Positive-Unlabeled Learning with Arbitrarily Non-Representative Labeled Data

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

Abstract

Positive-unlabeled (PU) *learning* trains a binary classifier using only labeled-positive and unlabeled data. A common simplifying assumption is that the labeled data is representative of the target positive class, but this assumption rarely holds in practice. This papers show that PU learning is possible even with arbitrarily non-representative labeled-positive data. Our key insight is that only the negative class's distribution need be fixed. We integrate this idea 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. 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. Often good labeled data are unavailable for one class. High negative-class diversity may make constructing a representative labeled set prohibitively difficult (du Plessis et al., 2015), and some domains (e.g., medical records) do not systematically record negative data (Bekker & Davis, 2018).

Positive-unlabeled (PU) *learning* addresses this problem by training classifiers using only labeled-positive and unlabeled data. Real world-domains where PU learning has been applied include: opinion spam detection (Hernández Fusilier et al., 2013), disease-gene identification (Yang et al., 2012), and protein similarity prediction (Elkan & Noto, 2008). *Negative-unlabeled* (NU) learning is functionally identical to PU learning but with labeled-negative data.

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

Presented at the ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning.

Most PU learning methods assume the labeled set is *selected completely at random* (SCAR) (Elkan & Noto, 2008) from the target positive distribution. External factors (e.g., temporal drift, domain shift, and adversarial drift) can cause a shift between the labeled-positive and target distributions. *Biased-positive, unlabeled* (bPU) *learning* relaxes SCAR by modeling *selection bias* for the labeled data (Bekker et al., 2019; Kato et al., 2019) or a *covariate shift* between the training and target distributions (Sakai & Shimizu, 2019).

We generalize bPU learning to the more challenging *arbitrary-positive, unlabeled* (aPU) learning, where the target-positive class may *shift arbitrarily* w.r.t. the labeled-positive data. Solving this problem would enable deep learning to be applied to more problems where labeled data is limited; it would also eliminate the cost to label new data after a positive class shift. But, devoid of some assumption, aPU learning is impossible (Elkan & Noto, 2008). Our key insight is that given a labeled-positive set and two unlabeled sets, aPU learning is possible when *all negative examples are generated from a single distribution*. The labeled and target positive distributions' *supports* may even be disjoint.

Many real-world PU learning tasks feature a shifting positive class but (largely) fixed negative class including:

1. **Virological Analysis**: Influenza varies annually in severity and type. Infection rate prediction in a *severe* flu season is possible provided data from the same geographic region the previous year when the flu was *mild* (Alaiz-Rodríguez & Japkowicz, 2008). Considering healthy patients from the same region year-on-year keeps the negative-class distribution largely static.
2. **Adversarial PU Learning**: Malicious adversaries (e.g., malware authors) rapidly adapt their attacks to avoid automated detection. The benign class changes much more slowly but may be too diverse to create a representative labeled set (Li et al., 2014; Zhang et al., 2017; Zhang et al., 2019).

Our primary contributions are three-fold with most experiments and all theorems (with proofs) in the supplementals:

1. We propose abs-PU — a simplified *consistent* technique to correct general PU risk overfitting.

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.

2. Ordinary Positive-Unlabeled Learning

As a brief overview of PU learning without bias[1], consider two random variables, covariate $X \in \mathbb{R}^d$ ($d \in \mathbb{N}^+$) and label $Y \in \{\pm 1\}$, with joint distribution $p(x, y)$. Marginal distribution $p_{\mathrm{u}}(x)$ is composed from prior $\pi \coloneqq p(Y{=}{+}1)$, positive class-conditional $p_{\mathrm{p}}(x) \coloneqq p(x|Y{=}{+}1)$, and negative class-conditional $p_{\mathrm{n}}(x) \coloneqq p(x|Y{=}{-}1)$.

Risk Let $g : \mathbb{R}^d \to \mathbb{R}$ be any *decision function* parameterized by θ, and let $\ell : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be the *loss function*. Risk $R(g) \coloneqq \mathbb{E}_{(X,Y)\sim p(x,y)}[\ell(Yg(X))]$ quantifies g's expected loss over $p(x, y)$. It decomposes via the product rule to $R(g) = \pi R^+_{\mathrm{p}}(g) + (1-\pi)R^-_{\mathrm{n}}(g)$, where the *labeled risk* is $R^{\hat{y}}_{\mathcal{D}}(g) \coloneqq \mathbb{E}_{X\sim p_{\mathcal{D}}(x)}[\ell(\hat{y}g(X))]$, for predicted label $\hat{y} \in \{\pm 1\}$ and distribution $\mathcal{D} \in \{\mathrm{p}, \mathrm{n}, \mathrm{u}\}$.

Consider the *case-control scenario* where each dataset is i.i.d. sampled from its associated distribution. PN learning has two labeled datasets: positive set $\mathcal{X}_{\mathrm{p}} \coloneqq \{x^{\mathrm{p}}_i\}_{i=1}^{n_{\mathrm{p}}} \overset{\text{i.i.d.}}{\sim} p_{\mathrm{p}}(x)$ and negative set $\mathcal{X}_{\mathrm{n}} \coloneqq \{x^{\mathrm{n}}_i\}_{i=1}^{n_{\mathrm{n}}} \overset{\text{i.i.d.}}{\sim} p_{\mathrm{n}}(x)$. These are used to calculate labeled *empirical risks* $\widehat{R}^+_{\mathrm{p}}(g) = \frac{1}{n_{\mathrm{p}}}\sum_{i=1}^{n_{\mathrm{p}}} \ell(g(x^{\mathrm{p}}_i))$ and $\widehat{R}^-_{\mathrm{n}}(g) = \frac{1}{n_{\mathrm{n}}}\sum_{i=1}^{n_{\mathrm{n}}} \ell(-g(x^{\mathrm{n}}_i))$. The empirical *positive-negative risk* is: $\widehat{R}_{\mathrm{PN}}(g) \coloneqq \pi\widehat{R}^+_{\mathrm{p}}(g) + (1-\pi)\widehat{R}^-_{\mathrm{n}}(g)$.

PU learning cannot directly estimate $\widehat{R}^{\hat{y}}_{\mathrm{n}}(g)$ since $\mathcal{X}_{\mathrm{n}} = \emptyset$. Let $\mathcal{X}_{\mathrm{u}} \coloneqq \{x^{\mathrm{u}}_i\}_{i=1}^{n_{\mathrm{u}}} \overset{\text{i.i.d.}}{\sim} p_{\mathrm{u}}(x)$ be an unlabeled set with $\widehat{R}^{\hat{y}}_{\mathrm{u}}(g) \coloneqq \frac{1}{n_{\mathrm{u}}}\sum_{i=1}^{n_{\mathrm{u}}} \ell(\hat{y}g(x^{\mathrm{u}}_i))$. du Plessis et al. (2014) make a foundational contribution that,

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

du Plessis et al.'s unbiased PU (uPU) estimator is then $\widehat{R}_{\mathrm{uPU}}(g) \coloneqq \pi\widehat{R}^+_{\mathrm{p}}(g) + \widehat{R}^-_{\mathrm{u}}(g) - \pi\widehat{R}^-_{\mathrm{p}}(g)$. Kiryo et al. (2017) observe that highly expressive models (e.g., neural networks) can overfit $\mathcal{X}_{\mathrm{p}}$ causing uPU to estimate that $\widehat{R}^-_{\mathrm{u}}(g) - \pi\widehat{R}^-_{\mathrm{p}}(g) < 0$. Since negative-valued risk is impossible, Kiryo et al.'s non-negative PU (nnPU) risk estimator ignores negative risk estimates via a $\max$ term:

$$\widehat{R}_{\mathrm{nnPU}}(g) \coloneqq \pi\widehat{R}^+_{\mathrm{p}}(g) + \max\{0, \widehat{R}^-_{\mathrm{u}}(g) - \pi\widehat{R}^-_{\mathrm{p}}(g)\}. \tag{2}$$

nnPU requires a custom ERM algorithm with specialized "defitting" to correct negative-valued risk estimates.

