PULNS: Positive-Unlabeled Learning with Effective Negative Sample Selector

Chuan Luo,[1,*] **Pu Zhao,**[1,*] **Chen Chen,**[1,2] **Bo Qiao,**[1] **Chao Du,**[1] **Hongyu Zhang,**[3] **Wei Wu,**[4] **Shaowei Cai,**[5,6] **Bing He,**[5,6] **Saravanakumar Rajmohan,**[2] **Qingwei Lin**[1,†]

[1]Microsoft Research, China [2]Microsoft 365, United States
[3]The University of Newcastle, Australia [4]L3S Research Center, Leibniz University Hannover, Germany
[5]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
[6]School of Computer Science and Technology, University of Chinese Academy of Sciences, China
{chuan.luo, puzhao, v-chch22, boqiao, chaodu, saravar, qlin}@microsoft.com,
hongyu.zhang@newcastle.edu.au, william.third.wu@gmail.com, {caisw, hebing}@ios.ac.cn

Abstract

Positive-unlabeled learning (PU learning) is an important case of binary classification where the training data only contains positive and unlabeled samples. The current state-of-the-art approach for PU learning is the *cost-sensitive* approach, which casts PU learning as a cost-sensitive classification problem and relies on unbiased risk estimator for correcting the bias introduced by the unlabeled samples. However, this approach requires the knowledge of class prior and is subject to the potential label noise. In this paper, we propose a novel PU learning approach dubbed *PULNS*, equipped with an effective negative sample selector, which is optimized by reinforcement learning. Our *PULNS* approach employs an effective negative sample selector as the agent responsible for selecting negative samples from the unlabeled data. While the selected, likely negative samples can be used to improve the classifier, the performance of classifier is also used as the reward to improve the selector through the REINFORCE algorithm. By alternating the updates of the selector and the classifier, the performance of both is improved. Extensive experimental studies on 7 real-world application benchmarks demonstrate that *PULNS* consistently outperforms the current state-of-the-art methods in PU learning, and our experimental results also confirm the effectiveness of the negative sample selector underlying *PULNS*.

Introduction

In a binary classification problem, the classifier is trained on a dataset with both positive and negative samples. In practice, the training data may only contain positive samples (P) and unlabeled samples (U). For instance, while we can label subscribers who had watched at least one Star Wars movie as being interested in this genre, we cannot be sure about the interests of a subscriber who never watched a Star Wars movie before. Similar examples can be found in many applications, including information retrieval, recommendation system, medical diagnosis, text classification and outlier detection (Blanchard, Lee, and Scott 2010; Nguyen, Li, and Ng 2012; Fei and Liu 2015; Schnabel et al. 2016). Classification using such dataset is called positive-unlabeled (PU) learning, which has attracted huge attention from both academia and industry in recent years.

Lacking of labeled negative samples (N), information essential for constructing the classifier must be learnt from the unlabeled samples. Depending on how the unlabeled samples are treated, most of the recent works on PU learning can be grouped into two categories, which we term as the *cost-sensitive* approach and the *sample-selection* approach.

Pioneered by (Elkan and Noto 2008), the *cost-sensitive* approach has emerged as the state-of-the-art method in PU learning during recent years (du Plessis, Niu, and Sugiyama 2014, 2015; Kiryo et al. 2017; Kato, Teshima, and Honda 2019; Hsieh, Niu, and Sugiyama 2019; Li et al. 2019). Following a cost-sensitive classification framework, this approach essentially treats each unlabeled sample as the weighted combination of positive and negative sample, and relies on unbiased risk estimator to correct the bias introduced by unlabeled data. Still, to apply *cost-sensitive* methods, one often needs to estimate the class prior first. This is not a simple task and is the focus of many studies (du Plessis and Sugiyama 2014; Jain, White, and Radivojac 2016; Ramaswamy, Scott, and Tewari 2016; Bekker and Davis 2018; Li et al. 2019; Perini, Vercruyssen, and Davis 2020; Zeiberg, Jain, and Radivojac 2020; Jain et al. 2020). Moreover, label noise caused by treating unlabeled positive samples as negative ones may still degrade the performance of *cost-sensitive* approach in practice.

