Positive-Unlabeled Learning with Non-Negative Risk Estimator

Ryuichi Kiryo, Gang Niu, Marthinus C. du Plessis, Masashi Sugiyama
(University of Tokyo)

13 April 2018

Presented by: Gregory P. Spell
Outline

1. Unbiased PU Learning
2. Non-Negative PU Learning
3. Experiments
Problem Setting

- Let \( X \in \mathbb{R}^d \) and \( Y \in \{\pm 1\} \) be input and output random variables with underlying joint density \( p(x, y) \).
- \( P \) and \( N \) marginals: \( p_p(x) = p(x|Y = +1), \ p_n(x) = p(x|Y = -1) \).
- Class-prior probabilities: \( \pi_p = p(Y = +1), \ \pi_n = 1 - \pi_p \).
  - Assumed known (other papers concern estimation from \( P \) and \( U \) data).
- Sample two sets independently:
  \[
  \mathcal{X}_p = \{x_i^p\}_{i=1}^{n_p} \sim p_p(x), \ \text{and} \ \mathcal{X}_u = \{x_i^u\}_{i=1}^{n_u} \sim p(x)
  \]  \( \quad (1) \)
- Seek to train a classifier from \( \mathcal{X}_p \) and \( \mathcal{X}_u \).
Risk Definition

- Let $g : \mathbb{R}^d \to \mathbb{R}$ and $\ell : \mathbb{R} \times \{\pm 1\} \to \mathbb{R}$ be a decision function and the loss function, respectively.
- Define (with analogous definition for $R_n^-(g)$):
  \[
  R_p^+(g) = \mathbb{E}_p[\ell(g(X), +1)], \text{ with } \mathbb{E}_p[\cdot] = \mathbb{E}_{X \sim p}[\cdot]
  \]
- Risk of $g$:
  \[
  R(g) = \mathbb{E}_{(X,Y) \sim p(x,y)}[\ell(g(X), Y)] = \pi_p R_p^+(g) + \pi_n R_n^-(g)
  \] (2)
- In PU learning, $\mathcal{X}_n$ is unavailable, so must approximate $R_n^-(g)$
Unbiased Risk Estimator

- Define:
  \[ R_p^-(g) = \mathbb{E}_p[\ell(g(X), -1)] \]  
  \[ R_u^-(g) = \mathbb{E}_{X \sim p(x)}[\ell(g(X), -1)] \]  

- Consider the transformation:
  \[ \pi_n p_n(x) = p(x) - \pi_p p_p(x) \rightarrow \pi_n R_n^-(g) = R_u^-(g) - \pi_p R_p^-(g) \]  

- Then we may approximate the risk as:
  \[ \hat{R}_{pu}(g) = \pi_p \hat{R}_p^+(g) - \pi_p \hat{R}_p^-(g) + \hat{R}_u^-(g) \]  
  where
  \[ \hat{R}_p^-(g) = \frac{1}{n_p} \sum_{i=1}^{n_p} \ell(g(x^p_i), -1) \]  
  \[ \hat{R}_p^+(g) = \frac{1}{n_p} \sum_{i=1}^{n_p} \ell(g(x^p_i), +1) \]  
  \[ \hat{R}_u^-(g) = \frac{1}{n_u} \sum_{i=1}^{n_u} \ell(g(x^u_i), -1) \]
Motivation

- Flexible models (e.g., neural networks) are susceptible to overfitting, leading to separation of positive and unlabeled data.
- This can lead to negative estimated risk, despite the fact that $R(g) \geq 0$ for any $g$.

(a) Plain linear model

(b) Multilayer perceptron (MLP)
Non-Negative Risk Estimator

- Modify the unbiased risk estimator to ensure risk never becomes negative
- Previous risk estimator (as model overfits):

$$\hat{R}_{pu}(g) = \pi_p \hat{R}_p^+(g) - \pi_p \hat{R}_p^-(g) + \hat{R}_u(g)$$

$$\rightarrow \pi_p \cdot 0 \quad \rightarrow \pi_p \cdot 1 \quad \rightarrow 0$$

- Propose constraint to non-negativity:

$$\tilde{R}_{pu}(g) = \pi_p \hat{R}_p^+(g) + \max\{0, \hat{R}_u^-(g) - \pi_p \hat{R}_p^-(g)\} \quad (7)$$
Loss Functions