3. Absolute Value Overfitting Correction

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

$$(1-\pi)\ddot{R}^{\hat{y}}_{\mathrm{n}}(g) \coloneqq \left|\widehat{R}^{\hat{y}}_{\mathrm{u}}(g) - \pi\widehat{R}^{\hat{y}}_{\mathrm{p}}(g)\right|, \tag{3}$$

replaces nnPU's $\max$ with absolute value to prevent the optimizer overfitting an implausible risk estimate by explicitly penalizing for $\widehat{R}^{\hat{y}}_{\mathrm{u}}(g) - \pi\widehat{R}^{\hat{y}}_{\mathrm{p}}(g) < 0$. This penalty "defits" the learner automatically eliminating the need for nnPU's custom ERM algorithm.

Our *abs-PU risk estimator* leverages abs. value correction:

$$\widehat{R}_{\text{abs-PU}}(g) \coloneqq \pi\widehat{R}^+_{\mathrm{p}}(g) + \left|\widehat{R}^-_{\mathrm{u}}(g) - \pi\widehat{R}^-_{\mathrm{p}}(g)\right|. \tag{4}$$

By Theorem 2, abs-PU is statistically *consistent* like nnPU. Empirically we saw that abs-PU yields models of similar or slightly better accuracy than nnPU but with simpler optimization. The following builds on abs-PU with a full empirical comparison to nnPU in suppl. Section G.5.

4. Arbitrary-Positive, Unlabeled Learning

Arbitrary-positive unlabeled (aPU) learning – the focus of this work – considers two joint distributions: train $p_{\mathrm{tr}}(x, y)$ and test $p_{\mathrm{te}}(x, y)$. Notation $p_{\text{tr-}\mathcal{D}}(x)$ where $\mathcal{D} \in \{\mathrm{p}, \mathrm{n}, \mathrm{u}\}$ 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.

Nothing is assumed about the posteriors, i.e., $p_{\mathcal{T}}(y|x)$, ($\mathcal{T} \in \{\mathrm{tr}, \mathrm{te}\}$) nor about the positive class-conditionals $p_{\mathcal{T}\text{-p}}(x)$. We only assume a fixed negative class-conditional,

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

The train and test positive-class priors, π_{tr} and π_{te} respectively, are treated as known throughout this work.

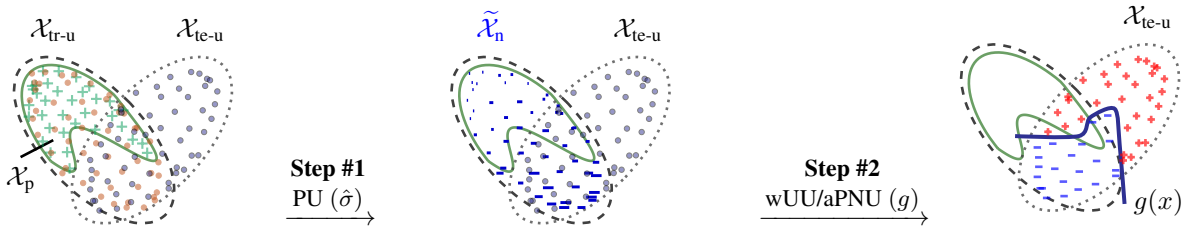
Fig. 1a shows the available aPU datasets: positive set $\mathcal{X}_{\mathrm{p}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-p}}(x)$ and unlabeled sets $\mathcal{X}_{\text{tr-u}} \coloneqq \{x_i\}_{i=1}^{n_{\text{tr-u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)$ and $\mathcal{X}_{\text{te-u}} \coloneqq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

Relating aPU and Covariate Shift Adaptation Methods

Covariate shift (Shimodaira, 2000) is a common technique to address differences between $p_{\mathrm{tr}}(x, y)$ and $p_{\mathrm{te}}(x, y)$. The *importance function* is $w(x) \coloneqq \frac{p_{\text{te-u}}(x)}{p_{\text{tr-u}}(x)}$. Under covariate shift's *consistent input-output relation* assumption, i.e., $p_{\mathrm{tr}}(y|x) = p_{\mathrm{te}}(y|x)$, it follows that $w(x)p_{\mathrm{tr}}(x, y) = p_{\mathrm{te}}(x, y)$.

Sakai & Shimizu (2019) exploit this relationship in their *PUc risk estimator*. $w(x)$ is approximated via direct density-ratio estimation over $\mathcal{X}_{\text{tr-u}}$ and $\mathcal{X}_{\text{te-u}}$. PUc couples $w(x)$ with uPU and serves as this work's primary baseline.

[1] A nomenclature reference appears in supplemental Section A.



(a) Example aPU dataset (b) Weighting $\mathcal{X}_{\text{tr-u}}$ by $\hat{\sigma}$ yields $\widetilde{\mathcal{X}}_{\text{n}}$ (c) Final classifier g

Figure 1. A toy aPU dataset is shown in 1a with (+) representing a labeled positive example, (•) an unlabeled train sample, and (•) an unlabeled test sample. Borders surround each set for clarity. In two-step aPU learning, step #1 trains probabilistic classifier $\hat{\sigma}$. 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.

5. aPU Learning via UU Learning

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

A perfect classifier is unrealistic. Our key insight is that weighting $\mathcal{X}_{\text{tr-u}}$ by $p_{\text{tr}}(Y=-1|x)$ transforms $\mathcal{X}_{\text{tr-u}}$ into a representative negative set. We then propose two methods to fit 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. Figures 1b and 1c visualize our two-step method, with a formal presentation in Algorithm 1 and a detailed description below.

Step #1: Create a Representative Negative Set

This step's goal is to learn the training distribution's negative class-posterior, $p_{\text{tr}}(Y=-1|x)$. We achieve this by training PU probabilistic classifier $\hat{\sigma}:\mathbb{R}^d\to[0,1]$ using $\mathcal{X}_{\text{p}}$ and $\mathcal{X}_{\text{tr-u}}$. In principle, any probabilistic PU method can be used; we focused on ERM-based PU methods with the logistic loss as ℓ and sigmoid output activation.

Theorem 3 shows that weighting $\mathcal{X}_{\text{tr-u}}$ by $\hat{\sigma}$ yields a consistent *surrogate negative set* $\widetilde{\mathcal{X}}_{\text{n}}$ that estimates the negative labeled risk $R^{\hat{y}}_{\text{n}}(g)$; we denote this estimator $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$.

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

Negative-unlabeled (NU) learning is functionally the same as PU learning. Sakai et al. (2017) formalize an unbiased NU risk estimator, $\widehat{R}_{\text{NU}}(g) \coloneqq \left|\widehat{R}^{+}_{\text{u}}(g) - (1-\pi)\widehat{R}^{+}_{\text{n}}(g)\right| + (1-\pi)\widehat{R}^{-}_{\text{n}}(g)$ (defined here with our absolute-value-correction). Our *weighted-unlabeled, unlabeled* (wUU) estimator,

$$\widehat{R}_{\text{wUU}}(g) \coloneqq \left|\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g)\right| + (1-\pi_{\text{te}})\widetilde{R}^{-}_{\text{n-u}}(g), \quad (6)$$

Algorithm 1 Two-step unlabeled-unlabeled aPU learning

Input: Data $(\mathcal{X}_{\text{p}}, \mathcal{X}_{\text{tr-u}}, \mathcal{X}_{\text{te-u}})$

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

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 $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ is consistent, wUU is also consistent.

