stochastic optimization algorithm proposed by Kiryo et al. [8].

Algorithm 2 wUU and aPNU with non-negativity correction custom ERM procedure

---

Input: Datasets $(\mathcal{X}_{\text{p}}, \widetilde{\mathcal{X}}_{\text{n}}, \mathcal{X}_{\text{te-u}})$, hyperparameters (γ, η) and risk estimator $\widehat{R}_{\text{TS}}(g) \in \{\widehat{R}_{\text{nn-wUU}}(g), \widehat{R}_{\text{nn-aPNU}}(g)\}$
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 $(\mathcal{X}_{\text{p}}^{(i)}, \widetilde{\mathcal{X}}_{\text{n}}^{(i)}, \mathcal{X}_{\text{te-u}}^{(i)})$ **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 θ by $\mathcal{A}$ with attenuated learning rate $\gamma\eta$
8:        **else**
9:            Set gradient $\nabla_{\theta}\widehat{R}_{\text{TS}}(g)$
10:           Update θ by $\mathcal{A}$ with default learning rate η
11: **return** θ minimizing validation loss

Algorithm 2 terminates after a fixed epoch count (see Table 9 for the number of epochs used for each dataset). Although not shown in Algorithm 2, the validation loss is measured at the end of each epoch. The algorithm returns the model parameters with the lowest validation loss.

C.2 PURR Non-Negativity ERM Algorithm

For readability and compactness, let $[a]_{+} := \max\{0, a\}$. PURR with non-negativity correction is defined as

$$\widehat{R}_{\text{nn-PURR}}(g) := \underbrace{\left[\widehat{R}_{\text{te-u}}^{+}(g) - (1 - \pi_{\text{te}})\underbrace{\left[\frac{\widehat{R}_{\text{tr-u}}^{+}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{+}(g)}{1 - \pi_{\text{tr}}}\right]_{+}}_{\widehat{R}_{\text{te-n}}^{+}(g)}\right]_{+}}_{\pi_{\text{te}}\widehat{R}_{\text{te-p}}^{+}(g)} + (1 - \pi_{\text{te}})\left[\underbrace{\frac{\widehat{R}_{\text{tr-u}}^{-}(g) - \pi_{\text{tr}}\widehat{R}_{\text{tr-p}}^{-}(g)}{1 - \pi_{\text{tr}}}}_{\widehat{R}_{\text{te-n}}^{-}(g)}\right]_{+}. \quad (25)$$

Like $\widehat{R}_{\text{PURR}}(g)$ from Section 6, $\widehat{R}_{\text{nn-PURR}}(g)$ is a consistent estimator.

When a risk estimator only has a single term that can be negative (like nnPU, $\widehat{R}_{\text{nn-wUU}}(g)$, and $\widehat{R}_{\text{nn-aPNU}}(g)$), the custom non-negativity ERM framework is relatively straightforward as shown in Algorithm 2. However, $\widehat{R}_{\text{nn-PURR}}(g)$ has three non-negativity corrections — one of which is nested inside another non-negativity correction.

Algorithm 3 details $\widehat{R}_{\text{nn-PURR}}(g)$'s custom ERM procedure with learning rate η. Each non-negativity correction is individually checked with the ordering critical. The optimizer minimizes risk on positive set $\mathcal{X}_{\text{p}}$ by both decreasing $\widehat{R}_{\text{p}}^{+}(g)$ and increasing $\widehat{R}_{\text{p}}^{-}(g)$. In contrast, each unlabeled example's minimizing risk is uncertain. This creates explicit tension and uncertainty for the optimizer. This enforced trade-off over the best unlabeled risk commonly delays or counteracts unlabeled set overfitting. As such, 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 its non-negativity is checked first (line 5). Nested term $\widehat{R}_{\text{te-n}}^{+}(g)$ receives second highest priority since whenever its value is implausible, 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 3 applies non-negativity correction by negating risk $\widehat{R}_{A}^{\hat{y}}(g)$'s gradient (see Eq. (10)). 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.

Algorithm 3 PURR with non-negativity correction custom ERM procedure

---

Input: Datasets $(\mathcal{X}_{\text{p}}, \mathcal{X}_{\text{tr-u}}, \mathcal{X}_{\text{te-u}})$ & hyperparameters (γ, η)
Output: Decision function g's parameters θ

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

D   Detailed Experimental Setup

This section details the experimental setup used to collect the results in Sections 7 and E.

D.1   Reproducing our Experiments

Our implementation is written and tested in Python 3.6.5 and 3.7.1 using the `PyTorch` [43] neural network framework versions 1.3.1 and 1.4. The source code is available at: https://github.com/ZaydH/arbitrary_pu. 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. Repository folder "`src/configs`" contains the configuration files for the experiments in Sections 7, E.1, and E.4. Prior probability shifts can be made by modifying the configuration files (see `yaml` fields `train_prior` and `test_prior`).

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 7.2's experimental setups are very similar to Hsieh et al. [11]'s experiments for positive, unlabeled, 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 suggests, 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 documents' 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.[7]

---

[7]We used the latest version of the 20 Newsgroups dataset with duplicates and cross-posts removed.

- **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 $4 * 0.1 = 0.4$.

For this experiment set, 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 TREC Spam Classification

As noted previously, PU learning has been applied to multiple adversarial domains including opinion spam [3, 16, 17, 18]. We use spam classification as a vehicle for testing our method in an adversarial domain.

Clearly, email spam classification is not a scenario where PU learning would generally be applied. Labeled data for both classes is generally plentiful (especially at the corporate level), and for most modern email systems, spam classification is a solved problem. For our purposes, spam email provides a good avenue for demonstrating our methods' performance in an adversarial setting for multiple reasons, including:

- The positive class (i.e., spam) evolves significantly faster than the negative class (i.e., not spam or "ham").

- Our fixed negative class-conditional distribution assumption (i.e., Eq. (7)) will not explicitly hold. This more closely represents what will be encountered "in-the-wild."

- Public spam/ham datasets exist eliminating the need to use our own proprietary adversarial learning dataset.

- Email dates provide a realistic criteria for partitioning the training and test datasets.

To be clear, what we propose here is not intended as a plausible, deployable spam classifier. Rather, we show that our methods apply to real-world adversarial domains.

Dataset Construction   The T̲ext R̲E̲trieval C̲onference (TREC) is organized annually be the United States' National Institute of Standards and Technology (NIST) to support information retrieval research [34]. In 2005, 2006, and 2007, TREC arranged annual spam classifier competitions where they released corpuses of spam and ham (i.e., not spam) emails.

As detailed in Table 5, the training set consisted of the TREC 2005 (TREC05) email dataset[8] while the test set was the TREC 2007 (TREC07) email dataset[9]. Basic statistics for the two datasets appear in Table 4.

The two sets of emails come from different domains. TREC05's ham emails derive largely from the Enron dataset. In contrast, TREC07's emails were received by a particular server between April and July 2007. Many of the ham emails were received by the University of Waterloo where the datasets were curated.

Due to the extended time required to encode all emails using the ELMo embedder (see Section D.7), we consider the first 10,000 emails from each dataset as defined by the dataset's `full/index` file.

Table 4: TREC05 & TREC07 dataset statistics

	TREC05	TREC07
Dataset Size	92,189	75,419
Approx. % Spam	~57%	~66%

D.2.3 Identical Positive Supports with Bias

Table 6 defines the positive and negative classes for the 10 LIBSVM datasets used in Section E.4. 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.

---

[8]The raw TREC05 emails can be downloaded from `https://plg.uwaterloo.ca/~gvcormac/treccorpus/`.

[9]The raw TREC07 emails can be downloaded from `https://plg.uwaterloo.ca/~gvcormac/treccorpus07/`.

Table 5: TREC spam email classification datasets

Class	Definition
Pos. Train	TREC05 Spam
Neg. Train	TREC05 Ham
Pos. Test	TREC07 Spam
Neg. Test	TREC07 Ham

Table 6: Positive & negative class definitions for the LIBSVM datasets in Section E.4

Dataset	d	Pos. Class	Neg. Class
banana	2	+1	2
cod-rna	8	+1	−1
susy	18	+1	0
ijcnn1	22	+1	−1
covtype.b	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 bPU Selection Bias Invariance of Order

Section E.5's experiments follow the invariance of order assumption as proposed and implemented by Kato et al. [13]. Their original experiments considered the MNIST dataset. For completeness, we expand our comparison to their method to also include the MNIST variants, FashionMNIST [44] and KMNIST [45]. Like MNIST, both FashionMNIST and KMNIST are multiclass datasets consisting of 10 disjoint labels. As described in Section D.2.1, binary classes are formed by partitioning the original set of labels.

As before, MNIST splits the labels between odds (positive class) and evens (negative class). For consistency, we used the same odd/even label partition for FashionMNIST and KMNIST. Note that such a partitioning lacks a corresponding semantic meaning for those two datasets.

D.4 Training, Validation, and Test Set Sizes

Table 7 lists the default size of each dataset's positive ($\mathcal{X}_p$), unlabeled train ($\mathcal{X}_{\text{tr-u}}$), unlabeled test ($\mathcal{X}_{\text{te-u}}$), and inductive test sets. All LIBSVM datasets (e.g., susy, a9a, etc. in Section E.4) used the dataset sizes defined by Sakai and Shimizu [14]. The separate validation set – made up of only positive and unlabeled examples – was one-fifth Table 7's training set sizes. Each learner observed identical dataset splits in each trial.

Special inductive test set sizes were needed for two of Section 7.2's disjoint positive-support experiments. To understand why, consider the MNIST disjoint-support experiment (i.e., the fourth MNIST row in Table 1) where the negative class (N) is comprised of labels $\{0, 2\}$ and the positive-test class (P_{test}) is composed of labels $\{5, 7\}$. 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 7). 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 8. To be clear, for all other datasets and experimental setups in Sections 7.2, 7.3, E.1, and E.4, Table 7 applies.

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 and Shimizu [14] 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. $\mathcal{X}_p$, $\mathcal{X}_{\text{tr-u}}$, $\mathcal{X}_{\text{te-u}}$ and the inductive test set are independently sampled at random from this monolithic set without replacement.

Table 7: Each dataset's default training set sizes. LIBSVM denotes all datasets downloaded directly from [30] and used in Section E.4. All quantities in the table 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	2,500	2,500	5,000
CIFAR10	1,000	5,000	5,000	3,000
TREC Spam	500	1,250	1,250	1,000
Synthetic	1,000	1,000	1,000	N/A
LIBSVM	250	583	583	2,000
FashionMNIST	833	$\leftarrow$ See Sec. E.5 $\rightarrow$		5,000
KMNIST	833	$\leftarrow$ See Sec. E.5 $\rightarrow$		5,000

Table 8: 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

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

D.5 CIFAR10 Image Representation

Each CIFAR10 [32] 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 [39]. The network's (linear) classification layer was removed, and the experiments used the 1,024-dimension feature vector output by DenseNet's convolutional backbone.

D.6 20 Newsgroups Document Representation

The 20 Newsgroups dataset is a collection of internet discussion board posts. The original dataset consisted of 20,000 documents [33]; it was pruned to 18,828 documents in 2007 after removal of duplicates and cross-posts [46]. 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. [37], 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. [11] who used this encoding scheme for positive, unlabeled, biased-negative (PUbN) (PUbN) learning, we used Rücklé et al. [38]'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 preceding 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.7 TREC Email Representation

The TREC05 and TREC07 emails are encoded using the ELMo embedder identical to 20 Newsgroups. See Section D.6 above for the details.

D.8 Models and Hyperparameters

This section reviews the experiments' hyperparameter methodology.

As specified by its authors, PUc's hyperparameters were tuned via importance-weighted cross validation (IWCV) [40]. PUc's author-supplied implementation includes a built-in hyperparameter tuning architecture that we used without modification.

Our hyperparameters and best-epoch weights were selected using the validation loss (using the associated risk estimation) on a validation set. 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 9 enumerates the general hyperparameter settings that applied to most/all learners. Batch sizes were selected based on the dataset sizes (see Tables 7 and 8) 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 $\{1, 2, 3\}$ for g and $\{0, 1, 2\}$ for $\hat{\sigma}$. With the exception of the output layer, each linear layer used ReLU activation and batch normalization [47]. The selected layer count minimized the median validation loss across all learners.

Tables 10, 11, and 12 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 $\lambda \in \{10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, 10^{-2}, 10^{-1}\}$, and (where applicable) gradient attenuator $\gamma \in \{0.1, 0.5, 1.0\}$[10].

By monitoring the (implausible) validation loss during Step #1, we observed overfitting when using the rich ELMo representations for the 20 Newsgroups and TREC email datasets. To address this, we added a dropout layer (with probability $p = 0.5$) before the input to each linear (i.e., fully-connected) layer. It is uncommon to use dropout even on the input dimension. However, we deliberately made this choice to still allow dropout even if we use a strictly linear-in-parameter model. Dropout was not used for any other dataset.