In contrast, the *sample-selection* approach uses heuristic methods to select likely negative samples from the unlabeled data. The chosen negative samples are then merged with the positive data to train a supervised classifier. Methods for selecting negative samples include Naïve Bayes (Liu et al. 2002), Rocchio method (Li and Liu 2003), 1-DNF method (Yu, Han, and Chang 2004), k-NN (Zhang and Zuo 2009), and k-means (Chaudhari and Shevade 2012). While mitigating the issues faced by the *cost-sensitive* approach, the *sample-selection* approach lacks the mechanisms for opti-

[*]Chuan Luo and Pu Zhao contributed equally to this work and share the first authorship of this work.

[†]Qingwei Lin is the corresponding author of this work.

mizing the negative samples selector which is crucial for its performance.

In this work, we propose a new *sample-selection* approach named Positive-Unlabeled Learning with Effective Negative Sample Selector (*PULNS*). Our *PULNS* approach is composed of a *negative sample selector* and a *binary classifier* which are optimized in a reinforcement learning based framework. Specifically, we treat the selector as the agent, the selection of negative samples as the action, and the performance of the classifier as the reward. This setup allows us to update the selector via reinforcement learning, which can then be used to train a better classifier. By iterating between the optimization of the classifier and the selector, this method improves both the selector and the classifier together to achieve better performance than the state-of-the-art *cost-sensitive* approach.

We summarize the main contributions of this work below:

- We propose an effective strategy to train and optimize the negative sample selector based on the performance of classifier. We also conduct experiments and show that our selector is more effective in selecting negative samples than the selectors used in other *sample-selection* methods.
- We construct an optimization framework based on reinforcement learning to enable end-to-end updates on both selector and classifier.
- We conduct extensive experiments on seven benchmarks to compare *PULNS* against other methods and show that *PULNS* can consistently outperform the state-of-the-art *cost-sensitive* approach.

Related Work

As the current state of the art in PU learning, the *cost-sensitive* approach adopts the framework of cost-sensitive classifier where unbiased risk estimator is constructed to correct the bias introduced by training a classifier over the positive and unlabeled samples. Since the first unbiased risk estimator was introduced for solving PU learning problem (du Plessis, Niu, and Sugiyama 2014), many works were done to enhance this technique. Notably, convex unbiased risk estimator can be used to reduce computational cost (du Plessis, Niu, and Sugiyama 2015). Non-negative risk estimator can be used to overcome the over-fitting issue when the model is too flexible (Kiryo et al. 2017). Similar unbiased risk estimator can also be constructed if a small subset of negative samples (which can be a biased representation of the negative population) is available for training the classifier (Hsieh, Niu, and Sugiyama 2019). While most PU learning methods assume that the labeled positive data follows the same distribution as the unlabeled positive data (no selection bias), a modified *cost-sensitive* approach can be applied to the scenario with selection bias (Kato, Teshima, and Honda 2019). Still, in order to make the proposed risk estimator be truly unbiased, the sampling procedure needs to satisfy certain assumptions, and the class prior should be known or at least can be accurately estimated. When these assumptions do not hold, the performance of the *cost-sensitive* approach might suffer from the label noise issue introduced by the unlabeled samples. In this work, we show that, by selecting a reliable set of negative samples, our method can outperform state-of-the-art approaches discussed in this category.

Our method is not the only approach that utilizes iterative procedure for improving the selection of negative samples. In fact, most *sample-selection* methods discussed in the introduction section employ iterative procedures to enlarge the set of the selected negative samples based on the prediction of trained classifier. Still, such procedures do not choose nor improve the initial selector, while *PULNS* is capable of optimizing the selector directly. These approaches cannot explore different class assignments for unlabeled data either, a feature that is naturally supported by *PULNS*. Moreover, as pointed out by (Gong et al. 2018), iterative procedure that gradually enlarges the set of negative sample cannot guarantee to improve the classifier unless the positive samples are located far away from decision boundary.

Our Proposed Approach

We present a new *sample-selection* approach for PU learning, dubbed Positive-Unlabeled Learning with Effective Negative Sample Selector (*PULNS*). First, we introduce the problem setting of PU learning. Then we provide an overview of *PULNS* and describe the major algorithmic components underlying *PULNS*. Finally, we discuss the training process of *PULNS*.