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. 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. (2017) propose the PNU risk estimator, $\widehat{R}_{\text{PNU}}(g) \coloneqq (1-\rho)\widehat{R}_{\text{PN}}(g) + \rho\widehat{R}_{\text{NU}}(g)$, where hyperparameter $\rho \in (0,1)$ weights the PN and NU risk estimators. Our *arbitrary-positive, negative, unlabeled* (aPNU) risk estimator below modifies PNU to use $\widetilde{\mathcal{X}}_{\text{n}}$ and absolute-value correction.

$$\begin{aligned}\widehat{R}_{\text{aPNU}}(g) \coloneqq{}& (1-\rho)\pi_{\text{te}}\widehat{R}^{+}_{\text{p}}(g) + (1-\pi_{\text{te}})\widetilde{R}^{-}_{\text{n-u}}(g)\\ &+ \rho\left|\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g)\right|\end{aligned} \quad (7)$$

A midpoint value of $\rho = 0.5$ empirically performed well when no knowledge about the positive shift is assumed.

6. PU Recursive Risk Estimation

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

Our Positive-Unlabeled Recursive Risk (*PURR*) estimator learns our aPU setting via a single, *joint* method. Due to limited space, PURR is described in supplemental Section B.

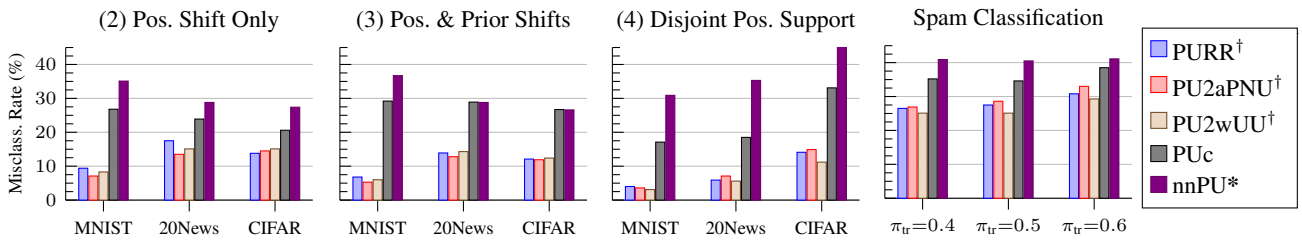


Figure 2. Mean *inductive* misclassification rate over 100 trials on the MNIST, 20 Newsgroups, CIFAR10, & TREC datasets for our three methods – PURR, PU2aPNU, & PU2wUU which are denoted with † – and the baselines. Each numbered plot (2–4) corresponds to one *shift task* in Section 7.2 and Table 2. The TREC spam email classification experiments are detailed in Section 7.3 and Table 3.

7. Experimental Results

Limited space allows us to discuss only two experiment sets here; Section G details additional experiments. The primary performance metric is inductive misclassification rate.

7.1. Experimental Setup

Supplemental Section F details the complete experimental setup[2] which we very briefly summarize below.

Baselines PUc (Sakai & Shimizu, 2019) with a linear-in-parameter model and Gaussian kernel basis is the primary baseline. Ordinary nnPU is the performance floor. To ensure the strongest baseline, we separately trained nnPU using $\mathcal{X}_{\text{te-u}}$ as well as with the combined $\mathcal{X}_{\text{tr-u}} \cup \mathcal{X}_{\text{te-u}}$ and report each experiment's best performing configuration, denoted nnPU*. PN-test (trained on labeled $\mathcal{X}_{\text{te-u}}$) is a reference for the performance ceiling.

Datasets Section 7.2 considers the MNIST, CIFAR10, and 20 Newsgroups datasets with binary classes formed by partitioning the dataset labels. Section 7.3 relates two TREC spam email datasets to replicate real-world adversarial concept drift.

Learner Architecture We trained neural networks (NNs) via stochastic optimization, i.e., AdamW (Loshchilov & Hutter, 2017). 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-in-parameter models with Gaussian kernels, we limited our NNs to at most three fully-connected layers of 300 neurons. For MNIST, we trained NNs from scratch. Transfer learning via pretrained deep networks was used to encode the CIFAR10, 20 Newsgroups, and TREC datasets into static representations used by all learners (see Sections F.4, F.5, & F.6 resp.).

Hyperparameters Our only individually tuned hyperparameters are learning rate and weight decay. For aPNU, $\rho = 0.5$. PUc's hyperparameters were tuned via importance-weighted cross validation (Sugiyama et al., 2007).

[2]Source code: https://github.com/ZaydH/udl_arbitrary_pu.

7.2. Partially and Fully Disjoint Positive Supports

Here we replicate scenarios where positive subclasses exist only in the test distribution (e.g., zero-day attacks). Suppl. Table 2 details the positive train/test and negative class definitions. Each dataset has four experimental conditions (ordered by row number): (1) $\text{P}_{\text{train}} = \text{P}_{\text{test}}$, i.e., no bias, (2 & 3 resp.) partially disjoint positive supports without and with prior shift, and (4) disjoint positive class definitions.

Results are shown in Figure 2 and suppl. Table 2. When there was a positive-shift, our three methods always outperformed PUc and nnPU* according to a 1% paired t-test. Our advantage over PUc is not attributable to our use of NNs as PUc generally outperformed nnPU* by a wide margin.

For partially disjoint positive supports (Table 2 rows 2 & 3 for each dataset), PU2aPNU was the top performer for five of six setups. This pattern reversed for fully disjoint supports (row 4) where PU2aPNU always lagged PU2wUU.

7.3. Case Study: Arbitrary Adversarial Concept Drift

PU learning has been applied to multiple adversarial domains including opinion spam (Hernández Fusilier et al., 2013; Zhang et al., 2019). 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 datasets' ham emails are also quite different: TREC05 relies on Enron emails while TREC07 is mostly emails from a university server. Thus, this represents a more challenging, realistic setting where Eq. (5)'s assumption does not hold.

Figure 2 and suppl. Table 3 show our methods outperformed PUc and nnPU* according a 1% paired t-test across three training priors. PU2wUU is the top-performer as $\hat{\sigma}$ accurately labels $\mathcal{X}_{\text{tr-u}}$, yielding a strong surrogate negative set.

Overall, this shows that our aPU setting arises in real-world domains. We handle large positive shifts better than prior work, even in realistic cases of a shifting negative class.

Acknowledgments

This research is partially supported by a grant from the Air Force Research Laboratory and Defense Advanced Research Projects Agency, under agreement number FA8750-16-C-0166, subcontract K001892-00-S05.

References

Alaiz-Rodríguez, R. and Japkowicz, N. Assessing the impact of changing environments on classifier performance. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 13–24. Springer, 2008.

Bekker, J. and Davis, J. Estimating the class prior in positive and unlabeled data through decision tree induction. In *AAAI Conference on Artificial Intelligence*, 2018.

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

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

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

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

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

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

Hernández Fusilier, D., Guzmán Cabrera, R., Montes-y Gómez, M., and Rosso, P. Using PU-learning to detect deceptive opinion spam. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pp. 38–45, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

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

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

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

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

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

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

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

Li, H., Chen, Z., Liu, B., Wei, X., and Shao, J. Spotting fake reviews via collective positive-unlabeled learning. In *Proceedings of the 2014 IEEE International Conference on Data Mining*, ICDM'14, pp. 899–904, USA, 2014. IEEE Computer Society. ISBN 9781479943029.

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

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

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

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

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

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

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

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

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

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

TREC. Text REtrieval Conference (TREC) overview. https://trec.nist.gov/overview.html, 2019.

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

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

Zhang, Y.-L., Li, L., Zhou, J., Li, X., Liu, Y., Zhang, Y., and Zhou, Z.-H. 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, pp. 2599–2601, New York, NY, USA, 2017. Association for Computing Machinery.