---

[10]Hyperparameter γ only applies when using Kiryo et al. [8]'s non-negativity correction. γ is not considered by our absolute-value correction.

Table 9: General hyperparameter settings

Dataset	#Epoch	Layer Count		Batch Size			Dropout?
		$g(x)$	$\hat{\sigma}(x)$	$g(x)$	$\hat{\sigma}(x)$	PN_{te}	
MNIST	200	3	1	5,000	5,000	4,000	
20 Newsgroups	200	1	1	5,000	2,500	2,000	✓
CIFAR10	200	2	1	10,000	2,500	1,500	
TREC Spam	200	1	0	1,000	1,000	1,000	✓
Synthetic	100	N/A	N/A	2,000	750	500	
banana	500	3	2	500	750	500	
cod-rna	500	2	1	500	750	500	
susy	500	2	2	500	750	500	
ijcnn1	500	2	2	500	750	500	
covtype.b	500	3	1	500	750	500	
phishing	500	2	2	500	750	500	
a9a	500	2	2	500	750	500	
connect4	500	2	1	500	750	500	
w8a	500	2	1	500	750	500	
epsilon	500	1	0	500	750	500	
FashionMNIST	200	3	1	⟵ See Section E.5 ⟶			
KMNIST	200	3	1	⟵ See Section E.5 ⟶			

Table 10: Dataset-specific hyperparameter settings for our aPU learners. Hyperparameter $\gamma*$ only applies when using Kiryo et al. [8]'s non-negativity correction instead of our absolute-value correction.

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

Table 11: Dataset-specific hyperparameter settings for nnPU.

Dataset	nnPU$_{\text{te} \cup \text{tr}}$			nnPU$_{\text{te}}$		
	η	λ	γ	η	λ	γ
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
TREC Spam	1E−3	1E−2	0.1	1E−3	1E−2	0.1
Synthetic	1E−2	0	1	1E−2	0	1
banana	1E−3	1E−3	1	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	1E−3	1E−3	0.5
connect4	1E−3	1E−3	0.1	1E−3	1E−4	1
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
FashionMNIST	1E−3	1E−3	0.5	1E−3	1E−3	0.5
KMNIST	1E−3	1E−3	0.5	1E−3	1E−3	0.5

Table 12: Dataset-specific hyperparameter settings for the positive-negative (PN) learners

Dataset	PN$_{\text{te}}$		PN$_{\text{tr}}$	
	η	λ	η	λ
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
TREC Spam	1E−3	1E−2	1E−3	1E−2
Synthetic	1E−2	0	1E−2	0
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
FashionMNIST	1E−3	1E−3	1E−3	1E−3
KMNIST	1E−3	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. With the exception of the synthetic data experiments (see Section E.1) which focus on visually illustrative examples to build intuitions, performance evaluation is based on the inductive misclassification rate since it approximates the expected zero-one loss for an unseen example.

E.1   Illustration using Synthetic Data

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

For simplicity, $\hat{\sigma}$ and g are linear-in-parameter models optimized by L-BFGS. PUc also trains a linear-in-parameter models without Gaussian kernels. Since all methods use the same classifier architecture, our methods' performance advantage comes solely from algorithmic design.

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 class-conditional distributions were

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

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

$\pi_{\text{te}} = \pi_{\text{tr}} = 0.5$ makes the ideal *test* decision boundary $x_1 = 0$. The datasets in Figure 3 vary only in the positive-train class-conditional distribution, denoted $p_{\text{tr-}(\cdot)\text{-p}}(x)$ where "$\cdot$" is subfigure a to c.

Figure 3a's positive-train class-conditional distribution is

$$p_{\text{tr-(a)-p}}(x) = \frac{1}{2}\mathcal{N}\left(\begin{bmatrix} 6 & -1 \end{bmatrix}, \mathbf{I}_2\right) + \frac{1}{2}\mathcal{N}\left(\begin{bmatrix} 6 & 1 \end{bmatrix}, \mathbf{I}_2\right), \tag{26}$$

making the training distribution's optimal separator linear. PUc performed poorly on this setup 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 3b adds to $p_{\text{tr-(a)-p}}(x)$ a third Gaussian where

$$p_{\text{tr-(b)-p}}(x) = \frac{2}{3}p_{\text{tr-(a)-p}}(x) + \frac{1}{3}\mathcal{N}\left(\begin{bmatrix} -6 & 0 \end{bmatrix}, \mathbf{I}_2\right), \tag{27}$$

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 3c uses the worst-case positive-train class-conditional, i.e., $p_{\text{tr-(c)-p}}(x) = p_{\text{n}}(x)$, making positive (labeled) data statistically identical to the (train and test) negative class-conditional 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.
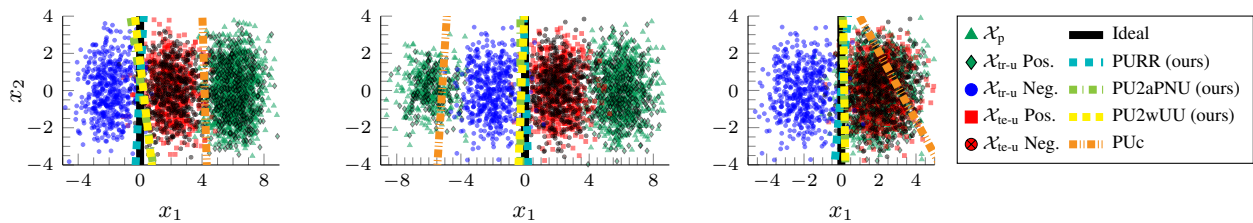
(a) Approx. linearly separable $\mathcal{X}_{\text{tr-u}}$ (b) Non-linearly separable $\mathcal{X}_{\text{tr-u}}$ (c) $p_{\text{tr-(c)-p}}(x) = p_{\text{n}}(x)$

Figure 3: Predicted linear decision boundaries for three synthetic datasets ($n_{\text{p}} = n_{\text{tr-u}} = n_{\text{te-u}} = 1{,}000$). Our three methods – PURR, PU2aPNU, and PU2wUU – are robust to non-linear & non-existent training class boundaries, but PUc fails in all three cases. Ideal boundary: $x_1 = 0$.

E.2 Expanded MNIST, 20 Newsgroups, and CIFAR10 Experiment Set

Table 13 is an expanded version of Section 7.2'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 (P_{train}) class configurations that are partially disjoint from the positive-test (P_{test}) class. For each such configuration, Table 13 contains three experiments (in order):

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

It is easier to directly compare the effects of increasing/decreasing π_{tr} when the magnitude of the training prior increase and decrease are equivalent (e.g., for MNIST $\pi_{te} = 0.5$ so we tested performance at $\pi_{tr} = \pi_{te} \pm 0.12$ and $\pi_{tr} = \pi_{te} \pm 0.21$ depending on the class partition). 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 13 is that across these additional, orthogonal definitions of P_{train}, our methods still outperform PUc and nnPU* — usually by a wide margin (statistical significance according to 1% paired t-test).

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

PUc performs best when $\pi_{tr} = \pi_{te}$. When π_{tr} diverges from that middle point, PUc's performance declines. To gain an intuition why that is, consider density-ratio estimation in terms of the component class conditionals. When $\pi_{tr} = \pi_{te}$, $w(x) = 1$ for all negative examples; from Table 13'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.

Table 13: Full MNIST, 20 Newsgroups, and CIFAR10 experimental class partition results. Each result is the inductive misclassification rate (%) mean and standard deviation over 100 trials for MNIST, 20 Newsgroups, and CIFAR10 with different positive & negative class definitions. For *all* experiments with positive bias (i.e., rows 2–8 for each dataset), all three of our methods had statistically significant better performance than PUc and nnPU* according to a 1% 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. Our three methods – PURR, PU2aPNU, and PU2wUU – are denoted with [†].

| | N | P_{test} | P_{train} | π_{tr} | π_{te} | PURR[†] | Two-Step (PU2) | | Baselines | | Ref. |
							aPNU[†]	wUU[†]	PUc	nnPU*	PN_{te}
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	P_{test}	0.5	0.5	10.0 (1.3)	10.0 (1.2)	11.6 (1.6)	8.6 (0.8)	5.5 (0.5)	↑
				0.29	0.5	6.8 (0.8)	**5.3 (0.6)**	6.0 (0.7)	29.2 (2.1)	36.7 (2.7)	
			7, 9	0.5	0.5	9.4 (1.5)	**7.1 (0.9)**	8.3 (1.5)	26.8 (2.4)	35.1 (2.5)	
				0.71	0.5	14.0 (3.0)	**11.1 (1.4)**	14.8 (3.1)	26.9 (3.0)	34.5 (2.9)	2.8 (0.2)
				0.38	0.5	8.1 (1.0)	**6.5 (0.8)**	7.6 (0.9)	20.2 (2.5)	25.9 (1.1)	
			1, 3, 5	0.5	0.5	10.0 (1.6)	**8.4 (1.1)**	10.2 (1.4)	18.5 (2.9)	26.9 (1.2)	
				0.63	0.5	12.5 (2.3)	**11.4 (1.3)**	14.3 (2.3)	18.6 (3.3)	28.5 (1.2)	↓
	0, 2	5, 7	1, 3	0.5	0.5	4.0 (0.8)	3.6 (0.9)	**3.1 (0.7)**	17.1 (4.6)	30.9 (5.3)	1.1 (0.2)
20 Newsgroups	sci, soc, talk	alt, comp, misc, rec	P_{test}	0.56	0.56	15.4 (1.3)	14.9 (1.0)	16.7 (2.3)	14.9 (1.0)	14.1 (0.8)	↑
				0.37	0.56	13.9 (0.7)	**12.8 (0.6)**	14.3 (0.9)	28.9 (1.8)	28.8 (1.3)	
			misc, rec	0.56	0.56	17.5 (2.1)	**13.5 (0.8)**	15.1 (1.3)	23.9 (3.0)	28.8 (1.7)	
				0.65	0.56	20.2 (2.8)	**14.0 (0.9)**	15.9 (1.5)	21.8 (3.3)	29.0 (1.8)	10.5 (0.5)
				0.37	0.56	**13.3 (0.6)**	13.7 (0.6)	14.4 (0.7)	30.3 (2.0)	31.4 (0.7)	
			comp	0.56	0.56	16.0 (1.5)	**14.9 (0.7)**	15.7 (0.9)	28.6 (2.6)	31.2 (0.8)	
				0.65	0.56	19.2 (2.4)	**15.6 (0.9)**	16.5 (1.2)	27.8 (2.7)	31.3 (0.7)	↓
	misc, rec	soc, talk	alt, comp	0.55	0.46	5.9 (1.0)	7.1 (1.1)	**5.6 (1.7)**	18.5 (4.3)	35.3 (5.2)	2.1 (0.3)
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	P_{test}	0.4	0.4	14.1 (0.8)	14.2 (1.3)	15.4 (1.7)	13.8 (0.7)	12.3 (0.6)	↑
				0.14	0.4	12.1 (0.7)	**11.9 (0.7)**	12.4 (0.9)	26.7 (1.4)	26.7 (1.0)	
			Plane	0.4	0.4	**13.8 (0.9)**	14.5 (1.4)	15.1 (1.6)	20.6 (1.5)	27.4 (1.0)	
				0.6	0.4	**16.1 (1.1)**	16.7 (1.5)	20.0 (2.7)	21.5 (1.6)	28.4 (1.0)	9.7 (0.5)
				0.25	0.4	12.7 (0.7)	**12.4 (0.7)**	12.8 (0.8)	19.2 (1.1)	20.3 (0.8)	
			Auto, Truck	0.4	0.4	14.1 (0.9)	**13.9 (1.1)**	14.4 (1.2)	17.7 (1.0)	20.3 (0.8)	
				0.55	0.4	**16.0 (1.1)**	16.2 (1.6)	17.1 (2.2)	18.3 (1.1)	20.5 (0.9)	↓
	Deer, Horse	Plane, Auto	Cat, Dog	0.5	0.5	14.1 (0.9)	14.9 (1.5)	**11.2 (0.8)**	33.1 (2.7)	47.5 (2.0)	7.7 (0.4)

E.3 Case Study: Arbitrary Adversarial Concept Drift

This section's experiments model adversarial settings where the positive class-conditional distribution shifts significantly faster than the negative class distribution. As explained in Section D.2.2, the training set was composed of spam and ham emails from the TREC0$\underline{5}$ dataset; the test set was composed of spam and ham emails from the TREC0$\underline{7}$ dataset. The two dataset's ham emails are quite different – TREC05 relies heavily on Enron emails while TREC07 contains many emails received on a university email server. We are therefore confident our fixed-negative-distribution assumption in Eq. (7) does not hold.