Problem Setting

We represent the training dataset in PU learning as a sequence of n samples: $T = \{t_1, \ldots, t_u, \ldots, t_n\}$, where $t_i = (x_i, y_i)$ is a feature-label pair ($i \in \{1, \ldots, n\}$). For each sample $t_i = (x_i, y_i)$, $x_i \in \mathbb{R}^d$ is the raw feature vector; $y_i \in \{0, 1\}$ is the class indicator: $y_i = 0$ means that sample t_i belongs to the negative class, and $y_i = 1$ indicates that sample t_i belongs to the positive class. The first u samples are assumed to be unlabeled with unknown y_i ($1 \leq i \leq u$) whose ground truth can be either positive or negative. The rest $p = n - u$ samples are labeled. In the context of PU learning, all labeled samples are positive. Thus we have $y_i = 1$ for $u + 1 \leq i \leq n$. For simplicity, we use $U = \{t_1, \cdots, t_u\}$ to represent the set of all unlabeled samples, and $P = \{t_{u+1}, \cdots, t_n\}$ to represent the set of all labeled, positive samples. The task of PU learning is to train a classifier based on the positive sample set P and the unlabeled sample set U.

Approach Overview

In this subsection, we provide an approach overview of *PULNS*, which is illustrated in Figure 1. Our *PULNS* approach is based on a reinforcement learning framework with four important components: 1) *state* (*i.e.,* features extracted from unlabeled samples); 2) *agent* (*i.e.,* negative sample selector); 3) *environment* (*i.e.,* classifier); 4) *reward*.

PULNS consists of two alternating phases: the update of selector and the update of classifier. In the first phase, the agent (*i.e.,* the selector underlying *PULNS*) traverses the unlabeled samples in U sequentially. For each sample $t_i \in U$, the selector takes its state s_i as input and generates an action a_i (whether t_i is chosen as a negative sample or not). The

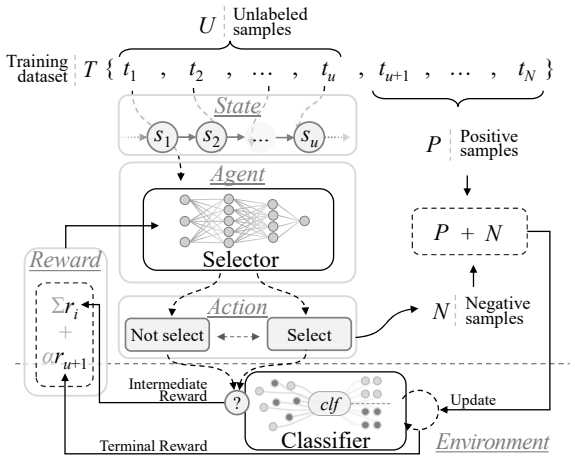

Figure 1: Approach Overview of *PULNS*.

environment (*i.e.,* classifier) generates rewards based on the actions. Also, the selector is updated using the REINFORCE algorithm (Sutton et al. 1999). In the second phase, the updated selector generates a negative sample set N from U. Both P and N are then used to update the classifier clf.

In this way, *PULNS* combines the selector and the classifier in an integral manner. Also, the interaction between the selector and the classifier forms an effective feedback loop, so the performance of the selector and the classifier can be improved together.

Components Underlying *PULNS*

In this subsection, we describe the major components underlying *PULNS* in detail.

Classifier For each sample t_i, the classifier takes the feature vector x_i as input and calculates the probability of x_i belonging to the positive class, denoted by $p(x_i)$. The classifier also needs to generate a vector representation x'_i for each x_i, which can be the output of the last hidden layer in a neural network classifier. It is recommended to choose a classifier that fits the data best. In our experiments, we use different classifiers for different benchmarks and achieve state-of-the-art performance on all benchmarks. Detailed descriptions on the classifiers used in our experiments can be found in the Experiments section.

State For each sample $t_i \in U$, to effectively represent t_i's status, its state s_i needs to encode the following information.