Positive-Unlabeled Learning with Arbitrarily Non-Representative Labeled Data

Supplemental Materials

A. Nomenclature

Table 1: 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 (du Plessis et al., 2014). See Section 2
nnPU	Non-negative Positive-Unlabeled risk estimator from (Kiryo et al., 2017). 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 (Bekker et al., 2019; Kato et al., 2019) and covariate shift methods (Sakai & Shimizu, 2019)
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 (Sakai & Shimizu, 2019). See Section 4
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 G.5.2
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, $\pi_{\mathcal{T}} \coloneqq p_{\mathcal{T}}(Y = +1)$ where $\mathcal{T} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{T}\text{-p}}(x)$	Positive class-conditional $p_{\mathcal{T}\text{-p}}(x) \coloneqq p_{\mathcal{T}}(x \mid Y = +1)$ where $\mathcal{T} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{T}\text{-n}}(x)$	Negative class-conditional $p_{\mathcal{T}\text{-n}}(x) \coloneqq p_{\mathcal{T}}(x \mid Y = -1)$ where $\mathcal{T} \in \{\text{tr}, \text{te}\}$ for train & test resp.
$p_{\mathcal{T}\text{-u}}(x)$	Marginal distribution where $p_{\mathcal{T}\text{-u}}(x) \coloneqq p_{\mathcal{T}}(x)$ 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 $\hat{\sigma}$

Table 1: aPU nomenclature reference (continued)

$\mathcal{X}_{\text{n}}$	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., $R(g) \coloneqq \mathbb{E}_{(X,Y)\sim p(x,y)}[\ell(Yg(X))]$
$\widehat{R}(g)$	Empirical estimate of risk $R(g)$
$\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$	Empirical risk when predicting label $\hat{y} \in \{\pm 1\}$ on data sampled from some distribution, $p_{\mathcal{D}}(x)$. See Section 2
$\ddot{R}^{\hat{y}}_{\text{n}}(g)$	Labeled negative risk with absolute-value correction. See Eq. (3) in Section 3
$\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$	Surrogate negative risk formed by weighting unlabeled set $\mathcal{X}_{\text{tr-u}}$ by probabilistic classifier $\hat{\sigma}$ where $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g) \coloneqq \frac{1}{n_{\text{tr-u}}} \sum_{x_i \in \mathcal{X}_{\text{tr-u}}} \frac{\hat{\sigma}(x_i)\ell(\hat{y}g(x_i))}{1-\pi_{\text{tr}}}$
$w(x)$	Covariate shift importance function based on density-ratio estimation where $w(x) \coloneqq \frac{p_{\text{te-u}}(x)}{p_{\text{tr-u}}(x)}$
n_{p}	Size of the labeled (positive) dataset, i.e., $n_{\text{p}} \coloneqq \lvert\mathcal{X}_{\text{p}}\rvert$
$n_{\text{tr-u}}$	Size of the unlabeled *training* dataset, i.e., $n_{\text{tr-u}} \coloneqq \lvert\mathcal{X}_{\text{tr-u}}\rvert$
$n_{\text{te-u}}$	Size of the unlabeled *test* dataset, i.e., $n_{\text{te-u}} \coloneqq \lvert\mathcal{X}_{\text{te-u}}\rvert$
n_{Test}	Size of the inductive test set
$\mathcal{A}$	Learning or optimization algorithm
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 G.1
$[a]_+$	$\coloneqq \max\{0, a\}$. See Section E.2

B. Positive-Unlabeled Recursive Risk Estimation

Here we describe in detail our recursive risk estimation method discussed briefly in Section 6.

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 & 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, $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$ (where $\mathcal{D} \in \{\mathrm{p}, \mathrm{n}\}$), 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,

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

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

In standard PU and NU learning, either $\widehat{R}^{\hat{y}}_{A}(g)$ or $\widehat{R}^{\hat{y}}_{B}(g)$ can always be estimated 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:

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

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

First, for $\widehat{R}^{-}_{\text{te-n}}(g)$, we consider its more general form $\widehat{R}^{\hat{y}}_{\text{te-n}}(g)$ below since $\widehat{R}^{+}_{\text{te-n}}(g)$ will be needed as well. Using Eq. (5)'s assumption, $\widehat{R}^{\hat{y}}_{\text{te-n}}(g)$ can be estimated directly from the training distribution. Combining Eq. (1) with absolute-value correction, we see that

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

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

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

Eq. (10) with $\hat{y} = +1$ substitutes for $\widehat{R}^{+}_{\text{te-n}}(g)$ in Eq. (11) yielding $\widehat{R}_{\text{PURR}}(g)$'s complete definition:

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

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

Optimization Using non-negativity instead of absolute-value correction significantly complicates PURR's optimization scheme by necessitating the consideration of four candidate gradients per update.[3] In contrast, PURR with absolute-value correction integrates into a standard ERM framework.

[3] A complete discussion of PURR's ERM algorithm with non-negativity correction is in suppl. Section E.2.

C. Experimental Results Table

This section contains the experimental results tables referenced in Section 7.

Table 2. Inductive misclassification rate mean and standard deviation 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 methods' performance was statistically better than 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. Our three methods – PURR, PU2aPNU, and PU2wUU – are denoted with †. Additional shifted tasks are in supplemental Section G.2.

	N	P_{test}	P_{train}	π_{tr}	π_{te}		Two-Step (PU2)		Baselines		Ref.
						PURR†	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)	↑
			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)	2.8 (0.2)
				0.29	0.5	6.8 (0.8)	**5.3 (0.6)**	6.0 (0.7)	29.2 (2.1)	36.7 (2.7)	↓
	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 News.	sci, soc, talk	alt, comp, misc, rec	P_{test}	0.56	0.56	15.4 (1.3)	14.9 (1.2)	16.7 (2.3)	14.9 (1.0)	14.1 (0.8)	↑
			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)	10.5 (0.4)
				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	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 (0.9)	14.2 (1.3)	15.5 (1.6)	13.8 (0.8)	12.3 (0.6)	↑
			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)	9.8 (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)	↓
	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)

Table 3. Inductive misclassification rate mean and standard deviation for the TREC spam email adversarial drift over 100 trials. Our three methods – PURR, PU2aPNU, and PU2wUU (denoted with †) – all outperformed PUc & nnPU* based on a 1% paired t-test across all three 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)	↓

D. Theorems and Proofs

D.1. Proof of Theorem 2

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 then $\ddot{R}^{\hat{y}}_{\mathrm{n}}(g)$ is a consistent estimator of $\widehat{R}^{\hat{y}}_{\mathrm{n}}(g)$.*

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

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

du Plessis et al. (2014) show that

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

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

$$\ddot{R}^{\hat{y}}_{\mathrm{n}}(g) = \left|\widehat{R}^{\hat{y}}_{\mathrm{n}}(g)\right|. \tag{14}$$

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 $\ddot{R}^{\hat{y}}_{\mathrm{n}}(g)$ in Eq. (14), it is sufficient to show that each of its individual terms is consistent.

Both $\widehat{R}^{\hat{y}}_{\mathrm{p}}(g)$ and $\widehat{R}^{\hat{y}}_{\mathrm{u}}(g)$ 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 $R^{\hat{y}}_{\mathcal{D}}(g)$. In addition, $\ell(\hat{y}g(X)) < C_\ell < \infty$ implies that $\mathrm{Var}(\ell(\hat{y}g(X))) < \infty$. By Chebyshev's Inequality, $\widehat{R}^{\hat{y}}_{\mathcal{D}}(g)$ is consistent as

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

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

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

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

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

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

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

decreases as

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

so

$$\begin{aligned}
\text{err}_{\ddot{R}} &= \left|\left|\widehat{R}^{\hat{y}}_{\text{n}}(g)\right| - R^{\hat{y}}_{\text{n}}(g)\right| \\
&=: \left|\ddot{R}^{\hat{y}}_{\text{n}}(g) - R^{\hat{y}}_{\text{n}}(g)\right| \\
&< \ddot{R}^{\hat{y}}_{\text{n}}(g) + R^{\hat{y}}_{\text{n}}(g) < \epsilon && \text{Since } \widehat{R}^{\hat{y}}_{\text{n}}(g) < 0 \text{ and } R^{\hat{y}}_{\text{n}}(g) > 0. \qquad (15)
\end{aligned}$$

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

D.2. Proof of Theorem 3

Theorem 3. *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 $\mathcal{X}_{\text{tr-u}} \coloneqq \{x_i\}_{i=1}^{n_{\text{tr-u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{tr-u}}(x)$, and restrict $\pi_{\text{tr}} \in [0, 1)$. Define $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g) \coloneqq \frac{1}{n_{\text{tr-u}}} \sum_{x_i \in \mathcal{X}_{\text{tr-u}}} \frac{\hat{\sigma}(x_i)\ell(\hat{y}g(x_i))}{1-\pi_{\text{tr}}}$. Let $\hat{\sigma} : \mathbb{R}^d \to [0, 1]$ in hypothesis set $\widehat{\Sigma}$. When $\hat{\sigma}(x) = p_{\text{tr}}(Y = -1|x)$, $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ is an unbiased estimator of $R^{\hat{y}}_{\text{n}}(g)$. When the concept class of functions that defines $p_{\text{tr}}(Y = -1|x)$ is probably approximately correct (PAC) learnable by some PAC-learning algorithm $\mathcal{A}$ that selects $\hat{\sigma} \in \widehat{\Sigma}$, then $\widetilde{R}^{\hat{y}}_{\text{n-u}}(g)$ is a consistent estimator of $R^{\hat{y}}_{\text{n}}(g)$.*

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

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

satisfying the definition of unbiased.