- Default loss is zero-one loss: $\ell_{01}(t, y) = \frac{1 - \text{sign}(ty)}{2}$
  - Tough to optimize, so typically replaced by a surrogate loss
- In previous papers, the authors show loss function cases in which estimators are unbiased:
  - $\ell$ satisfies symmetric condition: $\ell(t, +1) + \ell(t, -1) = 1$, which requires non-convex loss, then $\hat{R}_{pu}(g)$ is non-convex in $g$
  - $\ell$ satisfies linear-odd condition: $\ell(t, +1) - \ell(t, -1) = -t$, and then convex $\ell$ yields $\hat{R}_{pu}(g)$ convex in $g$
- Since flexible models won’t give convex optimizations anyway, don’t need to try and impose convexity in the loss.
- Here, use the sigmoid loss: $\ell_{\text{sig}}(t, y) = \frac{1}{1 + \exp(ty)}$
  - Satisfies symmetric condition
Scaling via Mini-Batching

- Let \((\mathcal{X}_p^i, \mathcal{X}_u^i)\) denote \(i\)-th mini-batch
- Note that:

\[
\tilde{R}_{pu}(g) = \sum_{i=1}^{N} \frac{\pi_p}{n_p} \sum_{x \in \mathcal{X}_p^i} \ell(g(x), +1) + \max \left\{ 0, \sum_{i=1}^{N} \left( \frac{1}{n_u} \sum_{x \in \mathcal{X}_u^i} \ell(g(x), -1) - \frac{\pi_p}{n_p} \sum_{x \in \mathcal{X}_p^i} \ell(g(x), -1) \right) \right\}
\]

is not easy to minimize in parallel because of the max operator

- Let:

\[
\tilde{R}'_{pu}(g) = \sum_{i=1}^{N} \left[ \frac{\pi_p}{n_p} \sum_{x \in \mathcal{X}_p^i} \ell(g(x), +1) + \max \left\{ 0, \frac{1}{n_u} \sum_{x \in \mathcal{X}_u^i} \ell(g(x), -1) - \frac{\pi_p}{n_p} \sum_{x \in \mathcal{X}_p^i} \ell(g(x), -1) \right\} \right]
\]

- Note that \(\tilde{R}_{pu}(g) \leq \tilde{R}'_{pu}(g)\) and so we may minimize the upper bound in parallel fashion.
Theoretical Properties

- Though risk estimator is biased, it is consistent and bias decreases exponentially.

- Under certain conditions, $\text{MSE}(\tilde{R}_{pu}(g)) \leq \text{MSE}(\hat{R}_{pu}(g))$.
  - $\ell$ satisfies previous symmetric condition (which sigmoid loss does).
  - $n_u >> n_p$

- Estimation error $R(\tilde{g}_{pu}) - R(g^*)$, where $g^*$ is the true risk minimizer, vanishes as $n_p, n_u \to \infty$.
  - Furthermore, for linear-in-parameter models, this happens at rate $O_p\left(\frac{\pi_p}{\sqrt{n_p}} + \frac{1}{\sqrt{n_u}}\right)$.
Experiments

- Conduct experiments on MNIST, 20Newsgroups, CIFAR-10, and epsilon datasets, each with classes partitioned into P and U
- Compare PN, uPU, and nnPU with the following setups:
  - for PN: $n_p = 1000$ and $n_n = \left(\frac{\pi_n}{2\pi_p}\right)^2 n_p$
  - for uPU: $n_p = 1000$ and $n_u$ is total number of training data
  - for nnPU: same as above
- Also investigate cases when $\pi_p$ is misspecified
  - Test nnPU by replacing $\pi_p$ with $\pi_p' \in \{0.8\pi_p, 0.9\pi_p, \ldots 1.2\pi_p\}$
Construct P-N classes: \( P = \{0, 2, 4, 6, 8\} \) and \( N = \{1, 3, 5, 7, 9\} \)

\( \pi_p = 0.49 \) so \( n_n = 271 \)

Model \( g(x; \theta) \) as 6-Layer MLP with ReLU
Construct P-N classes: $P = \{ 'alt.', 'comp.', 'misc.', 'rec.' \}$ and $N = \{ 'sci.', 'soc.', 'talk' \}$

$\pi_p = 0.44$ so $n_n = 405$

Model $g(x; \theta)$ as 5-Layer MLP with Softsign activation. Fed avg-pooled word embeddings as inputs
CIFAR-10

- Construct P-N classes: \( P = \{ \text{'airplane'}, \text{'automobile'}, \text{'ship'}, \text{'truck'} \} \) and \( N = \{ \text{'bird'}, \text{'cat'}, \text{'deer'}, \text{'dog'}, \text{'frog'}, \text{'horse'} \} \)
- \( \pi_p = 0.40 \) so \( n_n = 563 \)
- Model \( g(x; \theta) \) as 13-Layer CNN with ReLU activation.
Epsilon

- Already 2 classes
- \( \pi_p = 0.50 \) so \( n_n = 250 \)
- Model \( g(x; \theta) \) as 6-Layer MLP with Softsign activation