- Representation of current sample x'_i: The classifier adopted by *PULNS* is responsible for extracting the vector representation of the given sample t_i, denoted by x'_i. Incorporating x'_i into state s_i can make full use of the better feature extracting ability of the classifier and transfer learned knowledge from classifier to reduce the difficulty in training our selector.
- Representation of selected, negative samples c_i: It is the average of the vector representations of current negative samples in N, *i.e.,* $c_i = \frac{1}{|N|}\sum_{t_j \in N} x'_j$.[1]
- Representation of positive samples p: It is the average of the vector representations of all positive samples in P, *i.e.,* $p = \frac{1}{|P|}\sum_{t_j \in P} x'_j$. We note that the representation p is a fixed vector in each episode.

For each sample $t_i \in U$, its state s_i is designed as $s_i = (x'_i, c_i, p)$, in order to incorporate the above representations.

Negative Sample Selector In principle, for each sample t_i, the negative sample selector underlying our *PULNS* approach is required to take its state s_i and to output a real number ranging from 0 to 1 as t_i's predicted selection probability. In this work, our selector f_θ is specified as a multi-layer perceptron (MLP)[2]. For our selector f_θ, the activation function of each hidden layer is *ReLu*, while we use *sigmoid* as the output layer's activation function. In a word, for each sample t_i, based on its state s_i our selector computes t_i's predicted selection probability $f_\theta(s_i)$ $(0 \leq f_\theta(s_i) \leq 1)$.

The main focus of our selector is to provide an action $a_i \in \{0, 1\}$ to decide whether the corresponding sample t_i would be chosen as negative one. In particular, $a_i = 1$ indicates that t_i is selected as negative one; otherwise $(a_i = 0)$, t_i would not be selected. The policy function is as follows:

$$\pi_\theta(s_i, a_i) = P_\theta(a_i|s_i) = a_i f_\theta(s_i) + (1 - a_i)(1 - f_\theta(s_i))$$

Reward Since our *PULNS* approach is based on reinforcement learning, a core problem is how to design effective reward function to better train our negative sample selector.

In the l-th episode of *PULNS*'s training process, after selecting the negative sample set N' by our selector, *PULNS* trains a classifier based on P and N', and evaluates the accuracy of the trained classifier on a validation dataset, denoted by $z^{(l)}$. Since our ultimate goal is to improve the accuracy of the classifier, thus it is advisable to design a terminal reward by incorporating the accuracy $z^{(l)}$. A natural way is to directly treat $z^{(l)}$ as the reward; however, since the accuracy of the trained classifier is always non-negative, directly using z_l as the terminal reward would hinder the convergence of our negative sample selector. Hence, our *terminal reward* is designed as $r^{(l)}_{u+1} = z^{(l)} - b^{(l)}$, where $b^{(l)}$ is a baseline reward, *i.e.,* $b^{(l)} = \max\{z^{(0)}, \ldots, z^{(l-1)}\}$; here $z^{(0)}$ denotes the accuracy of the initial classifier trained based on P and U by treating all samples in U as negative ones.

Moreover, as we can only get the terminal reward at the end of each episode after taking u actions, it is recognized that the sparse reward issue would make reinforcement learning based approaches less effective (Riedmiller et al. 2018). To address this challenge, inspired by the success of reward shaping mechanism in other research fields (Ng, Harada, and Russell 1999; Pocius et al. 2018), we propose a new reward shaping mechanism to provide the selector with more guidance. In particular, in each episode, besides the terminal reward, *PULNS* calculates an *intermediate reward*

[1] If $N = \varnothing$, we use the average of the vector representations of all unlabeled samples in U, *i.e.,* $c_i = \frac{1}{|U|}\sum_{t_j \in U} x'_j$.

[2] More specially, the architecture of our adopted MLP is d-64-32-1, where d denotes the dimension of the input vector.

for each action. That is to say, in the l-th episode, for sample t_i, once our selector takes an action a_i, then the intermediate reward $r_i^{(l)}$ is computed as follows:

$$r_i^{(l)} = \begin{cases} \mathbb{C}(-1, \log \frac{1-p(x_i)}{p(x_i)}, 1) & \text{if } a_i = 1, \\ \\ \mathbb{C}(-1, \log \frac{p(x_i)}{1-p(x_i)}, 1) & \text{if } a_i = 0. \end{cases}$$

where $p(x_i)$ is given by the classifier based on x_i, and $\mathbb{C}(-1, \cdot, 1)$ is a clamping function: a) $\mathbb{C}(-1, x, 1) = -1$ if $x < -1$; b) $\mathbb{C}(-1, x, 1) = 1$ if $x > 1$; c) otherwise, $\mathbb{C}(-1, x, 1) = x$.