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

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

[4] No restrictions are placed on $\mathcal{A}$ other than its existence and that selected hypothesis $\hat{\sigma}_{\mathcal{S}}$ satisfies Eq. (16).

there exists an n' such that for all $n > n'$,

$$\Pr\left[\mathbb{E}_{X\sim p_{\text{tr-u}}(x)}[|\hat{\sigma}_{\mathcal{S}}(X) - p_{\text{tr}}(Y=-1|X)|] > \epsilon\right] < \delta. \tag{16}$$

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

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

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

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

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

□

D.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 $\mathcal{X}_{\text{u}} \coloneqq \{x_i\}_{i=1}^{n_{\text{u}}} \overset{\text{i.i.d.}}{\sim} p_{\text{u}}(x)$. 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. □

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 3's proof, sample inclusion in the negative set is a *hard* "in-or-out" decision. This does not change the sample's statistical properties, but it allows us to create an unweighted negative set, we denote $\mathcal{X}_{\text{tr-n}}$.

By Eq. (5)'s assumption, $\mathcal{X}_{\text{tr-n}}$ is representative of samples from the negative 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 (Ramaswamy et al., 2016; du Plessis et al., 2017) 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 G.7.

D.4. Proof of Theorem 1

The definition of "bounded loss" is identical to the proof of Theorem 2.

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

$$(1-\pi)R_{\text{n}}^{\hat{y}}(g) = R_{\text{u}}^{\hat{y}}(g) - \pi R_{\text{p}}^{\hat{y}}(g). \tag{17}$$

The positive labeled risk similarly decomposes as

$$\pi R_{\text{p}}^{\hat{y}}(g) = R_{\text{u}}^{\hat{y}}(g) - (1-\pi)R_{\text{n}}^{\hat{y}}(g). \tag{18}$$

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

$$\widehat{R}_{\text{uPURR}}(g) = \underbrace{\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\underbrace{\frac{\widehat{R}^{+}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{+}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}}_{\widehat{R}^{+}_{\text{te-n}}(g)}}_{\pi_{\text{te}}\widehat{R}^{+}_{\text{te-p}}(g)} + (1-\pi_{\text{te}})\underbrace{\frac{\widehat{R}^{-}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{-}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}}_{\widehat{R}^{-}_{\text{te-n}}(g)}}. \tag{19}$$

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. (17) and (18)) 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:

$$\widehat{R}_{\text{PURR}}(g) = \underbrace{\left|\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\left|\underbrace{\frac{\widehat{R}^{+}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{+}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}}_{\widehat{R}^{+}_{\text{te-n}}(g)}\right|\right|}_{\pi_{\text{te}}\widehat{R}^{+}_{\text{te-p}}(g)} + (1-\pi_{\text{te}})\left|\underbrace{\frac{\widehat{R}^{-}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{-}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}}_{\widehat{R}^{-}_{\text{te-n}}(g)}\right|. \tag{20}$$

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 2 showed that $\ddot{R}_{\text{n}}^{\hat{y}}(g)$ is consistent. Following the same logic in Theorem 2'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 2 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. (8)) were estimated directly from training data. By induction, it is again straightforward from Theorem 2 that any decomposed term (e.g., $R_{A}^{\hat{y}}(g)$ in Eq. (8)) formed from the sum of consistent estimators must be itself consistent.

Theorem 2 further demonstrated that applying absolute-value correction does not affect the consistency of a risk estimator's. Therefore, any recursive risk estimator with absolute-value correction is consistent. PURR's consistency is just a single, specific example of this general property. □

E. Non-Negativity Correction Empirical Risk Minimization Algorithms

Kiryo et al. (2017)'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.

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

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

$$\widehat{R}_{\text{nn-wUU}}(g) \coloneqq \max\left\{0, \widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g)\right\} + (1-\pi_{\text{te}})\widetilde{R}^{-}_{\text{n-u}}(g). \tag{21}$$

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

$$\widehat{R}_{\text{nn-aPNU}}(g) \coloneqq (1-\rho)\pi_{\text{te}}R^{+}_{\text{p}}(g) + (1-\pi_{\text{te}})\widetilde{R}^{-}_{\text{n-u}}(g) + \rho\max\left\{0, R^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g)\right\}. \tag{22}$$

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 $\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g) < 0$ (see line 7). The basic algorithm is heavily influenced by the stochastic optimization algorithm proposed by Kiryo et al. (2017).

Algorithm 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 $(\mathcal{X}^{(i)}_{\text{p}}, \widetilde{\mathcal{X}}^{(i)}_{\text{n}}, \mathcal{X}^{(i)}_{\text{te-u}})$ **do**
5: **if** $\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g) < 0$ **then**
6: Set gradient $-\nabla_\theta \left(\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\widetilde{R}^{+}_{\text{n-u}}(g)\right)$
7: Update 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.

E.2. PURR Non-Negativity ERM Algorithm

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

$$\widehat{R}_{\text{nn-PURR}}(g) \coloneqq \underbrace{\left[\widehat{R}^{+}_{\text{te-u}}(g) - (1-\pi_{\text{te}})\underbrace{\left[\frac{\widehat{R}^{+}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{+}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}\right]_+}_{\widehat{R}^{+}_{\text{te-n}}(g)}\right]_+}_{\pi_{\text{te}}\widehat{R}^{+}_{\text{te-p}}(g)} + (1-\pi_{\text{te}})\underbrace{\left[\frac{\widehat{R}^{-}_{\text{tr-u}}(g) - \pi_{\text{tr}}\widehat{R}^{-}_{\text{tr-p}}(g)}{1-\pi_{\text{tr}}}\right]_+}_{\widehat{R}^{-}_{\text{te-n}}(g)}. \tag{23}$$

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 $\widehat{R}_A^{\ell}(g)$'s gradient (see Eq. (8)). 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

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

F. Detailed Experimental Setup

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

F.1. Reproducing our Experiments

Our implementation is written and tested in Python 3.6.5 and 3.7.1 using the `PyTorch` neural network framework versions 1.3.1 and 1.4. The source code is available at: https://github.com/ZaydH/udl_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, G.1, and G.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.

F.2. Class Definitions

F.2.1. Partially and Fully Disjoint Positive Distribution Supports

Section 7.2's experimental setups are very similar to Hsieh et al. (2019)'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.[5]

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

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).

F.2.2. TREC SPAM CLASSIFICATION

As noted previously, PU learning has been applied to multiple adversarial domains including opinion spam (Hernández Fusilier et al., 2013; Li et al., 2014; Zhang et al., 2017; Zhang et al., 2019). 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. (5)) 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 Text REtrieval Conference (TREC) is organized annually be the United States' National Institute of Standards and Technology (NIST) to support information retrieval research (TREC, 2019). 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[6] while the test set was the TREC 2007 (TREC07) email dataset[7]. 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 F.6), we consider the first 10,000 emails from each dataset as defined by the dataset's `full/index` file.