Table 14 and Figure 4 compare our methods to PUc and nnPU across three different training priors (π_{tr}). Under all three experimental conditions, our three methods outperformed both PUc and nnPU* according to a 1% paired t-test. PU2wUU was the top performer for all experiments. As evidenced by the PN misclassification rate, a highly accurate classifier can be constructed for this dataset. Similarly, $\hat{\sigma}$ accurately labels $\mathcal{X}_{tr-u}$. The resulting surrogate negative set is more useful than $\mathcal{X}_p$ to classify the spam emails from the test distribution. PU2aPNU performed slightly worse than PU2wUU because the spam emails in $\mathcal{X}_p$ are of very limited value due to the significant adversarial concept drift.[11]

Table 14: Inductive misclassification rate (%) mean and standard deviation over 100 trials for arbitrary adversarial concept drift on the TREC spam email datasets. In all experiments, our three methods – PURR, PU2aPNU, & PU2wUU – (which are denoted by [†]) statistically outperformed PUc and nnPU* according to a paired t-test ($p < 0.01$) with PU2wUU the top performer across all training priors (π_{tr}).

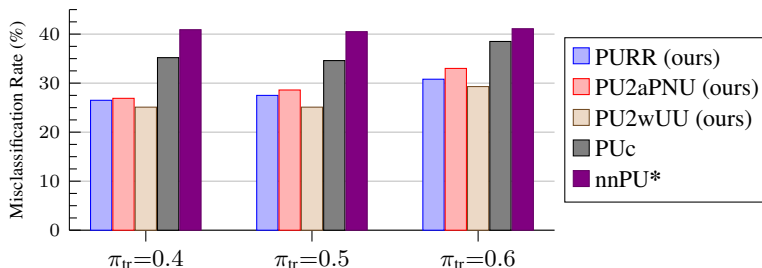| Train		Test		π_{tr}	π_{te}		Two-Step (PU2)		Baselines		Ref.
Pos.	Neg.	Pos.	Neg.			PURR[†]	aPNU[†]	wUU[†]	PUc	nnPU*	PN$_{te}$
2005 Spam	2005 Ham	2007 Spam	2007 Ham	0.4	0.5	26.5 (2.6)	26.9 (3.1)	**25.1 (3.1)**	35.2 (11.3)	40.9 (3.1)	↑
				0.5	0.5	27.5 (3.4)	28.6 (4.5)	**25.1 (3.3)**	34.6 (10.2)	40.5 (2.7)	0.6 (0.3)
				0.6	0.5	30.8 (4.2)	33.0 (5.7)	**29.3 (6.5)**	38.5 (10.8)	41.1 (2.9)	↓

Figure 4: Mean inductive misclassification rate (%) over 100 trials for the TREC spam datasets across three training priors (π_{tr}). Our PU2wUU method was the top performer across all experiments.

---

[11]PU2wUU and PU2aPNU used top-k weighting (see Section E.7.1) for step #1.

E.4 Identical Positive Supports with Bias

The positive bias applied in this section's experiments is totally different from that in Sections 7.2 and 7.3. Here we mimic situations where the labeled data are complete but non-representative resulting in identical marginal distribution supports but shifts in the marginal distribution's magnitude. We follow the experimental setup described in Sakai and Shimizu [14]'s PUc paper. LIBSVM [30] benchmarks are used exclusively to ensure suitability with the SVM-like PUc; benchmarks "banana," "susy," "ijcnn1," and "a9a" appear in Sakai and Shimizu [14]'s PUc paper.

Sakai and 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 $\{\|x - \bar{x}\|_2 : x \in \mathcal{X}\}$ where $\|\cdot\|_2$ is the L_2 (Euclidean) norm and $\bar{x}$ is $\mathcal{X}$'s mean vector, i.e.,

$$\bar{x} = \frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} x.$$

Partition $\mathcal{X}$ into subsets $\mathcal{X}_{\mathrm{lo}} := \{x \in \mathcal{X} : \|x - \bar{x}\|_2 < c_{\mathrm{med}}\}$ and $\mathcal{X}_{\mathrm{hi}} := \mathcal{X} \setminus \mathcal{X}_{\mathrm{lo}}$. Examples in $\mathcal{X}_{\mathrm{p}}$ and $\mathcal{X}_{\mathrm{tr\text{-}u}}$ are selected from $\mathcal{X}_{\mathrm{lo}}$ with probability $p = 0.9$ and from $\mathcal{X}_{\mathrm{hi}}$ with probability $1 - p$. $p = 0.1$ is used when constructing $\mathcal{X}_{\mathrm{te\text{-}u}}$ and the test set. This bias operation simplifies density-ratio estimation since $\forall_{x \in \mathcal{X}} \ w(x) \in \{\frac{1}{9}, 9\}$. Their setting $\pi_{\mathrm{tr}} = \pi_{\mathrm{te}} = 0.5$ also simplifies density estimation as detailed in Section E.2.

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

Analysis The experiments enumerated in Table 15 and shown visually in Figure 5 used the bias procedure described above on 10 LIBSVM datasets. According to a 1% paired t-test, PURR and PU2aPNU outperformed the baselines, PUc and nnPU*, on all ten benchmarks; PU2wUU outperformed the baselines on nine of ten benchmarks.

PURR was the top performer on three benchmarks; PU2aPNU was the top performer on five benchmarks while PU2wUU was the top performer on two benchmarks. Each estimator is best suited to a different feature dimension range. PURR performed best when the dataset had fewer features (e.g., <50) while PU2aPNU performed well when the dimension was moderate. PU2wUU was the top performer when the dimension was large (e.g., ≥300).

Accurate risk estimation is more challenging when the training sets are comparatively small but the feature count is high. We expect that is causing PURR to struggle to reconcile/relate the different labeled losses (e.g., positive-labeled, unlabeled train, unlabeled test) in these higher dimension datasets.

Table 15: Inductive misclassification rate (%) mean and standard deviation over 100 trials with Sakai and Shimizu [14]'s median feature vector-based bias for 10 LIBSVM datasets. Underlining denotes a statistically significant performance improvement versus PUc and nnPU* according to a 1% paired t-test. Boldface indicates each dataset's best performing method. $n_p = 300$ and $n_{tr\text{-}u} = n_{te\text{-}u} = 700$. Datasets are ordered by increasing dimension. Our three methods – PURR, PU2aPNU, and PU2wUU – are denoted with †.

| Dataset	d		Two-Step (PU2)		Baselines		Ref.
		PURR†	aPNU†	wUU†	PUc	nnPU*	PN$_{te}$
banana	2	12.9 (2.1)	**11.8 (1.6)**	13.3 (2.3)	17.4 (3.4)	28.8 (3.8)	8.6 (0.6)
cod-rna	8	**14.7 (2.6)**	15.1 (3.2)	15.5 (2.9)	25.2 (5.0)	24.9 (2.3)	6.5 (0.9)
susy	18	**24.2 (2.1)**	25.6 (2.2)	25.8 (2.2)	27.3 (4.3)	45.9 (3.9)	20.5 (1.3)
ijcnn1	22	22.7 (2.8)	**17.7 (2.8)**	24.6 (3.1)	23.9 (3.6)	34.7 (3.6)	6.8 (0.8)
covtype.b	54	**29.5 (2.9)**	32.5 (3.2)	29.9 (2.4)	39.4 (4.2)	55.5 (2.8)	22.3 (1.4)
phishing	68	11.3 (1.4)	**9.6 (1.0)**	11.1 (1.8)	13.8 (4.1)	22.5 (4.1)	6.2 (0.6)
a9a	123	27.1 (2.1)	**26.6 (1.8)**	27.1 (2.1)	32.8 (2.6)	32.5 (2.3)	20.6 (1.0)
connect4	126	34.9 (3.1)	**32.9 (2.7)**	35.0 (2.9)	37.0 (2.8)	45.1 (2.6)	21.6 (1.3)
w8a	300	17.2 (2.6)	21.0 (2.9)	**16.8 (2.9)**	29.3 (6.2)	41.1 (4.3)	6.6 (0.7)
epsilon	2,000	33.5 (4.8)	36.5 (5.0)	**31.5 (1.7)**	62.8 (6.7)	64.6 (1.5)	23.7 (1.1)
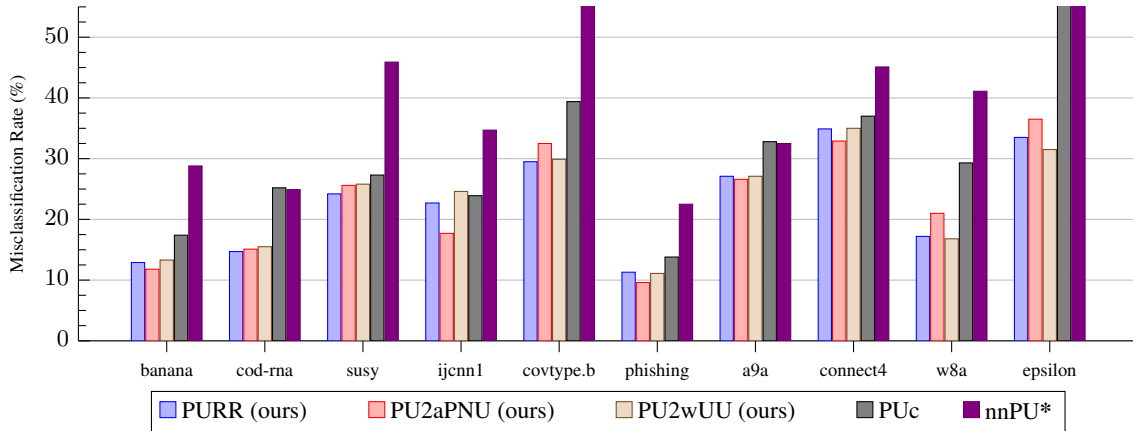
Figure 5: Mean inductive misclassification rate (%) over 100 trials with Sakai and Shimizu [14]'s median feature vector-based bias for the 10 LIBSVM datasets in Section E.4.

E.5 Comparison to bPU Selection Bias Method PUSB

Recall that baseline PUc is a covariate-shift bPU method. A NeurIPS reviewer requested an experiment comparing our proposed approaches to a selection bias bPU learning baseline. This section compares our algorithms to Kato et al. [13]'s Positive-Unlabeled Selection Bias (PUSB) method.

Let random variable $S \in \{\pm 1\}$ denote whether some training example $(X, Y) \sim p(x, y)$ is labeled. For all types of PU learning (e.g., unbiased, bPU, aPU), it is straightforward that $S = +1$ implies $Y = +1$, i.e.,

$$p(Y = +1 | S = +1) = 1 \tag{28}$$

and

$$p(S = +1 | Y = -1) = 0. \tag{29}$$

PUSB makes what Kato et al. term the *invariance-of-order assumption*. Formally, for any pair of training examples $x_i, x_j \in \mathbb{R}^d$, it holds that

$$p(Y = +1 | x_i) \geq p(Y = +1 | x_j) \iff p(S = +1 | x_i) \geq p(S = +1 | x_j). \tag{30}$$

In words, a training example is at least as likely to be positive-valued as another example if and only if it is at least as likely to be labeled as that other example. As mentioned in Section 4, it is not possible to directly compare our approaches to existing selection bias bPU methods like PUSB. Such bPU learning methods assume access to only a single unlabeled set ($\mathcal{X}_{\text{te-u}}$) drawn from the test distribution while aPU learning provides two unlabeled sets ($\mathcal{X}_{\text{tr-u}}$ and $\mathcal{X}_{\text{te-u}}$).

To ensure a fair comparison, we sought to replicate Kato et al. [13]'s experimental setup as closely as possible provided the constraints of our method – even using their source code[12] verbatim where possible (e.g., PUSB used the `Chainer` [48] neural network framework as specified by Kato et al.). Like in the PUSB paper, we analyzed the performance of all methods on the MNIST [31] dataset. To enrich the comparison, we also consider the drop-in MNIST variants FashionMNIST [44] and KMNIST [45].

Dataset Construction Our experiments exactly duplicate Kato et al.'s procedure for constructing biased-positive set $\mathcal{X}_{\text{p}}$. Specifically, a multilayer perceptron (MLP) with four hidden layers of 300 neurons each and ReLU activation is trained using the PN logistic loss on the dataset's complete training and test sets. $\mathcal{X}_{\text{p}}$ is then selected u.a.r. without replacement from those positive-valued training examples the aforementioned MLP identifies as having the highest positive posterior. Unlabeled test set $\mathcal{X}_{\text{te-u}}$ and the inductive test set are drawn u.a.r. without replacement from the complete training and test sets respectively.

