We thank the reviewers for their very constructive feedback! We are encouraged that most reviewers (R1,R2,R4)[1] found our paper to be well-written and that our experiments were thorough/extensive and reproducible. Space prohibits us addressing all points raised by the reviewers so below we focus on the most pressing points from each reviewer.

@R1,R2,R3,R4: We will use the final submission's extra allowed page to clarify all points discussed below.

@R2 **Is our absolute-value correction the same as Kiryo et al.'s approach?**: No. Our approach is different in two ways. First, during optimization, when $\widehat{R}_{\text{u}}^{-}(g) - \pi\widehat{R}_{\text{p}}^{-}(g) < 0$, Kiryo et al.'s update gradient is: $\gamma\nabla_\theta(-\widehat{R}_{\text{u}}^{-}(g) + \pi\widehat{R}_{\text{p}}^{-}(g))$. In contrast, our method's gradient is: $\nabla_\theta(\pi\widehat{R}_{\text{p}}^{+}(g) - \widehat{R}_{\text{u}}^{-}(g) + \pi\widehat{R}_{\text{p}}^{-}(g))$. Put simply, Kiryo et al. stop optimizing $\widehat{R}_{\text{p}}^{+}(g)$ whenever $\widehat{R}_{\text{u}}^{-}(g) - \pi\widehat{R}_{\text{p}}^{-}(g) < 0$ whereas we always optimize $\widehat{R}_{\text{p}}^{+}(g)$. This means that we spend comparatively more time optimizing the positive-labeled risk. Second, when estimating performance (i.e., risk) on validation data, our approach penalizes implausible negative risk estimates while Kiryo et al.'s method does not.

Empirically, we find that this "soft-constraint" approach to implausible negative risk yields comparable or better models (see supplemental Sec. E.5). We also show in the supplementals (e.g., Sec. C(.2) and Alg. 3) that all of our methods can work with the Kiryo et al. approach, albeit with a (much) more complex implementation.

@R4 **Assumption of same bias between positives in positive-labeled set $\mathcal{X}_{\text{p}}$ and training unlabeled set $\mathcal{X}_{\text{tr-u}}$**: In our problem setting, $\mathcal{X}_{\text{p}}$ is an unbiased sample of the positive examples in $\mathcal{X}_{\text{tr-u}}$, but $\mathcal{X}_{\text{p}}$ could be arbitrarily different from the positive examples in test unlabeled set $\mathcal{X}_{\text{te-u}}$. However, if $\mathcal{X}_{\text{p}}$ is not representative of the positives in $\mathcal{X}_{\text{tr-u}}$, then our two-step methods could still be adapted to handle this case by using any bPU algorithm in step 1 to create surrogate negative set $\widetilde{\mathcal{X}}_{\text{n}}$, and our wUU estimator would be used in step 2 as normal. Thank you for this suggestion; we will note this additional flexibility of our two-step methods in the paper.

@R2 **Why do our two-step methods sometimes outperform our joint approach (PURR)?**: Supplemental tables 13 and 15 detail many empirical setups where our PURR estimator outperforms both the baselines, PUc and nnPU*, and our two-step approach. Nonetheless, the somewhat counterintuitive finding that a joint aPU learning method is not consistently the top performer is itself interesting and supports PURR's inclusion in the paper.

As an intuition, note that PURR, with its three risk decompositions/corrections, is strictly harder to optimize than wUU, aPNU, and nnPU, which each have one correction. This can lead to worse accuracy compared to the two-step methods, especially on easier problems like MNIST, where each step can be solved accurately on its own. When the dimensionality is low and dataset sizes are small (see Sec. E.4), PURR is generally the top performer since PURR is able to extract more information by better combining the limited available data.

@R2 **How to empirically compare our methods to the baselines given the implementation differences**: For PUc, we used the original authors' implementation, which relies on kernel methods for density-ratio estimation. For baseline nnPU* as well as our methods (none of which require density estimation), we followed the standard of most recent PU learning work (including Kiryo et al. in their nnPU paper), which uses neural networks.

However, our experiments show that PUc's biggest limitation is not its representation: On *unshifted* data (Table 1 row 1 for each dataset), PUc's performance is close to nnPU* and slightly outperforms our methods. On *shifted* data (Tab. 1 rows 2 & 3 for each dataset), PUc's & nnPU*'s performance degrades while our methods' performance improves. Thus, data shift (and methods for handling it) are the biggest factor in performance. See also Sec. E.1 which shows simple aPU learning tasks that our methods can successfully learn while PUc cannot – specifically with identical models.

We will add a "Discussion" subsection to the paper's "Experimental Results" (Sec. 7) to include the additional context in our response to this question as well as to the previous question on our joint method PURR.

@R4 **Negative-Class Shift in Epidemiology Example**: We expect that negative-class shifts will be small relative to shifts in the positive class, both in epidemiology and many other domains. Our experiments on TREC spam emails & in Sec. E.6 show that our methods are still useful even when our assumption is violated and the negative class shifts some.

@R1 **Problem Variations**: Sakai & Shimizu discuss additional problem variations including PU learning with arbitrary-negative shift, which they show is trivially equivalent to standard PU learning using positive set $\mathcal{X}_{\text{p}}$ and test unlabeled set $\mathcal{X}_{\text{te-u}}$. If both classes shift, learning is impossible without additional datasets or assumptions (e.g., consistent input/output for covariate shift). This expanded discussion of problem variations will be added to the paper.

@R3 **Not Reproducible**: Supplemental Sec. D enumerates our complete experimental setup, and our submission included runnable source code to replicate all experiments. We respectfully believe our experiments are reproducible.

@R3 **Notation Used Before Defined?**: We will update the paper to clarify that line 70's notation is based on the distribution notation listed at the end of the preceding paragraph (lines 65–66).

[1] We denote each reviewer RX, where X is the corresponding reviewer ID in CMT. Each reviewer identifier is also color coded.