F.2.3. IDENTICAL POSITIVE SUPPORTS WITH BIAS

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

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

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

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

Table 4. TREC05 & TREC07 dataset statistics

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

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

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

Table 7 lists the default size of each dataset's positive ($\mathcal{X}_{\text{p}}$), 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 G.4) used the dataset sizes defined by Sakai & Shimizu (2019). The separate validation set 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 2) 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, G.1, and G.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 & Shimizu (2019) do not specify whether the test set was held out in their experiments. When applicable, we merge the LIBSVM train and test datasets together as if there was only a single monolithic training set. $\mathcal{X}_{\text{p}}$, $\mathcal{X}_{\text{tr-u}}$, $\mathcal{X}_{\text{te-u}}$ and the inductive test set are independently sampled at random from this monolithic set without replacement.

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

F.4. CIFAR10 Image Representation

Each CIFAR10 (Krizhevsky et al., 2014) image is 32 pixels by 32 pixels with three (RGB) color channels (3,072 dimensions total). PUc specifies a convex model so it cannot be used to train (non-convex) deep convolutional networks directly. To ensure a meaningful comparison, we leveraged the DenseNet-121 deep convolutional network architecture pretrained on 1.2 million images from ImageNet (Huang et al., 2017). The network's (linear) classification layer was removed, and the experiments used the 1,024-dimension feature vector output by DenseNet's convolutional backbone.

F.5. 20 Newsgroups Document Representation

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

Classic word embedding models like GloVe and Word2Vec yield token representations that are independent of context. Proposed by Peters et al. (2018), ELMo (embeddings for language models) enhances classic word embeddings by making

Table 6. Positive & negative class definitions for the LIBSVM datasets in Section G.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

Table 7. Each dataset's default training set sizes. LIBSVM denotes all datasets downloaded directly from (Chang & Lin, 2011) and used in Section G.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

the token representations context dependent. We use ELMo to encode each 20 Newsgroup document as described below.

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

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

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

When documents are encoded serially, each document implicitly contains information about all 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.

F.6. TREC Email Representation

The TREC05 and TREC07 emails are encoded using the ELMo embedder identical to 20 Newsgroups. See Section F.5 above for the details.

F.7. Models and Hyperparameters

This section reviews the experiments' hyperparameter methodology.

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

As specified by its authors, PUc's hyperparameters were tuned via importance-weighted cross validation (IWCV) (Sugiyama et al., 2007). 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 (Ioffe & Szegedy, 2015). 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 $\gamma \in \{0.1, 0.5, 1.0\}$[8].

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.

[8]Hyperparameter γ only applies when using Kiryo et al. (2017)'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	

Table 10. Dataset-specific hyperparameter settings for our aPU learners. Hyperparameter $\gamma*$ only applies when using Kiryo et al. (2017)'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

Table 11. Dataset-specific hyperparameter settings for nnPU.

Dataset	$\text{nnPU}_{\text{te}\cup\text{tr}}$			nnPU_{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

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

Dataset	PN_{te}		PN_{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

G. 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 G.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.

G.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

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

$\pi_{\text{te}} = \pi_{\text{tr}} = 0.5$ makes the ideal *test* decision boundary $x_1 = 0$. 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

$$p_{\text{tr-(a)-p}}(x) = \frac{1}{2}\mathcal{N}(\begin{bmatrix}6 & -1\end{bmatrix}, \mathbf{I}_2) + \frac{1}{2}\mathcal{N}(\begin{bmatrix}6 & 1\end{bmatrix}, \mathbf{I}_2), \tag{24}$$

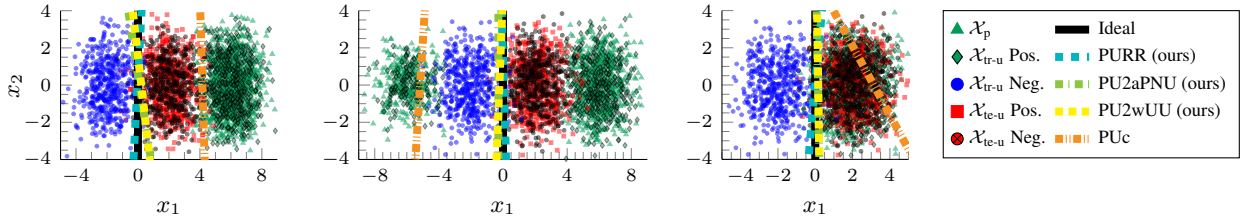
making the training distribution's optimal separator linear. PUc performed poorly on this setup for two reasons: covariate shift's assumption $p_{\text{tr}}(y|x) = p_{\text{te}}(y|x)$ does not hold, and the positive-train supports are functionally disjoint so importance function $w(x)$ is practically 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

$$p_{\text{tr-(b)-p}}(x) = \frac{2}{3}p_{\text{tr-(a)-p}}(x) + \frac{1}{3}\mathcal{N}(\begin{bmatrix}-6 & 0\end{bmatrix}, \mathbf{I}_2), \tag{25}$$

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: $x_1 = 0$.

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

Table 13 is an expanded version of Section 7.2's Table 2. 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}		Two-Step (PU2)		Baselines		Ref.
						PURR†	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)	↑
			7, 9	0.29	0.5	6.8 (0.8)	**5.3 (0.6)**	6.0 (0.7)	29.2 (2.1)	36.7 (2.7)	↑
				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)
			1, 3, 5	0.38	0.5	8.1 (1.0)	**6.5 (0.8)**	7.6 (0.9)	20.2 (2.5)	25.9 (1.1)	↓
				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)	↑
			misc, rec	0.37	0.56	13.9 (0.7)	**12.8 (0.6)**	14.3 (0.9)	28.9 (1.8)	28.8 (1.3)	↑
				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)
			comp	0.37	0.56	**13.3 (0.6)**	13.7 (0.6)	14.4 (0.7)	30.3 (2.0)	31.4 (0.7)	↓
				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)	↑
			Plane	0.14	0.4	12.1 (0.7)	**11.9 (0.7)**	12.4 (0.9)	26.7 (1.4)	26.7 (1.0)	↑
				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)
			Auto, Truck	0.25	0.4	12.7 (0.7)	**12.4 (0.7)**	12.8 (0.8)	19.2 (1.1)	20.3 (0.8)	↓
				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)

G.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 F.2.2, the training set was composed of spam and ham emails from the TREC05 dataset; the test set was composed of spam and ham emails from the TREC07 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. (5) does not hold.

As described in Section 5, the two-step methods' first step transforms $\mathcal{X}_{\text{tr-u}}$ into a representative negative set via probabilistic classifier $\hat{\sigma}$. Our standard approach uses $\hat{\sigma}$ to weight the samples. In this section's experiment, we instead follow a *top-k* approach. Recall that $\mathcal{X}_{\text{tr-u}}$ contains $n_{\text{tr-u}}$ examples. After training $\hat{\sigma}$, the $\pi_{\text{tr}} \cdot n_{\text{tr-u}}$ examples in $\mathcal{X}_{\text{tr-u}}$ with the highest predicted posteriors (according to $\hat{\sigma}$) are marked as positive-valued which the remaining $(1 - \pi_{\text{tr}}) \cdot n_{\text{tr-u}}$ examples are treated as negative-valued. This top-k approach was needed due to the propensity of $\hat{\sigma}$'s neural network to overfit the rich, high-dimensional ELMo representations. The two-step learners' second step (e.g., wUU and aPNU) was otherwise unchanged.

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 $\mathcal{X}_{\text{tr-u}}$. The resulting surrogate negative set is more useful than $\mathcal{X}_{\text{p}}$ to classify the spam emails from the test distribution. PU2aPNU performed slightly worse than PU2wUU because the spam emails in $\mathcal{X}_{\text{p}}$ are of very limited value due to the significant adversarial concept drift.