Kato et al. uses the complete MNIST training set as the unlabeled set. Since both PUc and our methods require two unlabeled sets, we cannot follow the same methodology here. Instead, we limit the size of the test unlabeled set and create unlabeled training set $\mathcal{X}_{\text{tr-u}}$ by selecting its positive examples according to the procedure described above for $\mathcal{X}_{\text{p}}$ and selecting its negative-valued examples u.a.r. without replacement from the training set's negative elements. Table 16 details our experiments' positive priors as well as the dataset and mini-batch sizes.

Hyperparameters Identical hyperparameters were used for the MNIST, FashionMNIST, and KMNIST datasets. Our methods, nnPU*, and PN$_{\text{te}}$ used identical hyperparameter settings as those tuned for MNIST in Section 7's experiments.

PUSB's hyperparameters match those specified by Kato et al. for MNIST, e.g., learning rate $\eta = 10^{-5}$ and weight decay $\lambda = 5 \cdot 10^{-3}$. PUSB learners were trained for 250 epochs using the Adam [49] optimizer. As in the original paper, PUSB's neural network had four hidden layers of 300 neurons each and batch normalization before each ReLU activation.

Results Analysis Table 17 compares the performance of our methods – PURR, PU2aPNU, and PU2wUU – to the extended baseline set – PUc, PUSB, and nnPU* – for the experimental setup described above. To mitigate the effects of different unlabeled set configurations, our experiments tested two unlabeled set sizes, with one size half the other (Table 16). PUc's and our methods' results in Table 17a used 6,000 total unlabeled samples, i.e., the same quantity used by PUSB in Table 17b. Figure 6 visualizes these cross-table, matching-unlabeled-set-size results graphically. For nnPU* in Figure 6, three unlabeled set configurations are considered namely, nnPU$_{\text{te}}$ and nnPU$_{\text{te}\cup\text{tr}}$ with $|\mathcal{X}_{\text{tr-u}}| = |\mathcal{X}_{\text{te-u}}| = 3,000$ as well as nnPU$_{\text{te}}$ with $|\mathcal{X}_{\text{te-u}}| = 6,000$. Observe that these are the only nnPU* configurations using at most 6,000 unlabeled examples.

---

[12]Kato et al.'s source code is publicly available at `https://github.com/MasaKat0/PUlearning`.

As mentioned in Section 7.2, when there is little to no dataset shift, shift-unaware methods (e.g., nnPU) are expected to be the top performer. As an intuition why – when a method searches for a non-existent phenomenon, any patterns found will not generalize. Since nnPU* is the top performer for MNIST and KMNIST despite not accounting for shift at all, it then stands to reason that Kato et al. [13]'s invariance-of-order bias induces only a small shift here.

We saw in Section 7.2 that for such mild shifts (e.g., no bias), PUc often outperforms our methods. We generally see the same trend in Table 17 for MNIST and KMNIST (primary exception being PU2aPNU for MNIST). This is again expected. Under mild shifts, covariate shift's consistent input-output relation assumption generally holds. In addition, importance function $w(x) \approx 1$ for all x under limited bias, in which case PUc simplifies to essentially standard nnPU.

All of our methods outperformed all baselines for FashionMNIST. What is more, our methods outperformed PUSB in all but one case (PU2wUU for KMNIST) even after accounting for unlabeled set size (Figure 6). In fact, PUSB always lagged nnPU*. This hints at a level of brittleness for Kato et al.'s method since PUSB struggled on a bias condition it specifically targets.

Table 16: Positive priors, dataset sizes (including the validation set), and mini-batch sizes for Section E.5's invariance of order selection bias experiments. The first column lists the table where each setup's corresponding results are enumerated.

	Prior		Dataset Size				Batch Size			
	π_{tr}	π_{te}	n_{p}	$n_{\text{tr-u}}$	$n_{\text{te-u}}$	n_{Test}	$g(x)$	$\hat{\sigma}(x)$	PUSB	PN_{te}
Table 17a	0.5	0.5	1,000	3,000	3,000	5,000	2,500	2,500	1,000	2,000
Table 17b	0.5	0.5	1,000	6,000	6,000	5,000	5,000	5,000	1,000	4,000

Table 17: Inductive misclassification rate (%) mean and standard deviation over 100 trials for the experiments using Kato et al. [13]'s invariance-of-order setup on the MNIST, FashionMNIST, and KMNIST datasets. Bold face denotes each dataset's best performing method according to mean misclassification rate. Our methods – PURR, PU2aPNU, and PU2wUU – are denoted with [†].

(a) $|\mathcal{X}_{\text{tr-u}}| = |\mathcal{X}_{\text{te-u}}| = 3,000$

Dataset		Two-Step (PU2)		Baselines			Ref.
	PURR[†]	aPNU[†]	wUU[†]	PUc	PUSB	nnPU*	PN_{te}
MNIST	13.0 (2.3)	9.7 (1.3)	11.6 (1.4)	10.6 (1.1)	15.9 (1.0)	**8.8 (0.9)**	3.6 (0.3)
FashionMNIST	6.4 (1.4)	**5.3 (0.7)**	5.9 (1.0)	9.0 (1.1)	10.5 (1.2)	8.5 (1.3)	3.5 (0.3)
KMNIST	31.6 (2.4)	29.7 (2.2)	33.7 (2.3)	27.3 (1.4)	33.4 (1.2)	**24.6 (1.4)**	16.4 (0.8)

(b) $|\mathcal{X}_{\text{tr-u}}| = |\mathcal{X}_{\text{te-u}}| = 6,000$

Dataset		Two-Step (PU2)		Baselines			Ref.
	PURR[†]	aPNU[†]	wUU[†]	PUc	PUSB	nnPU*	PN_{te}
MNIST	10.5 (1.8)	8.5 (1.2)	9.3 (1.0)	10.2 (1.1)	14.2 (1.0)	**8.0 (0.9)**	2.8 (0.2)
FashionMNIST	5.6 (1.3)	**4.8 (0.6)**	5.0 (0.8)	9.1 (1.2)	10.0 (1.2)	8.2 (1.2)	3.1 (0.2)
KMNIST	29.6 (2.2)	29.3 (2.1)	32.0 (2.2)	27.0 (1.4)	32.1 (1.2)	**24.1 (1.4)**	13.7 (0.7)

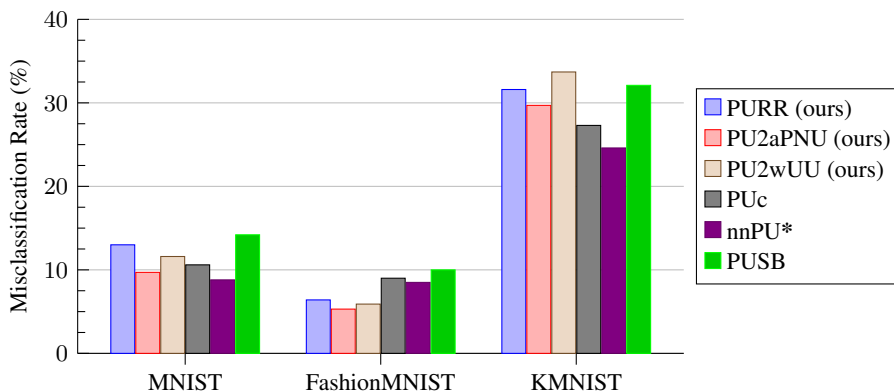Figure 6: Mean inductive misclassification rate (%) over 100 trials for the experiments using Kato et al. [13]'s invariance-of-order setup on the MNIST, FashionMNIST, and KMNIST datasets. All learners saw up to 6,000 total unlabeled examples with results cross-compiled between Table 17a (for our methods and PUc) and Table 17b (for PUSB). Here nnPU* considers three different unlabeled set configurations as described in Section E.5.

E.6 Empirical Comparison of Absolute-Value and Non-Negativity Corrections

Section 3 describes our streamlined absolute-value correction to address PU learning overfitting. This section compares our simpler absolute-value correction to Kiryo et al. [8]'s non-negativity correction using max and "defitting."

E.6.1 Ordinary Positive-Unlabeled Learning Performance Without Distributional Shift

We first consider a direct comparison of nnPU and abs-PU on *unshifted* data. $\mathcal{X}_p$ and $\mathcal{X}_u$ are constructed identically to the procedure used to construct the positive-labeled and unlabeled-train datasets in our aPU learning experiments. Unlike before, the inductive test set is now drawn from the *training* distribution. We then trained classifiers using nnPU and abs-PU with the sigmoid loss. In all experiments, the classifiers had identical initial weights and were trained on identical dataset splits.

Hyperparameters (including γ) were tuned using nnPU; these identical hyperparameters were then used for abs-PU (i.e., not in any way tuned for absolute-value correction). Therefore, the results represent the *performance floor* when transitioning from nnPU to abs-PU. This was done due to time constraints.

Table 18 compares abs-PU and nnPU for the datasets in Sections 7.2[13], 7.3, and E.4. We also report the difference between abs-PU and nnPU with a positive number indicating that abs-PU performed better that nnPU.

abs-PU was the top performer on eight of fourteen benchmarks and tied with nnPU on two others; the results are generally too close to be statistically significant. Both methods had comparable variances. In summary, abs-PU is both simpler and saw similar or slightly better performance than nnPU on unbiased data, even under conditions (i.e., hyperparameters) that favor nnPU.

Table 18: Comparison of inductive misclassification rate (%) mean and standard deviation over 100 trials for abs-PU and nnPU on unshifted data. Boldface denotes the best performing algorithm according to mean misclassification rate. For the difference (Diff.) column, a positive value denotes that abs-PU outperformed nnPU.

Dataset	abs-PU	nnPU	nnPU – abs-PU (Diff.)
MNIST	6.6 (0.7)	**6.5 (0.7)**	–0.1 (0)
20 Newsgroups	**13.3 (1.3)**	13.5 (1.2)	0.2 (–0.1)
CIFAR10	**12.4 (0.7)**	**12.4 (0.7)**	0 (0)
TREC Spam	**2.0 (1.0)**	2.1 (0.9)	0.1 (–0.1)
banana	**10.5 (1.0)**	**10.5 (1.1)**	0 (0.1)
cod-rna	**10.3 (1.8)**	10.4 (2.0)	0.1 (0.2)
susy	28.8 (1.7)	**28.7 (1.8)**	–0.1 (0.1)
ijcnn1	**10.1 (1.4)**	10.2 (1.5)	0.1 (0.1)
covtype.b	**32.8 (2.2)**	33.3 (2.1)	0.5 (–0.1)
phishing	8.6 (1.3)	**8.5 (1.2)**	–0.1 (–0.1)
a9a	**15.9 (1.1)**	16.0 (1.2)	0.1 (0.1)
connect4	24.6 (2.2)	**24.4 (2.0)**	–0.2 (–0.2)
w8a	**17.8 (1.6)**	17.9 (1.6)	0.1 (0)
epsilon	**31.1 (1.4)**	31.2 (1.7)	0.1 (0.3)

E.6.2 Ordinary Positive-Unlabeled Learning Performance Under Distribution Shift

The previous section compared the performance of nnPU and abs-PU under ideal conditions, i.e., no positive shift. This section compares nnPU and abs-PU *with positive shift*, specifically under the aPU learning conditions we use in our experimental evaluation.

Like in the previous section, all classifiers in each experimental trial had identical initial weights and saw identical dataset splits. Hyperparameters (including γ) were tuned using nnPU; these identical hyperparameters were then used for abs-PU (i.e., not in any way tuned for absolute-value correction). Therefore, the results again represent the *performance floor* if transitioning from nnPU to abs-PU. This choice was made due to limited time.

Recall from Section 7 that evaluation baseline nnPU* considers two nnPU-based classifiers – one trained with unlabeled set $\mathcal{X}_{te\text{-}u}$ and the other trained with unlabeled set $\mathcal{X}_{tr\text{-}u} \cup \mathcal{X}_{te\text{-}u}$ (using the true composite prior), and we report whichever of those two classifiers performed best on average. In this section, we introduce abs-PU*, which like nnPU*, considers

---

[13]The test conditions for MNIST, 20 Newsgroups, and CIFAR10 correspond to the unbiased test conditions (i.e., row 1 for each dataset where $P_{train} = P_{test}$) in Table 1/Table 13.

two classifiers separately trained with the different unlabeled set configurations: $\mathcal{X}_{\text{te-u}}$ and $\mathcal{X}_{\text{tr-u}} \cup \mathcal{X}_{\text{te-u}}$. The only difference is that abs-PU*, as its name would suggest, uses our abs-PU risk estimator. We specifically separated this section to delineate the baseline performance of our contribution (abs-PU) versus existing methods (nnPU).