Since our selector aims to choose negative samples, the intuition behind our intermediate reward design is: for each unlabeled sample t_i, if the classifier tends to identify t_i as a negative sample, then our selector should receive positive rewards when $a_i = 1$, and receive negative rewards when $a_i = 0$; if the classifier tends to identify t_i as a positive sample, then our selector should receive negative rewards when $a_i = 1$, and receive positive rewards when $a_i = 0$.

It is worth noting that a small validation dataset containing both labeled positive and negative samples is needed for evaluating the performance of classifier. In real applications, while collecting large amount of negative samples can be impractical, labelling a small portion of negative data for validation is generally quite feasible and is commonly used. Moreover, as suggested by (Hsieh, Niu, and Sugiyama 2019), a small set of negative samples can bring considerable improvement in PU learning even if this set is a biased representation of the negative population.

Model Training

In this subsection, we first introduce how to optimize the selector, and then describe how to conduct the end-to-end training process on the whole *PULNS* framework.

Selector Optimization As discussed before, in each episode, our selector traverses the unlabeled samples in U sequentially: for each sample $t_i \in U$, our selector takes action a_i based on state s_i, and collects intermediate reward r_i. Then trajectory $\tau = \{(s_1, a_1, r_1), ..., (s_u, a_u, r_u)\}$ and terminal reward r_{u+1} can be obtained. Our selector aims to maximize the expected total reward including all intermediate rewards and the terminal reward. We define the objective function of our selector as follows:

$$J(\theta) = \mathbb{E}[\sum_{i=1}^{u} r_i + \alpha r_{u+1} | \pi_\theta] = \sum_{\tau} P_\theta(\tau)(\sum_{i=1}^{u} r_i + \alpha r_{u+1})$$

where α is a weight to emphasize terminal reward r_{u+1}, and $P_\theta(\tau)$ is the probability of the trajectory under current policy. We use the REINFORCE algorithm (Sutton et al. 1999) to update the policy parameter θ of our selector in each episode. The objective function with respect to θ is:

$$\begin{aligned} \nabla_\theta J(\theta) &= \sum_{\tau} \nabla_\theta P_\theta(\tau)(\sum_{i=1}^{u} r_i + \alpha r_{u+1}) \\ &= \sum_{\tau} P_\theta(\tau) \nabla_\theta \log P_\theta(\tau)(\sum_{i=1}^{u} r_i + \alpha r_{u+1}) \\ &= \mathbb{E}[\sum_{i=1}^{u} \nabla_\theta \log \pi_\theta(s_i, a_i)(\sum_{i'=i}^{u} r_{i'} + \alpha r_{u+1}) | \pi_\theta] \end{aligned}$$

where $(\sum_{i'=i}^{u} r_{i'} + \alpha r_{u+1})$ is the total future reward at step i after action a_i is taken.

In *PULNS*, we employ factor $\beta \in [0, 1]$ to discount the intermediate rewards, and emphasize the terminal reward r_{u+1} with weight α. We estimate the value-action function at step i after taking action a_i as $v_i = \sum_{t=i}^{u} \beta^{t-i} r_t + \alpha r_{u+1}$, and approximate the gradient as follows:

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{u} v_i \nabla_\theta \log \pi_\theta(s_i, a_i)$$

Then, our *PULNS* approach updates the current policy parameters as follows:

$$\theta \leftarrow \theta + \sum_{i=1}^{u} v_i \nabla_\theta \log \pi_\theta(s_i, a_i)$$

End-to-End Training As discussed before, the interaction between the selector and the classifier forms an effective feedback loop, and they are trained jointly and enhanced together through reinforcement learning. This subsection introduces the end-to-end training process. To update the se-

Algorithm 1: End-to-End Training Process of *PULNS*

Input: Positive sample set, P; Unlabeled sample set, U; Validation dataset, V; Episode number, L;
Output: Optimized selector, f_θ^*; Optimized classifier, clf^*;