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$_{\text{te}}$
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)	↓

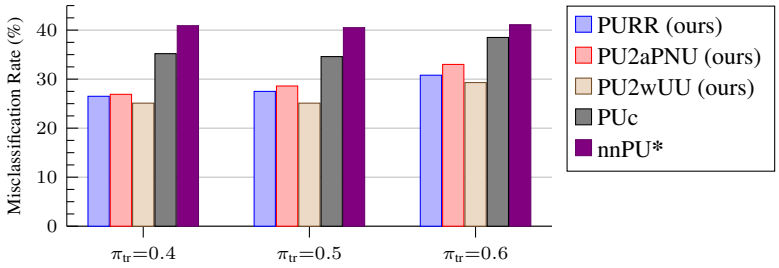


Figure 4. Mean inductive misclassification rate over 100 trials for the TREC spam datasets across three training priors (π_{tr}). Our PU2wUU method was the top performer across all experiments.

G.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 & Shimizu (2019)'s PUc paper. LIBSVM (Chang & Lin, 2011) benchmarks are used exclusively to ensure suitability with the SVM-like PUc; benchmarks "banana," "susy," "ijcnn1," and "a9a" appear in Sakai & Shimizu (2019)'s PUc paper.

Table 15. Inductive misclassification rate mean & standard deviation over 100 trials with Sakai & Shimizu (2019)'s median feature vector-based bias for 10 LIBSVM datasets. Underlining denotes a statistically significant performance improvement versus PUc and nnPU* according to a 1% paired t-test. Boldface indicates each dataset's best performing method. $n_{\text{p}} = 300$ and $n_{\text{tr-u}} = n_{\text{te-u}} = 700$. 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)

Sakai & Shimizu's bias operation is based on the median feature vector. Formally, given dataset $\mathcal{X} \subset \mathbb{R}^d$, define c_{med} as the median of set $\{\|x - \bar{x}\|_2 : x \in \mathcal{X}\}$ 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 $\mathcal{X}_{\text{lo}} \coloneqq \{x \in \mathcal{X} : \|x - \bar{x}\|_2 < c_{\text{med}}\}$ and $\mathcal{X}_{\text{hi}} \coloneqq \mathcal{X} \setminus \mathcal{X}_{\text{lo}}$. Examples in $\mathcal{X}_{\text{p}}$ and $\mathcal{X}_{\text{tr-u}}$ are selected from $\mathcal{X}_{\text{lo}}$ with probability $p = 0.9$ and from $\mathcal{X}_{\text{hi}}$ with probability $1 - p$. $p = 0.1$ is used when constructing $\mathcal{X}_{\text{te-u}}$ and the test set. This bias operation simplifies density-ratio estimation since $\forall_{x \in \mathcal{X}}\ w(x) \in \{\frac{1}{9}, 9\}$. Their setting $\pi_{\text{tr}} = \pi_{\text{te}} = 0.5$ also simplifies density estimation as detailed in Section G.2.

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

Analysis Table 15's experiments used the bias procedure described above. According to a 1% paired t-test, PURR and PU2aPNU outperformed PUc and nnPU* on all ten benchmarks; PU2wUU outperformed PUc and nnPU* 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.

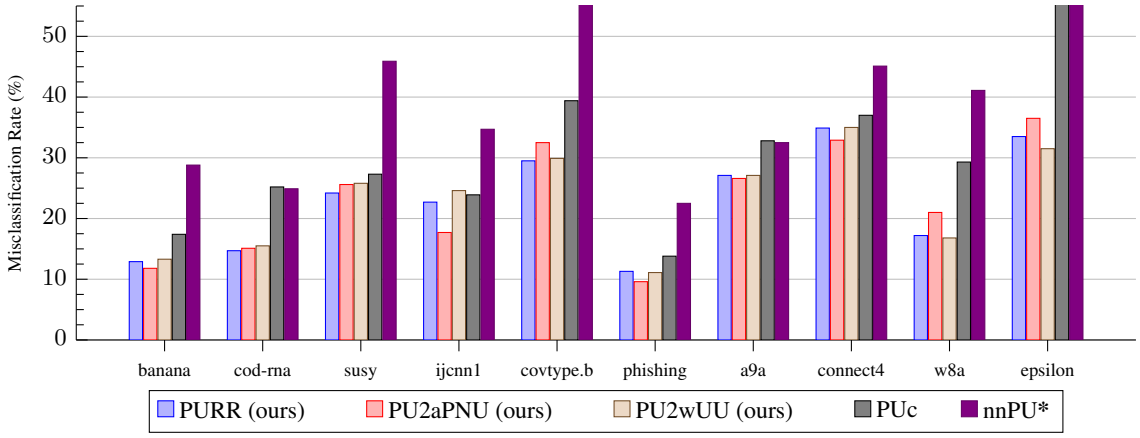


Figure 5. Mean inductive misclassification rate over 100 trials with Sakai & Shimizu (2019)'s median feature vector-based bias for the 10 LIBSVM datasets in Section G.4.

G.5. 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. (2017)'s non-negativity correction using $\max$ and "defitting."

G.5.1. Ordinary Positive-Unlabeled Learning Performance Without Distributional Shift

We first consider a direct comparison of nnPU and abs-PU on *unshifted* data. $\mathcal{X}_\text{p}$ and $\mathcal{X}_\text{u}$ 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 16 compares abs-PU and nnPU for the datasets in Sections 7.2[9], 7.3, and G.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 16. 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)

G.5.2. 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

[9]The test conditions for MNIST, 20 Newsgroups, and CIFAR10 correspond to the unbiased test conditions (i.e., row 1 for each dataset where $\text{P}_{\text{train}} = \text{P}_{\text{test}}$) in Table 2/Table 13.

set $\mathcal{X}_{\text{te-u}}$ and the other trained with unlabeled set $\mathcal{X}_{\text{tr-u}} \cup \mathcal{X}_{\text{te-u}}$ (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 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 17 compares abs-PU* and nnPU* for the extended set of experiments in Table 13 (see Section G.2). Recall that those experiments tested cases where some positive subclasses exist only in the test distribution. Similar to Table 16, 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 $\text{P}_{\text{train}}= \{1, 3, 5\}$), abs-PU* and nnPU* exhibited similar performance. When there was a large difference between the two methods (e.g., 20 Newsgroups $\text{P}_{\text{train}}= \{\text{misc}, \text{rec}\}$), 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 18 compares abs-PU* and nnPU* for the 10 LIBSVM datasets in Table 15 (see Section G.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 17. 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 18. 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 & Shimizu (2019)'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)

G.5.3. 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 E 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 hyperparameters used for the absolute-value based methods. Therefore, these results maximally favor the baseline of non-negativity correction.

Table 19's experiments are identical to Table 13 in Section G.2. "abs" denotes our standard aPU learning methods (see Sections 5 and 6) while "nn" denotes our methods modified to use Kiryo et al. (2017)'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 19. 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 News.	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
		Plane	0.14	0.4	**11.9**	12.0	0.1	**11.9**	12.0	0.1	**12.4**	**12.4**	0
			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 20's experiments match the experimental conditions for the 10 LIBSVM datasets in Table 15 from Section G.4. Biasing follows Sakai & Shimizu (2019)'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 20. Comparison of inductive misclassification rate mean and standard deviation over 100 trials for Table 15's LIBSVM dataset experiments using Sakai & 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)

G.6. 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 21 enumerates each dataset's positive and negative class definitions; these definitions apply for both train and test. The dataset sizes are listed in Table 22; 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 21. 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 22. 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,

$$\Pr[\text{Label}_{\text{tr}}{=}C_2|Y=+1] = 1 - \Pr[\text{Label}_{\text{tr}}{=}C_1|Y=+1]. \tag{26}$$

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 6a, 6c, and 6e 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.