Table 19 compares abs-PU* and nnPU* for the extended set of experiments in Table 13 (see Section E.2). Recall that those experiments tested cases where some positive subclasses exist only in the test distribution. Similar to Table 18, a positive value in the column labeled "Diff." denotes that abs-PU* performed better than nnPU*.

For multiple positive-train (P_{train}) class configurations (e.g., MNIST $P_{\text{train}}= \{1, 3, 5\}$), abs-PU* and nnPU* exhibited similar performance. When there was a large difference between the two methods (e.g., 20 Newsgroups $P_{\text{train}}= \{\text{misc}, \text{rec}\}$), abs-PU* had significantly better mean accuracy – reducing the misclassification rate by multiple percentage points. The difference between the methods was most pronounced when P_{train} and P_{test} are disjoint.

These results indicate that in some cases, abs-PU* is learning decision boundaries that better generalize to *unseen types of data*. To be clear, this does not apply to all datasets (CIFAR10 exhibited little difference between the methods except when the positive supports were disjoint) nor even to all class partitions within a dataset (see MNIST positive-train classes $\{7, 9\}$ versus $\{1, 3, 5\}$). It should also be noted that missing positive subclasses is a more extreme form of positive shift. The next set of results considers the more mild case of marginal-distribution magnitude shifts.

Table 19: Comparison of inductive misclassification rate (%) mean and standard deviation over 100 trials for abs-PU* and nnPU* for the experimental shift tasks (eight per dataset) in Table 13 with partially/fully disjoint positive class supports. Boldface denotes the best performing task according to mean misclassification rate. For the difference column, a positive value indicates abs-PU* outperformed nnPU*.

	N	P_{test}	P_{train}	π_{tr}	π_{te}	abs-PU*	nnPU*	Diff.
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	7, 9	0.29	0.5	**34.4 (2.6)**	36.7 (2.7)	2.3 (0.1)
				0.5	0.5	**33.1 (2.3)**	35.1 (2.5)	2.0 (0.2)
				0.71	0.5	**32.7 (2.2)**	34.5 (2.9)	1.8 (0.7)
			1, 3, 5	0.38	0.5	**25.9 (1.2)**	**25.9 (1.1)**	0 (–0.1)
				0.5	0.5	27.1 (1.3)	**26.9 (1.2)**	–0.2 (–0.1)
				0.63	0.5	28.7 (1.1)	**28.5 (1.2)**	–0.2 (0.1)
	0, 2	5, 7	1, 3	0.5	0.5	**25.7 (6.9)**	30.9 (5.3)	5.2 (–1.6)
20 Newsgroups	sci, soc, talk	alt, comp, misc, rec	misc, rec	0.37	0.56	**27.0 (1.9)**	28.8 (1.3)	1.8 (–0.6)
				0.56	0.56	**26.0 (1.7)**	28.8 (1.7)	2.8 (0)
				0.65	0.56	**25.9 (1.7)**	29.0 (1.8)	3.1 (0.1)
			comp	0.37	0.56	**31.2 (0.7)**	31.4 (0.7)	0.2 (0)
				0.56	0.56	**31.0 (0.9)**	31.2 (0.8)	0.2 (–0.1)
				0.65	0.56	**31.0 (0.8)**	31.3 (0.7)	0.3 (–0.1)
	misc, rec	soc, talk	alt, comp	0.55	0.46	**34.6 (5.0)**	35.3 (5.2)	0.7 (0.2)
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	Plane	0.14	0.4	**26.5 (1.0)**	26.7 (1.0)	0.2 (0)
				0.4	0.4	**27.4 (1.0)**	**27.4 (1.0)**	0 (0)
				0.6	0.4	**28.3 (1.1)**	28.4 (1.0)	0.1 (–0.1)
			Auto, Truck	0.25	0.4	**20.3 (0.8)**	**20.3 (0.8)**	0 (0)
				0.4	0.4	20.4 (0.9)	**20.3 (0.8)**	–0.1 (–0.1)
				0.55	0.4	20.9 (0.9)	**20.5 (0.9)**	–0.4 (0)
	Deer, Horse	Plane, Auto	Cat, Dog	0.5	0.5	**44.6 (1.8)**	47.5 (2.0)	2.9 (0.2)

Table 20 compares abs-PU* and nnPU* for the 10 LIBSVM datasets in Table 15 (see Section E.4). Recall that in these experiments, the positive-train and positive-test class-conditionals have identical supports. For seven of ten benchmarks, abs-PU* had better mean performance than nnPU* and had equivalent performance on one other benchmark. abs-PU* did have generally higher result variance. For some benchmarks (e.g., `ijcnn1`, `covtype.b`, `epsilon`, etc.), the change in variance was more than offset by the improvement in mean accuracy. Had the abs-PU* learning rates been tuned directly instead of using nnPU*'s hyperparameter settings, we expect this variance difference would have been mitigated. Again however, limited time prevented that experiment.

In summary, abs-PU*'s performance is comparable or slightly/significantly better than that of nnPU* under aPU learning conditions that are deleterious to ordinary PU risk estimators but that may be more realistic to real-world data.

Table 20: Comparison of inductive misclassification rate (%) mean and standard deviation over 100 trials for abs-PU*
and nnPU* for the 10 LIBSVM datasets in Table 15 under Sakai and Shimizu [14]'s mean feature vector bias. Boldface
denotes the best performing task according to mean misclassification rate. For the difference column, a positive value
indicates abs-PU* outperformed nnPU*.

Dataset	d	abs-PU*	nnPU*	Diff.
banana	2	**28.5 (4.1)**	28.8 (3.8)	0.3 (–0.3)
cod-rna	8	25.1 (2.5)	**24.9 (2.3)**	–0.2 (–0.2)
susy	18	**45.9 (3.9)**	**45.9 (3.9)**	0 (0)
ijcnn1	22	**33.3 (3.9)**	34.7 (3.6)	1.4 (–0.3)
covtype.b	54	**54.6 (3.1)**	55.5 (2.8)	0.9 (–0.3)
phishing	68	22.9 (4.2)	**22.5 (4.1)**	–0.4 (–0.1)
a9a	123	**32.0 (2.5)**	32.5 (2.3)	0.5 (–0.2)
connect4	126	**44.9 (3.1)**	45.1 (2.6)	0.2 (–0.5)
w8a	300	**40.0 (4.0)**	41.1 (4.3)	1.1 (0.3)
epsilon	2,000	**64.1 (1.4)**	64.6 (1.5)	0.5 (0.1)

E.6.3 Effect of Absolute-Value Correction on Our aPU Learning Methods

This section examines the effect of using absolute-value correction over non-negativity correction for our three
aPU learning methods – PURR, PU2aPNU, and PU2wUU. Recall that non-negativity correction requires custom
ERM algorithms to support "defitting." Section C describes our methods' custom ERM frameworks when using
non-negativity.

Due to time constraints, hyperparameter tuning was performed using non-negativity correction with the same hy-
perparameters used for the absolute-value based methods. Therefore, these results maximally favor the baseline of
non-negativity correction.

Table 21's experiments are identical to Table 13 in Section E.2. "abs" denotes our standard aPU learning methods
(see Sections 5 and 6) while "nn" denotes our methods modified to use Kiryo et al. [8]'s non-negativity correction.
For MNIST, neither absolute-value correction nor non-negativity clearly outperformed the other. For the more
challenging 20 Newsgroups and CIFAR10 datasets, absolute-value correction had consistently better performance than
non-negativity. The only exception were the disjoint support experiments and one experimental setup for PU2wUU
on 20 Newsgroups. Although not shown in Table 13 due to limited space, both correction strategies had comparable
variance.

Table 22's experiments match the experimental conditions for the 10 LIBSVM datasets in Table 15 from Section E.4. Bi-
asing follows Sakai and Shimizu [14]'s median feature vector-based approach. Neither absolute-value nor non-negativity
correction consistently outperformed the other in these LIBSVM experiments. Note though that since absolute-value
correction is a simpler method with one less hyperparameter, γ, to tune, comparable performance implicitly favors
absolute-value correction over non-negativity.

Table 21: Comparison of mean inductive misclassification rate (%) over 100 trials for the non-overlapping support experiments in Table 13 when using absolute-value (abs) and non-negativity (nn) corrections for our aPU learning methods. The best performing method (according to mean misclassification rate) is shown in bold. A positive difference (Diff.) denotes that our absolute-value correction had better performance. Result standard deviations are comparable for both correction methods but are not shown here to improve table clarity.

	P_{test}	P_{train}	π_{tr}	π_{te}	PURR			PU2aPNU			PU2wUU		
					abs	nn	Diff.	abs	nn	Diff.	abs	nn	Diff.
MNIST	1, 3, 5, 7, 9	P_{test}	0.5	0.5	**10.0**	10.2	0.2	10.0	**9.8**	–0.2	**11.6**	11.7	0.1
		7, 9	0.29	0.5	6.8	**6.6**	–0.2	**5.3**	**5.3**	0	**6.0**	**6.0**	0
			0.5	0.5	**9.4**	9.4	0	**7.1**	**7.1**	0	**8.3**	**8.3**	0
			0.71	0.5	**14.0**	14.6	0.6	**11.1**	11.3	0.2	**14.8**	15.2	0.4
		1, 3, 5	0.38	0.5	8.1	**8.0**	–0.1	**6.5**	**6.5**	0	**7.6**	7.7	0.1
			0.5	0.5	10.0	**9.9**	–0.1	**8.4**	**8.4**	0	**10.2**	**10.2**	0
			0.63	0.5	**12.5**	12.9	0.4	**11.4**	**11.4**	0	**14.3**	14.5	0.2
	5, 7	1, 3	0.5	0.5	4.0	**3.9**	–0.1	**3.6**	**3.6**	0	**3.1**	3.2	0.1
20 Newsgroups	alt, comp, misc, rec	P_{test}	0.56	0.56	**15.4**	15.5	0.1	**14.9**	15.0	0.1	**16.7**	16.7	0
		misc, rec	0.37	0.56	**13.9**	13.9	0	**12.8**	12.8	0	**14.3**	14.3	0
			0.56	0.56	**17.5**	17.7	0.2	**13.5**	13.5	0	**15.1**	15.1	0
			0.65	0.56	**20.2**	20.8	0.6	**14.0**	14.0	0	**15.9**	15.9	0
		comp	0.37	0.56	**13.3**	13.3	0	**13.7**	13.7	0	14.5	**14.4**	–0.1
			0.56	0.56	**16.0**	16.5	0.5	**14.9**	14.9	0	**15.7**	15.7	0
			0.65	0.56	**19.2**	19.6	0.4	**15.6**	15.6	0	**16.5**	16.5	0
	soc, talk	alt, comp	0.55	0.46	5.9	**5.8**	–0.1	**7.1**	**7.1**	0	**5.6**	5.7	0.1
CIFAR10	Plane, Auto, Ship, Truck	P_{test}	0.4	0.4	**14.1**	14.3	0.2	**14.2**	14.4	0.2	**15.4**	15.8	0.4
			0.14	0.4	**11.9**	12.0	0.1	**11.9**	12.0	0.1	**12.4**	**12.4**	0
		Plane	0.4	0.4	**13.8**	14.0	0.2	**14.5**	14.6	0.1	**15.1**	15.5	0.4
			0.6	0.4	**16.1**	16.6	0.5	**16.7**	17.1	0.4	**20.0**	20.2	0.2
		Auto, Truck	0.25	0.4	**12.7**	12.8	0.1	**12.4**	12.5	0.1	**12.8**	13.0	0.2
			0.4	0.4	**14.1**	14.3	0.2	**13.9**	14.0	0.1	**14.4**	14.6	0.2
			0.55	0.4	**16.0**	16.4	0.4	**16.2**	16.3	0.1	**17.1**	17.4	0.3
	Plane, Auto	Cat, Dog	0.5	0.5	14.1	**14.0**	–0.1	14.9	**14.8**	–0.1	**11.2**	11.3	0.1

Table 22: Comparison of inductive misclassification rate (%) mean and standard deviation over 100 trials for Table 15's LIBSVM dataset experiments using Sakai and Shimizu's mean feature vector biasing with absolute-value (abs) and non-negativity (nn) corrections for our aPU learning methods. The best performing method (according to mean misclassification rate) is shown in bold. A positive difference (Diff.) denotes that our absolute-value correction had better performance than non-negativity correction.