Initialize a classifier clf which is trained based on P and U (treating all samples in U as negative ones);
Initialize a selector f_θ randomly;
$f_\theta^* \leftarrow f_\theta$, $clf^* \leftarrow clf$;
for $l \leftarrow 1$ *to* L **do**
  Shuffle the unlabeled sample set U;
  for *step* $i \leftarrow 1$ *to* $|U|$ **do**
    Get state s_i;
    Sample action $a_i \sim \pi_\theta(s_i, a_i)$;
    Obtain intermediate reward r_i;
  Compute terminal reward r_{u+1};
  Update selector f_θ based on all rewards;
  $f_\theta^* \leftarrow f_\theta$;
  Get calibrated negative sample set N from U by the updated selector f_θ^*;
  Update classifier clf based on N and P;
  if clf performs better than clf^* **then** $clf^* \leftarrow clf$;
return f_θ^*, clf^*;

lector underlying *PULNS*, we employ the REINFORCE algorithm (Sutton et al. 1999), while to update the classifier, we adopt gradient descent to minimize its own loss function.

The end-to-end training process of our *PULNS* approach is outlined in Algorithm 1, and is described as follows. First of all, our classifier is pre-trained based on P and U by treating all samples in U as negative ones, and our selector is initialized randomly. Then *PULNS* conducts an iterative process to jointly train our selector and classifier. In each episode, *PULNS* first shuffles the unlabeled sample set U. Then, for each sample $t_i \in U$, *PULNS* gets state s_i, takes action a_i, and obtains intermediate reward r_i. After all samples in U are processed, the terminal reward for this episode can be computed. Subsequently, the selector underlying *PULNS* can be updated based on all rewards (including all intermediate ones and the terminal one). After that, *PULNS* employs the updated selector to obtain the calibrated negative sample set N from U. Finally, the classifier underlying *PULNS* can be updated based on P and N, and the next episode starts. The end-to-end training process is stopped once the maximum episode number L is reached.

Experiments

In this section, we conduct extensive experiments on 7 public, application benchmarks to evaluate the performance of our *PULNS* approach along with 4 state-of-the-art competitors. We first describe the benchmarks, the competitors, and our experimental setups. Then we report and analyze the experimental results to present the effectiveness of *PULNS*.

Benchmarks

In the context of PU learning, seven benchmarks are commonly used to evaluate PU learning approaches (Kato, Teshima, and Honda 2019): 1) `CIFAR-10`[3] and 2) six benchmarks collected from UCI[4,5], including `MNIST`, `mushrooms`, `shuttle`, `spambase`, `usps` and `landsat`. Following the common practice, we adopt those seven benchmarks to evaluate the performance of *PULNS* and its competitors. As the `CIFAR-10`, `MNIST`, `shuttle`, `usps` and `landsat` benchmarks all contain multiple classes (10, 10, 7, 10 and 6 classes, respectively), we preprocess these benchmarks following the existing conventions (Kato, Teshima, and Honda 2019), as shown in Table 1.

A brief summary of the benchmarks used in our experiments, including the numbers of positive and negative samples and dimension of features, are listed in Table 2.

Benchmark	Positive class	Negative class
`CIFAR-10`	airplane, truck, automobile, ship	bird, cat, deer, dog, frog, horse
`MNIST`	0, 2, 4, 6, 8	1, 3, 5, 7, 9
`shuttle`	5, 6, 7	1, 2, 3, 4
`usps`	6, 7, 8, 9, 10	1, 2, 3, 4, 5
`landsat`	4, 5, 7	1, 2, 3

Table 1: Preprocessing of the benchmarks.

Benchmark	#Samples	Dim.	#P / #N
`CIFAR-10`	60,000	3072	24,000 / 36,000
`MNIST`	70,000	784	34,418 / 35,582
`mushrooms`	8,124	22	3,916 / 4,208
`shuttle`	58,000	9	12,414 / 45,586
`spambase`	4,601	57	1,813 / 2,788
`usps`	9,298	256	4,876 / 4,422
`landsat`	67,557	36	44,473 / 23,084

Table 2: Details of the benchmarks in our experiments.