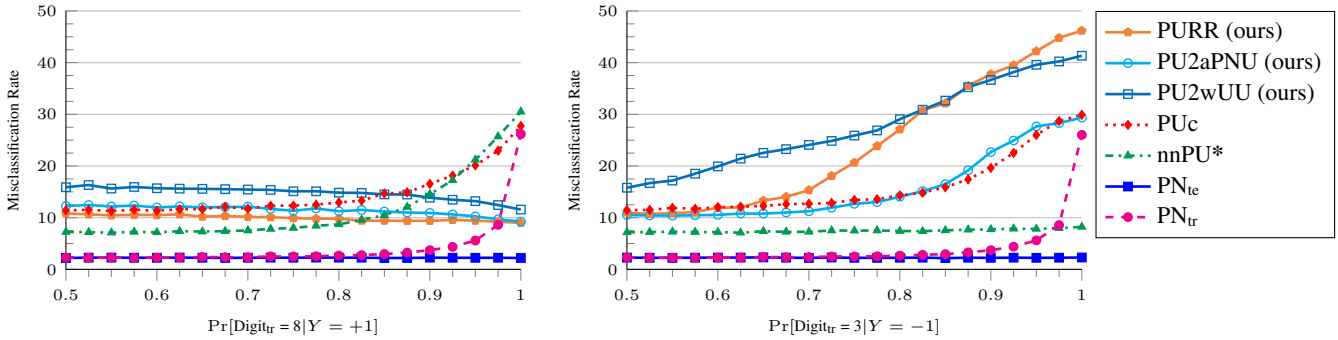
PN_{tr} and PN_{te} are trained using (labeled) $\mathcal{X}_{\text{tr-u}}$ and $\mathcal{X}_{\text{te-u}}$. Since the test distributions are never biased, PN_{te} is unaffected by shift. In contrast, as $\Pr[\text{Label}_{\text{tr}}{=}C_1|Y=+1]$ increases, there are fewer examples in $\mathcal{X}_{\text{tr-u}}$ 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.* (5)*'s assumption* upon which our methods are predicated. The goal here is to understand our methods' robustness under intentionally deleterious conditions. It is more deleterious to bias the negative class in $\mathcal{X}_{\text{tr-u}}$ since both two-step methods and PURR use $\mathcal{X}_{\text{tr-u}}$'s negative risk in dependent calculations; any error propagates and compounds in these subsequent operations.

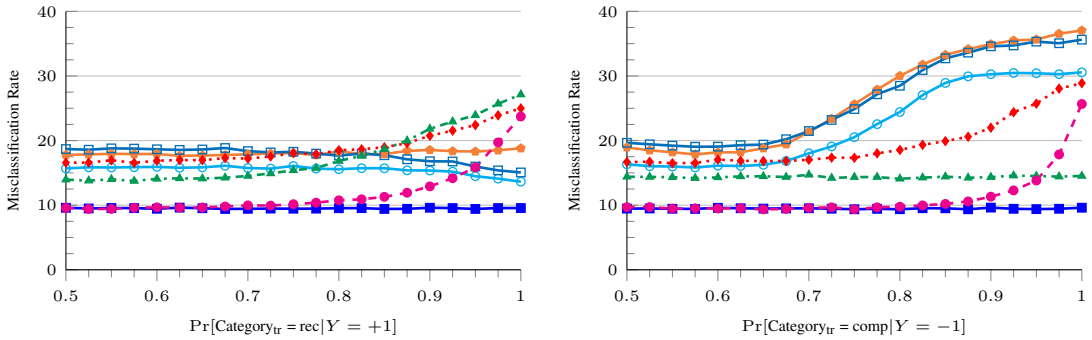
Let C_1 and C_2 now be the two labels that make up the negative class (e.g., $C_1 =$ Ship and $C_2 =$ Truck for CIFAR10). Now, $\Pr[\text{Label}_{\text{tr}} = C_1 | Y = -1]$ is swept along the x-axis from 0.5 to 1 (unbiased to maximally biased). The results for MNIST, 20 Newsgroups, and CIFAR10 are in Figures 6b, 6d, and 6f 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.



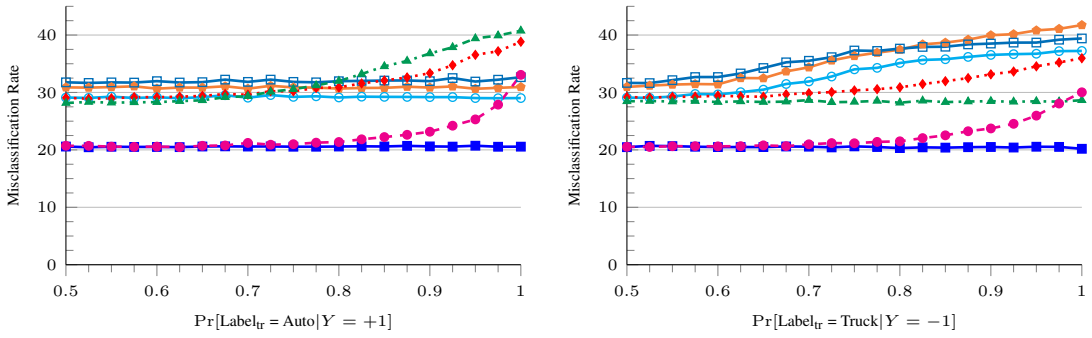
(a) MNIST positive train bias

(b) MNIST negative train bias



(c) 20 Newsgroups positive train bias

(d) 20 Newsgroups negative train bias



(e) CIFAR10 positive train bias

(f) CIFAR10 negative train bias

Figure 6. 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.

G.7. Effect of Prior Probability Estimation Error

As explained in Section 4, this work assumes that positive-class priors, π_{tr} and π_{te}, are known. The goal of these experiments is to study our methods' performance when the priors are misspecified.

Experimental Setup These experiments reuse the partially disjoint positive-support experiment setups from Section 7.2's Table 2. Therefore, we are specifically considering the MNIST, 20 Newsgroups, and CIFAR10 datasets with Table 23 summarizing all setups.

π_{tr} and π_{te} in Table 23 are the *actual* prior probabilities used to construct each training and test data set. We tested our methods' performance when each prior was specified correctly and when each prior was misspecified by $\pm 20\%$ for a total of $9 = 3 \times 3$ conditions per learner. π_{tr} was only misspecified when training g. $\hat{\sigma}$ was always provided the correct prior; this decision was made due to constraints in the implementation of our code. It is not an algorithmic limitation. Like all previous experiments, performance was evaluated using the inductive misclassification rate.

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

Tables 24, 25, and 26 contain the results for MNIST, 20 Newsgroups, and CIFAR10 respectively. Each learner's results are presented in a 3×3 grid with π_{tr} changing row to row while π_{te} changes column to column. Each cell is shaded red, with a darker background denoting worse performance (i.e., a greater misclassification rate). Even under the worst-case bias where both π_{tr} and π_{te} were shifted, all of our methods outperformed PUc.

Similar to Section G.6's experiments, the MNIST results were most affected by bias. The 20 Newsgroups and CIFAR10 results were more immune due to the richer feature representations generated through transfer learning. In most cases, the worst performance was observed when π_{tr} and π_{te} saw opposite bias, e.g., π_{tr} was overestimated while π_{te} was underestimated or vice versa; these values appear in the upper-right or lower-left corners of each learner's 3×3 grid.

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

Table 23. 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 24. Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on MNIST's inductive misclassification rate. Each result is the average of 100 trials.

	PURR			aPNU			wUU			PUc
	$0.8\pi_{te}$	π_{te}	$1.2\pi_{te}$	$0.8\pi_{te}$	π_{te}	$1.2\pi_{te}$	$0.8\pi_{te}$	π_{te}	$1.2\pi_{te}$	
$0.8\pi_{tr}$	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
$1.2\pi_{tr}$	25.3	15.8	12.7	18.0	7.7	7.5	16.9	8.9	10.3	26.3

Table 25. Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on 20 Newsgroups's inductive misclassification rate. Each result is the average of 100 trials.

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

Table 26. Combined heat map and table showing the effect of incorrectly specified priors π_{tr} and π_{te} on CIFAR10's inductive misclassification rate. Each result is the average of 100 trials.

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