Dataset	PURR			PU2aPNU			PU2wUU		
	abs	nn	Diff.	abs	nn	Diff.	abs	nn	Diff.
banana	**12.9 (2.1)**	**12.9 (2.2)**	0 (0.1)	11.8 (1.6)	**11.7 (1.6)**	–0.1 (0)	**13.3 (2.3)**	14.0 (2.3)	0.7 (0)
cod-rna	14.7 (2.6)	**14.6 (2.9)**	–0.1 (0.3)	15.1 (3.2)	15.1 (3.2)	0 (0)	**15.5 (2.9)**	15.5 (3.3)	0 (0.4)
susy	**24.2 (2.1)**	24.6 (2.1)	0.4 (0)	25.6 (2.2)	25.6 (2.2)	0 (0)	**25.8 (2.2)**	26.0 (2.1)	0.2 (–0.1)
ijcnn1	**22.7 (2.8)**	23.0 (2.8)	0.3 (0)	**17.7 (2.8)**	19.0 (2.9)	1.3 (0.1)	**24.6 (3.1)**	24.9 (2.9)	0.3 (–0.2)
covtype.b	**29.5 (2.9)**	29.6 (2.9)	0.1 (0)	**32.5 (3.2)**	32.6 (3.1)	0.1 (–0.1)	**29.9 (2.4)**	30.1 (2.7)	0.2 (0.3)
phishing	**11.3 (1.4)**	11.9 (1.4)	0.6 (0)	**9.6 (1.0)**	**9.6 (1.0)**	0 (0)	**11.1 (1.8)**	11.7 (1.9)	0.6 (0.1)
a9a	27.1 (2.1)	**27.0 (2.1)**	–0.1 (0)	26.6 (1.8)	**26.5 (1.8)**	–0.1 (0)	27.1 (2.1)	**27.0 (2.0)**	–0.1 (–0.1)
connect4	34.9 (3.1)	**34.2 (2.6)**	–0.7 (–0.5)	**32.9 (2.7)**	33.0 (2.7)	0.1 (0)	35.0 (2.9)	**34.9 (2.6)**	–0.1 (–0.3)
w8a	17.2 (2.6)	**17.1 (2.4)**	–0.1 (–0.2)	21.0 (2.9)	**20.3 (2.9)**	–0.7 (0)	**16.8 (2.9)**	18.4 (2.7)	1.6 (–0.2)
epsilon	33.5 (4.8)	**32.7 (3.1)**	–0.8 (–1.7)	**36.5 (5.0)**	37.8 (6.9)	1.3 (1.9)	31.5 (1.7)	**31.3 (1.7)**	–0.2 (0)

E.7 Alternate Methods for Step #1 of Our Two-Step Methods

Recall from Section 5 that our two-step methods' first step transform unlabeled training set $\mathcal{X}_{\text{tr-u}}$ into surrogate negative set $\widetilde{\mathcal{X}}_{\text{n}}$ by *soft weighting* each $x \in \mathcal{X}_{\text{tr-u}}$ using classifier

$$\hat{\sigma}_{\text{soft}}(x) := \hat{\sigma}(x) \approx p_{\text{tr}}(Y = -1|x). \tag{31}$$

In this section, we propose and empirically evaluate two alternative step #1 methods – hard and top-k weighting. Regardless of which step #1 method is used to create $\widetilde{\mathcal{X}}_{\text{n}}$, no changes are required to our step #2 risk estimators – wUU and aPNU.

E.7.1 Overview of the Alternate Step #1 Methods

Hard Weighting Guo et al. [50] show that modern neural networks are generally poorly calibrated and tend to report "peaky" confidence estimates. $\hat{\sigma}$ is vulnerable to similar "peaky" behavior. *Hard weighting* assigns each unlabeled training example, $x \in \mathcal{X}_{\text{tr-u}}$, weight

$$\hat{\sigma}_{\text{hard}}(x) := \lfloor \hat{\sigma}(x) \rceil \tag{32}$$

where for $a \in \mathbb{R}$, $\lfloor a \rceil$ rounds a to the nearest integer (i.e., 0 or 1 for probabilistic classifier $\hat{\sigma}$).

Hard weighting simulates worst-case "peaked" behavior. Although not statistically consistent for non-separable data, hard weighting may sometimes outperform soft-weighting due to its thresholding effect.

Top-k Weighting To broadly summarize Guo et al.'s primary contribution, neural network probability estimates may be inaccurate. Our *top-k weighting* method attempts to overcome that inaccuracy by focusing, not on the specific probability values predicted by $\hat{\sigma}$, but instead on the *ordering* of those posterior estimates.

By definition, the expected number of positive-labeled examples in $\mathcal{X}_{\text{tr-u}}$ is $\pi_{\text{tr}} \cdot n_{\text{tr-u}}$, where $n_{\text{tr-u}} := |\mathcal{X}_{\text{tr-u}}|$ is the unlabeled training set size and π_{tr} is the positive training prior. Define $k := \lfloor \pi_{\text{tr}} \cdot n_{\text{tr-u}} \rceil \in \mathbb{Z}_+$. After training $\hat{\sigma}$ (same as before), let set $\mathcal{X}_{\text{tr-u-k}}$ be the k examples in $\mathcal{X}_{\text{tr-u}}$ with the highest predicted posteriors according to $\hat{\sigma}$. Top-k weighting assigns weight 1 to any $x \in \mathcal{X}_{\text{tr-u-k}}$ and weight 0 to any $x \in (\mathcal{X}_{\text{tr-u}} \setminus \mathcal{X}_{\text{tr-u-k}})$. Formally, for any $x \in \mathcal{X}_{\text{tr-u}}$,

$$\hat{\sigma}_{\text{top-k}}(x) := \begin{cases} 1 & x \in \mathcal{X}_{\text{tr-u-k}} \\ 0 & \text{Otherwise} \end{cases}. \tag{33}$$

Observe that top-k weighting uses strictly more information than both soft and hard weighting. However, by relying on π_{tr} to estimate k, top-k weighting is generally more deleteriously affected by misestimation of π_{tr}.

E.7.2 Step #1 Labeling Accuracy

These experiments examine how accurately our three proposed step #1 methods label $\mathcal{X}_{\text{tr-u}}$. The labeling error rate is defined as

$$\text{Error Rate}_{\mathcal{M}} := \frac{100\%}{n_{\text{tr-u}}} \sum_{x \in \mathcal{X}_{\text{tr-u}}} \frac{|2\hat{\sigma}_{\mathcal{M}}(x) - 1 - y_x|}{2}, \tag{34}$$

where $y_x \in \{\pm 1\}$ is unlabeled training example x's true (unknown) label and $\mathcal{M} \in \{\text{soft}, \text{hard}, \text{top-k}\}$ denotes the step #1 method. For hard and top-k weighting, Eq. (34) corresponds to their (scaled) transductive misclassification rate on $\mathcal{X}_{\text{tr-u}}$. Note that the difference between the soft and hard weightings' labeling error rates is indicative of the "peakiness" of $\hat{\sigma}$, with a smaller gap indicating that $\hat{\sigma}$'s estimates are more peaked.

For all experiments in this section, the three step #1 methods saw identical dataset splits and used the same initial model parameters.

Analysis Table 23 compares the three weighting methods' step #1 labeling error rate for the 10 LIBSVM datasets in Table 15 (see Section E.4). Recall that in these experiments, the positive-train and positive-test class conditionals have identical supports. The step #1 methods' labeling error rates varied widely from around 10% on the `phishing` dataset to 30–40% for the `covtype.b` and `epsilon` datasets.

Recall from Table 15 that PURR was the top performer for the `cod-rna`, `susy`, and `covtype.b` datasets. The step #1 labeling error on those three datasets ranged from moderate to poor. However, `epsilon` had soft weighting's worst step #1 labeling error rate yet PU2wUU still outperformed PURR (see Table 15). This demonstrates that step #1 labeling accuracy alone does not determine which algorithm class, i.e., two-step or joint, is best.

Table 23: Comparison of the soft, hard, and top-k weighting schemes' step #1 labeling error rate mean and standard deviation across 100 trials for the 10 LIBSVM datasets in Table 15.

Dataset	d	Soft	Hard	Top-k
banana	2	20.1 (3.8)	13.2 (1.8)	12.4 (1.8)
cod-rna	8	20.8 (4.0)	13.2 (1.9)	12.7 (1.7)
susy	18	39.6 (2.7)	30.7 (2.4)	30.6 (2.4)
ijcnn1	22	27.0 (4.5)	19.3 (2.5)	15.8 (2.7)
covtype.b	54	44.1 (4.2)	37.1 (3.3)	34.9 (2.7)
phishing	68	13.5 (3.6)	10.4 (1.5)	9.6 (1.1)
a9a	123	24.4 (3.8)	17.4 (1.4)	18.1 (1.8)
connect4	128	34.2 (5.5)	27.8 (4.4)	24.7 (2.4)
w8a	300	27.9 (5.1)	19.6 (1.6)	18.5 (1.7)
epsilon	2,000	44.2 (1.5)	32.5 (2.6)	33.7 (2.0)

Table 24 compares the three weighting methods' step #1 labeling error rate for the extended set of experiments in Table 13 (see Section E.2). Recall that those experiments, on datasets MNIST, 20 Newsgroups, and CIFAR10, replicated scenarios where some positive subclasses exist only in the test distribution. As expected, the easier MNIST dataset had better step #1 labeling accuracy than the more challenging 20 Newsgroups and CIFAR10 datasets. On the whole, these three datasets had better average step #1 labeling error rate than the 10 LIBSVM datasets discussed above.

E.7.3 Step #1 Method's Effect on Overall Two-Step Performance

These experiments study how each step #1 method affects our two step methods' – PU2aPNU and PU2wUU – inductive, test (i.e., end-to-end) misclassification rate. As in the previous section, all methods saw identical dataset splits and initial model parameters in each experimental trial.

Analysis   Table 25 compares the two-step inductive misclassification rate when using the three step #1 methods for the 10 LIBSVM datasets in Table 15 (see Section E.4). Soft weighting was the best performing method for all ten datasets for PU2aPNU and for seven of ten datasets for PU2wUU. It is also noteworthy that only soft weighting learned a meaningful classifier for the epsilon dataset. In fact, hard and top-k weighting performed worse than random chance for epsilon.

Table 26 compares the two-step, inductive misclassification rate when using the three step #1 methods for the experiments in Table 13 on MNIST, 20 Newsgroups, and CIFAR10 (see Section E.2). For the vast majority of setups, top-k weighting was the best performing method for both PU2aPNU and PU2wUU. Top-k often improved performance over soft weighting by 10–20% or more – in particular for PU2wUU. The one experimental setup where soft weighting consistently performed as well or better than top-k was when the positive train and test supports were disjoint. Observe that in those experiments, the positive and negative classes are composed of fewer constituent labels. As such, we believe that top-k weighting is exacerbating overfitting in those models resulting in the worse performance.

E.7.4 Discussion

The experiments in the previous subsection demonstrate that the best performing step #1 method is benchmark/setup dependent. If a user is highly confident that their data is readily and easily separable (like MNIST), top-k weighting may perform particularly well. Although not shown here, we empirically observed that misestimation of training prior π_{tr} negatively affects top-k weighting's accuracy – many times severely.

If the training datasets (e.g., $\mathcal{X}_p$, $\mathcal{X}_{tr-u}$, and $\mathcal{X}_{te-u}$) are large enough that asymptotic consistency guarantees generally apply, soft weighting may perform best. We made soft-weighting the focus of Section 5 due to its stronger statistical guarantees. Had top-k weighting been used in Section 7.2's experiments instead of soft weighting, our performance advantage over the baselines, PUc and nnPU*, would have widened.

Table 24: Comparison of the soft, hard, and top-k weighting schemes' step #1 labeling error rate mean and standard deviation across 100 trials for Table 13's experiments on partially/fully disjoint positive-class support for MNIST, 20 Newsgroups, and CIFAR10.

	N	P_{test}	P_{train}	π_{tr}	π_{te}	Soft	Hard	Top-k
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	P_{test}	0.5	0.5	16.2 (2.3)	12.6 (1.0)	10.6 (1.0)
				0.29	0.5	11.0 (2.0)	6.8 (0.8)	5.4 (0.5)
			7, 9	0.5	0.5	10.8 (1.9)	8.3 (0.9)	6.5 (0.6)
				0.71	0.5	11.1 (2.4)	8.3 (0.5)	7.7 (0.6)
				0.38	0.5	12.6 (1.6)	9.0 (0.9)	7.3 (0.7)
			1, 3, 5	0.5	0.5	14.0 (2.4)	10.7 (0.9)	9.0 (1.0)
				0.63	0.5	15.0 (3.1)	11.4 (0.7)	10.3 (1.0)
	0, 2	5, 7	1, 3	0.5	0.5	8.9 (1.7)	6.5 (0.7)	5.4 (0.4)
20 Newsgroups	sci, soc, talk	alt, comp, misc, rec	P_{test}	0.56	0.56	23.1 (4.3)	16.9 (1.3)	16.5 (1.3)
				0.37	0.56	15.5 (1.3)	12.0 (1.4)	9.3 (1.3)
			misc, rec	0.56	0.56	14.1 (1.5)	11.6 (1.0)	10.2 (1.2)
				0.65	0.56	12.8 (1.5)	10.3 (0.9)	9.8 (1.1)
				0.37	0.56	15.7 (0.9)	12.1 (0.8)	11.1 (1.0)
			comp	0.56	0.56	14.3 (1.2)	12.0 (1.1)	11.6 (1.2)
				0.65	0.56	12.8 (1.3)	10.7 (1.1)	11.1 (1.3)
	misc, rec	soc, talk	alt, comp	0.55	0.46	12.4 (1.0)	10.6 (1.1)	10.1 (1.2)
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	P_{test}	0.4	0.4	21.3 (3.1)	16.6 (1.3)	14.6 (0.7)
				0.14	0.4	16.6 (3.1)	9.2 (0.6)	8.4 (0.5)
			Plane	0.4	0.4	21.3 (4.0)	14.7 (1.1)	13.1 (0.7)
				0.6	0.4	21.8 (3.0)	15.0 (1.0)	13.8 (0.6)
				0.25	0.4	14.7 (1.4)	10.4 (0.8)	9.1 (0.5)
			Auto, Truck	0.4	0.4	15.7 (2.6)	12.5 (1.1)	10.9 (0.7)
				0.55	0.4	17.0 (3.3)	13.3 (1.2)	11.8 (0.6)
	Deer, Horse	Plane, Auto	Cat, Dog	0.5	0.5	30.9 (2.6)	21.4 (1.1)	21.0 (1.0)

Table 25: Effect of step #1 method on our two-step methods' overall inductive misclassification rate (%) for the 10 LIBSVM datasets in Table 15. The table's upper half reports each method's misclassification rate mean and standard deviation over 100 trials. Boldface denotes each experimental setup's best performing method according to mean misclassification rate. The table's lower half is an alternate visualization showing the difference (Diff.) in misclassification rate mean and standard deviation w.r.t. to our soft method. Red denotes that the associated alternate step #1 method had worse (i.e., higher) mean misclassification rate than soft weighting while green denotes that the alternate method had a better (i.e., lower) mean misclassification rate.

Dataset	d	PU2aPNU			PU2wUU		
		Soft	Hard	Top-k	Soft	Hard	Top-k
banana	2	**11.7 (1.6)**	13.4 (2.0)	13.1 (1.9)	13.4 (2.4)	14.0 (2.6)	**13.3 (2.5)**
cod-rna	8	**14.6 (3.8)**	18.6 (3.7)	18.3 (3.9)	**15.5 (3.2)**	17.4 (3.2)	16.7 (3.2)
susy	18	**25.8 (2.6)**	27.8 (3.2)	27.6 (2.5)	**25.8 (2.4)**	26.3 (3.5)	26.1 (2.6)
ijcnn1	22	**18.0 (2.7)**	22.1 (3.6)	18.4 (2.9)	25.1 (3.4)	24.3 (3.3)	**21.4 (2.9)**
covtype.b	54	**32.1 (3.2)**	40.6 (3.5)	37.0 (3.5)	**29.7 (2.5)**	40.1 (3.7)	34.8 (4.2)
phishing	68	**9.6 (0.9)**	9.8 (1.0)	10.0 (1.1)	11.6 (2.1)	10.9 (1.4)	**10.1 (1.2)**
a9a	123	**26.8 (1.6)**	28.6 (1.7)	27.9 (1.6)	**27.4 (2.1)**	28.5 (1.8)	28.3 (1.9)
connect4	126	**32.9 (2.1)**	37.2 (2.8)	35.6 (2.3)	**34.8 (2.7)**	38.0 (3.1)	35.3 (2.8)
w8a	300	**21.6 (2.4)**	23.7 (2.0)	24.6 (2.3)	**16.9 (2.7)**	22.4 (2.4)	22.0 (2.7)
epsilon	2,000	**35.0 (4.4)**	58.6 (3.2)	54.6 (3.4)	**31.2 (1.1)**	52.6 (3.9)	52.9 (5.8)

Dataset	d	PU2aPNU			PU2wUU		
		Soft	Diff. Hard	Diff. Top-k	Soft	Diff. Hard	Diff. Top-k
banana	2	11.7 (1.6)	1.7 (0.4)	1.4 (0.4)	13.4 (2.4)	0.5 (0.2)	–0.1 (0.1)
cod-rna	8	14.6 (3.8)	4.1 (–0.1)	3.7 (0.1)	15.5 (3.2)	1.9 (0)	1.2 (0)
susy	18	25.8 (2.6)	2.0 (0.6)	1.9 (–0.1)	25.8 (2.4)	0.6 (1.1)	0.4 (0.2)
ijcnn1	22	18.0 (2.7)	4.1 (0.8)	0.3 (0.2)	25.1 (3.4)	–0.8 (0.2)	–3.7 (–0.5)
covtype.b	54	32.1 (3.2)	8.5 (0.2)	4.9 (0.3)	29.7 (2.5)	10.3 (1.2)	5.0 (1.7)
phishing	68	9.6 (0.9)	0.2 (0)	0.4 (0.2)	11.6 (2.1)	–0.6 (–0.7)	–1.5 (–0.9)
a9a	123	26.8 (1.6)	1.9 (0.2)	1.2 (0)	27.4 (2.1)	1.2 (–0.4)	0.9 (–0.2)
connect4	126	32.9 (2.1)	4.3 (0.7)	2.7 (0.2)	34.8 (2.7)	3.3 (0.4)	0.6 (0.1)
w8a	300	21.6 (2.4)	2.1 (–0.4)	3.0 (–0.1)	16.9 (2.7)	5.6 (–0.3)	5.1 (0)
epsilon	2,000	35.0 (4.4)	23.7 (–1.2)	19.6 (–0.9)	31.2 (1.1)	21.4 (2.8)	21.7 (4.6)

Table 26: Effect of step #1 method on our two-step methods' overall inductive misclassification rate (%) for the MNIST, 20 Newsgroups, and CIFAR10 datasets. The table's upper half reports each method's misclassification rate mean and standard deviation over 100 trials. Boldface denotes each experimental setup's best performing method according to mean misclassification rate. The table's lower half is an alternate visualization showing the difference (Diff.) in misclassification rate mean and standard deviation w.r.t. to our soft method. Red denotes that the associated alternate step #1 method had worse (i.e., higher) mean misclassification rate than soft weighting while green denotes that the alternate method had a better (i.e., lower) mean misclassification rate.

	N	P_{test}	P_{train}	π_{tr}	π_{te}	PU2aPNU Soft	Hard	Top-k	PU2wUU Soft	Hard	Top-k
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	P_{test}	0.5	0.5	10.2 (1.5)	9.8 (1.3)	**7.8 (1.0)**	11.8 (1.5)	10.6 (1.1)	**9.3 (1.0)**
				0.29	0.5	5.4 (0.5)	5.3 (0.5)	**4.9 (0.4)**	6.1 (0.7)	**5.5 (0.4)**	5.6 (0.3)
			7, 9	0.5	0.5	6.9 (0.9)	7.7 (1.2)	**5.9 (0.6)**	8.0 (1.3)	7.5 (0.9)	**6.4 (0.5)**
				0.71	0.5	11.0 (1.4)	12.9 (1.2)	**9.9 (1.3)**	14.9 (3.7)	12.9 (1.1)	**9.9 (1.0)**
				0.38	0.5	6.4 (0.8)	6.6 (0.8)	**5.7 (0.6)**	7.6 (0.9)	7.0 (0.7)	**6.6 (0.6)**
			1, 3, 5	0.5	0.5	8.4 (1.1)	9.0 (1.0)	**7.2 (0.9)**	10.0 (1.4)	9.3 (0.9)	**8.0 (0.8)**
				0.63	0.5	11.3 (1.4)	12.8 (1.4)	**10.2 (1.4)**	14.1 (2.5)	12.9 (1.2)	**10.5 (1.1)**
	0, 2	5, 7	1, 3	0.5	0.5	3.5 (1.0)	4.1 (1.1)	**2.8 (0.6)**	3.1 (0.7)	3.7 (0.9)	**2.8 (0.4)**
20 Newsgroups	sci, soc, talk	alt, comp, misc, rec	P_{test}	0.56	0.56	14.9 (1.3)	14.9 (1.4)	**14.4 (1.5)**	16.6 (2.5)	15.9 (1.8)	**15.5 (1.9)**
				0.37	0.56	12.8 (0.6)	12.9 (0.8)	**12.4 (0.6)**	14.2 (0.9)	13.7 (0.9)	**13.1 (0.7)**
			misc, rec	0.56	0.56	13.6 (0.9)	14.0 (0.9)	**13.4 (0.9)**	15.1 (1.3)	14.8 (1.1)	**14.1 (1.1)**
				0.65	0.56	14.0 (0.9)	14.4 (0.9)	**13.8 (0.9)**	15.8 (1.3)	15.3 (1.2)	**14.6 (0.9)**
				0.37	0.56	13.7 (0.6)	13.8 (0.6)	**13.0 (0.7)**	14.5 (0.8)	14.1 (0.7)	**13.3 (0.7)**
			comp	0.56	0.56	14.9 (0.7)	15.7 (0.7)	**14.3 (0.8)**	15.7 (0.9)	15.9 (0.8)	**14.6 (0.9)**
				0.65	0.56	15.5 (1.1)	16.5 (1.0)	**15.2 (1.2)**	16.3 (1.4)	16.7 (1.3)	**15.4 (1.3)**
	misc, rec	soc, talk	alt, comp	0.55	0.46	**7.2 (1.2)**	8.1 (1.2)	7.5 (1.2)	**5.8 (1.6)**	7.1 (1.5)	**5.8 (1.4)**
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	P_{test}	0.4	0.4	13.9 (1.2)	13.6 (0.9)	**12.0 (0.7)**	15.0 (1.2)	14.7 (0.9)	**13.2 (0.8)**
				0.14	0.4	**12.0 (0.8)**	12.1 (0.6)	12.2 (0.7)	12.5 (0.9)	11.8 (0.6)	**11.7 (0.7)**
			Plane	0.4	0.4	14.4 (1.3)	15.4 (1.1)	**14.1 (0.8)**	14.9 (1.4)	15.0 (1.2)	**13.2 (0.8)**
				0.6	0.4	**16.7 (1.5)**	20.0 (1.5)	16.9 (1.1)	20.1 (2.3)	20.0 (1.8)	**15.5 (1.1)**
			Auto, Truck	0.25	0.4	**12.4 (0.7)**	12.6 (0.7)	**12.4 (0.7)**	12.8 (0.7)	12.4 (0.7)	**12.2 (0.7)**
				0.4	0.4	14.0 (1.2)	14.7 (1.1)	**13.4 (0.8)**	14.4 (1.2)	14.6 (1.3)	**13.1 (0.8)**
				0.55	0.4	16.2 (1.6)	17.7 (1.8)	**15.3 (1.0)**	17.0 (2.1)	17.7 (2.1)	**14.8 (1.0)**
	Deer, Horse	Plane, Auto	Cat, Dog	0.5	0.5	**15.1 (1.7)**	20.2 (1.2)	19.2 (1.1)	**11.2 (0.8)**	16.3 (1.3)	14.2 (1.0)

	N	P_{test}	P_{train}	π_{tr}	π_{te}	PU2aPNU Soft	Diff. Hard	Diff. Top-k	PU2wUU Soft	Diff. Hard	Diff. Top-k
MNIST	0, 2, 4, 6, 8	1, 3, 5, 7, 9	P_{test}	0.5	0.5	10.2 (1.5)	−0.4 (−0.2)	−2.5 (−0.5)	11.8 (1.5)	−1.3 (−0.4)	−2.6 (−0.5)
				0.29	0.5	5.4 (0.5)	−0.1 (0)	−0.5 (−0.1)	6.1 (0.7)	−0.5 (−0.3)	−0.5 (−0.3)
			7, 9	0.5	0.5	6.9 (0.9)	0.8 (0.3)	−1.0 (−0.3)	8.0 (1.3)	−0.5 (−0.4)	−1.6 (−0.8)
				0.71	0.5	11.0 (1.4)	1.9 (−0.2)	−1.1 (−0.1)	14.9 (3.7)	−2.0 (−2.7)	−5.0 (−2.7)
				0.38	0.5	6.4 (0.8)	0.2 (0)	−0.7 (−0.2)	7.6 (0.9)	−0.6 (−0.2)	−1.0 (−0.3)
			1, 3, 5	0.5	0.5	8.4 (1.1)	0.6 (0)	−1.2 (−0.2)	10.0 (1.4)	−0.8 (−0.5)	−2.1 (−0.7)
				0.63	0.5	11.3 (1.4)	1.5 (0)	−1.1 (0)	14.1 (2.5)	−1.2 (−1.3)	−2.1 (−0.7)
	0, 2	5, 7	1, 3	0.5	0.5	3.5 (1.0)	0.6 (0.1)	−0.7 (−0.4)	3.1 (0.7)	0.6 (0.3)	−0.3 (−0.3)
20 Newsgroups	sci, soc, talk	alt, comp, misc, rec	P_{test}	0.56	0.56	14.9 (1.3)	0 (0.2)	−0.5 (0.3)	16.6 (2.5)	−0.7 (−0.8)	−1.1 (−0.6)
				0.37	0.56	12.8 (0.6)	0.1 (0.1)	−0.4 (0)	14.2 (0.9)	−0.5 (0)	−1.1 (−0.2)
			misc, rec	0.56	0.56	13.6 (0.9)	0.3 (0)	−0.2 (0)	15.1 (1.3)	−0.3 (−0.2)	−1.0 (−0.2)
				0.65	0.56	14.0 (0.9)	0.4 (0)	−0.2 (−0.1)	15.8 (1.3)	−0.5 (−0.1)	−1.2 (−0.4)
				0.37	0.56	13.7 (0.6)	0.1 (0)	−0.6 (0)	14.5 (0.8)	−0.4 (0)	−1.2 (0)
			comp	0.56	0.56	14.9 (0.7)	0.8 (0)	−0.6 (0)	15.7 (0.9)	0.2 (−0.1)	−0.1 (−0)
				0.65	0.56	15.5 (1.1)	1.0 (0)	−0.4 (0.1)	16.3 (1.4)	0.4 (−0.1)	−0.9 (−0.1)
	misc, rec	soc, talk	alt, comp	0.55	0.46	7.2 (1.2)	1.0 (0.1)	0.3 (0)	5.8 (1.6)	1.4 (−0.1)	0 (−0.3)
CIFAR10	Bird, Cat, Deer, Dog, Frog, Horse	Plane, Auto, Ship, Truck	P_{test}	0.4	0.4	13.9 (1.2)	−0.3 (−0.3)	−1.9 (−0.5)	15.0 (1.2)	−0.4 (−0.3)	−1.9 (−0.4)
				0.14	0.4	12.0 (0.8)	0.1 (−0.1)	0.1 (−0.1)	12.5 (0.9)	−0.8 (−0.3)	−0.8 (−0.2)
			Plane	0.4	0.4	14.4 (1.3)	1.0 (−0.2)	−0.3 (−0.5)	14.9 (1.4)	0.1 (−0.2)	−1.8 (−0.6)
				0.6	0.4	16.7 (1.5)	3.3 (0)	0.2 (−0.4)	20.1 (2.3)	−0.1 (−0.5)	−4.6 (−1.2)
			Auto, Truck	0.25	0.4	12.4 (0.7)	0.2 (0)	0 (0)	12.8 (0.7)	−0.4 (−0.1)	−0.6 (0)
				0.4	0.4	14.0 (1.2)	0.6 (−0.1)	−0.6 (−0.4)	14.4 (1.2)	0.3 (0.1)	−1.3 (−0.4)
				0.55	0.4	16.2 (1.6)	1.5 (0.2)	−0.9 (−0.7)	17.0 (2.1)	0.7 (0)	−2.2 (−1.1)
	Deer, Horse	Plane, Auto	Cat, Dog	0.5	0.5	15.1 (1.7)	5.1 (−0.5)	4.1 (−0.6)	11.2 (0.8)	5.1 (0.6)	3.0 (0.3)

E.8 Analyzing the Effect of Positive and Negative Class-Conditional Distribution Shift

The goal of these experiments is to:

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

We look at these trends across three datasets (as in Section 7.2): 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 27 enumerates each dataset's positive and negative class definitions; these definitions apply for both train and test. The dataset sizes are listed in Table 28; 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 27: 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 28: 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 class-conditional 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 class-conditional 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., $C_1 =$ Auto and $C_2 =$ 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,

$$\Pr[\text{Label}_{\text{tr}} = C_2 | Y = +1] = 1 - \Pr[\text{Label}_{\text{tr}} = C_1 | Y = +1]. \tag{35}$$

The positive-train class-conditional 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 7a, 7c, and 7e 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 7.2). 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_{tr} and PN_{te} are trained using (labeled) $\mathcal{X}_{\text{tr-u}}$ and $\mathcal{X}_{\text{te-u}}$. Since the test distributions are never biased, PN_{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_{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 class-conditional distribution shift described above except that the bias is instead applied to the negative-train class-conditional 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. (7)'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). Now, $\Pr[\text{Label}_{\text{tr}} = C_1 | Y = -1]$ is 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 7b, 7d, and 7f respectively.

With the exception of PU2wUU on MNIST, all of our methods showed moderate robustness to some negative class-conditional 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.
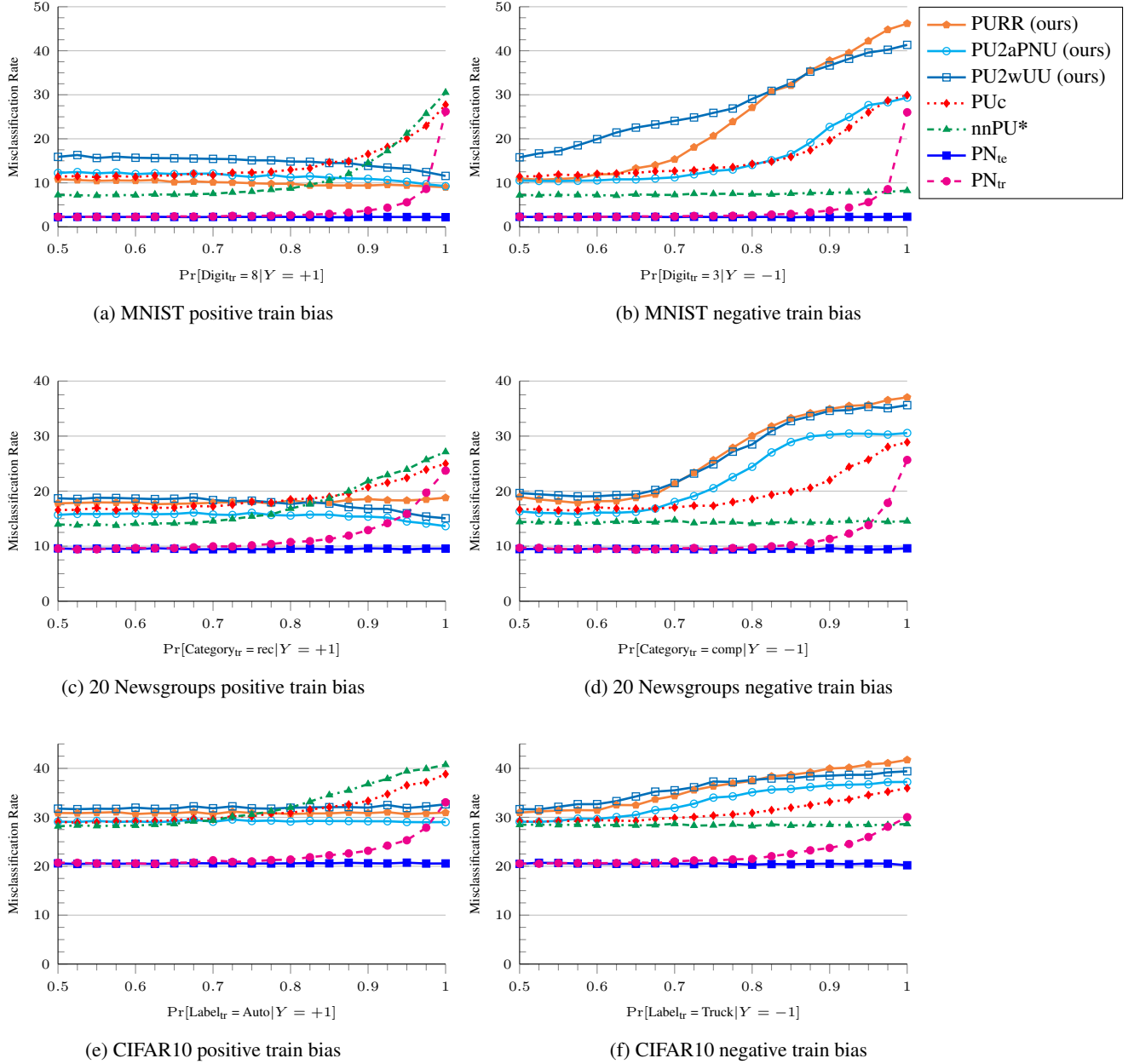
Figure 7: Effect of positive ($p_{\text{tr-p}}(x)$) or negative ($p_{\text{tr-n}}(x)$) training class-conditional distribution shift on inductive misclassification rate (%) for the MNIST, 20 Newsgroups, and CIFAR10 datasets. The x-axis corresponds to $\Pr[\text{Label}_{\text{tr}} = C_1 | y = \hat{y}]$ where $\hat{y} \in \{\pm 1\}$. Each data point is the average of 100 trials.

E.9 Effect of Prior Probability Misestimation

As explained in Section 4, this work treats the positive-class priors, π_{tr} and π_{te}, as known. This set of experiments examines our methods' performance when the priors are misspecified.

Experimental Setup   These experiments reuse the partially disjoint positive-support experiment setups from Section 7.2's Table 1. Therefore, we are specifically considering the MNIST, 20 Newsgroups, and CIFAR10 datasets with Table 29 summarizing the experimental setups.

π_{tr} and π_{te} in Table 29 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. 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}. Like all previous experiments, performance was evaluated using the inductive (test) misclassification rate, and all methods saw identical datasets splits in each trial.

Analysis   Tables 30, 31, and 32 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 by row and π_{te} changing column by column. Each cell is shaded red, with a darker background denoting worse performance, i.e., a greater misclassification rate. In all but one setup, our methods outperformed PUc.

Similar to Section E.8'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.

Of our three methods, PU2aPNU was the least affected by misspecified priors. For the two-step methods, the worst performing misestimation profile was dataset specific. In contrast, PURR's performance was always worst when the train and test priors were misspecified in opposite directions. To understand why this is, recall that PURR's definition in Eq. (14) includes prior ratio $\frac{1-\pi_{\text{te}}}{1-\pi_{\text{tr}}}$. This ratio compounds prior misestimations with opposite signs.

As an example, consider the MNIST experiment below with true priors $\pi_{\text{tr}} = \pi_{\text{te}} = 0.5$, making PURR's ideal prior ratio $\frac{1-0.5}{1-0.5} = 1$. This ratio remains 1 even if the priors are misspecified as $\pi_{\text{tr}} = \pi_{\text{te}} = 0.6$ or $\pi_{\text{tr}} = \pi_{\text{te}} = 0.4$. In contrast, if $\pi_{\text{tr}} = 0.4$ and $\pi_{\text{te}} = 0.6$, PURR's (erroneous) prior ratio is $\frac{1-0.6}{1-0.4} \approx 0.67$ – a 33% error. Furthermore, when the priors are misspecified as $\pi_{\text{tr}} = 0.6$ and $\pi_{\text{te}} = 0.4$, the prior ratio jumps to $\frac{1-0.4}{1-0.6} = 1.5$ – a 50% error. This is why over-estimation of the training prior and underestimation of the test prior is always PURR's worst performing configuration.

Table 29: 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 30: Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on <u>MNIST</u>'s inductive misclassification rate (%). Each result is the average of 100 trials.

	PURR			PU2aPNU			PU2wUU			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 31: Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on <u>20 Newsgroups</u>'s inductive misclassification rate (%). Each result is the average of 100 trials.

	PURR			PU2aPNU			PU2wUU			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}	$1.2\pi_{\text{te}}$	
$0.8\pi_{\text{tr}}$	15.2	14.9	16.5	12.3	12.5	13.3	13.4	14.3	16.7	34.1
π_{tr}	15.9	13.8	15.1	12.6	12.8	13.5	13.4	14.2	16.1	28.6
$1.2\pi_{\text{tr}}$	18.7	13.3	14.0	13.4	14.2	15.0	14.4	15.6	17.2	24.9

Table 32: Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on <u>CIFAR10</u>'s inductive misclassification rate (%). Each result is the average of 100 trials.

	PURR			PU2aPNU			PU2wUU			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}	$1.2\pi_{\text{te}}$	
$0.8\pi_{\text{tr}}$	16.4	15.6	18.0	13.9	15.1	17.5	18.2	19.3	21.1	23.4
π_{tr}	16.0	13.7	15.3	13.3	14.4	16.6	14.2	14.9	16.9	20.1
$1.2\pi_{\text{tr}}$	20.8	15.7	14.7	16.8	16.4	17.5	15.6	15.9	17.9	19.7