Competitors

In our experiments, we compare our proposed *PULNS* approach against 4 recent, state-of-the-art and open-source competitors, including ***uPU*** (du Plessis, Niu, and Sugiyama 2015), ***nnPU*** (Kiryo et al. 2017), ***PUSB*** (Kato, Teshima, and Honda 2019) and ***PUbN*** (Hsieh, Niu, and Sugiyama 2019). We briefly describe those competitors as follows. ***uPU*** is an effective approach for PU learning based on a general unbiased risk generator. ***nnPU*** adopts a non-negative risk estimator and exhibits good performance on the `CIFAR-10` and `MNIST` benchmarks. ***PUSB*** is a recent, high-performance PU learning approach capable of alleviating the impact of selection bias and shows competitive performance on a number of benchmarks. ***PUbN*** is a recent, powerful PU learning approach based on empirical risk minimization, which achieves state-of-the-art performance on the `CIFAR-10` and `MNIST` benchmarks. The source codes for these four state-of-the-art competitors (*i.e., uPU*[6], *nnPU*[6], *PUSB*[7], *PUbN*[8]) are all available online.

Experimental Setup

Here we introduce: 1) how we construct the positive sample set P and the unlabeled sample set U for each benchmark; 2) the evaluation metric we adopt in our experiments; 3) the classifier we specify for each benchmark.

Construction of P and U For each benchmark, the training dataset is divided into P consisting of $|P|$ positive samples, and U that contains $|U|$ samples with both positive and negative ones. All class labels in U are removed in the follow-up analysis. Samples in P are randomly selected from the population of positive samples in the benchmark. As for U, we use γ to denote the proportion of positive samples, and randomly select $|U| \cdot \gamma$ positive samples and $|U| \cdot (1-\gamma)$ negative samples from the corresponding populations respectively.

The values of $|P|$ and $|U|$ are adopted from the setup used by (Kato, Teshima, and Honda 2019): for benchmarks `mushrooms`, `shuttle`, `spambase`, `usps` and

[3]https://www.cs.toronto.edu/~kriz/cifar.html

[4]https://archive.ics.uci.edu/ml/index.php

[5]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

[6]https://github.com/kiryor/nnPUlearning

[7]https://github.com/MasaKat0/PUlearning

[8]https://github.com/cyber-meow/PUbiasedN

γ	Benchmark	*uPU*	*nnPU*	*PUSB*	*PUbN*	*PULNS*
0.2	CIFAR-10	12.03 (0.013)	11.04 (0.013)	10.01 (0.013)	9.76 (0.009)	**7.98** (0.007)
	MNIST	8.48 (0.010)	7.01 (0.016)	6.24 (0.015)	6.17 (0.011)	**3.63** (0.006)
	mushrooms	5.23 (0.019)	5.21 (0.019)	4.97 (0.020)	4.78 (0.019)	**3.13** (0.011)
	shuttle	7.00 (0.019)	6.79 (0.019)	4.91 (0.007)	4.11 (0.020)	**2.74** (0.017)
	spambase	12.38 (0.021)	12.22 (0.021)	12.20 (0.020)	14.78 (0.013)	**9.59** (0.076)
	usps	9.44 (0.003)	8.63 (0.016)	7.77 (0.005)	7.51 (0.007)	**5.43** (0.008)
	landsat	8.19 (0.017)	7.96 (0.015)	6.48 (0.024)	6.35 (0.012)	**4.22** (0.026)
0.4	CIFAR-10	16.01 (0.015)	15.37 (0.017)	14.59 (0.013)	13.93 (0.017)	**11.44** (0.011)
	MNIST	15.45 (0.018)	8.62 (0.011)	11.54 (0.011)	8.10 (0.013)	**5.27** (0.009)
	mushrooms	8.64 (0.018)	8.41 (0.018)	5.68 (0.020)	6.45 (0.019)	**4.30** (0.024)
	shuttle	10.01 (0.022)	8.97 (0.021)	6.31 (0.012)	5.78 (0.018)	**4.43** (0.016)
	spambase	20.94 (0.021)	20.78 (0.022)	19.40 (0.027)	17.31 (0.011)	**15.70** (0.014)
	usps	12.77 (0.009)	10.26 (0.004)	9.51 (0.005)	10.13 (0.009)	**8.53** (0.010)
	landsat	9.30 (0.010)	8.77 (0.014)	8.10 (0.014)	7.60 (0.013)	**7.03** (0.009)
0.6	CIFAR-10	17.39 (0.016)	16.56 (0.015)	15.82 (0.014)	14.96 (0.013)	**11.73** (0.014)
	MNIST	27.30 (0.019)	10.37 (0.010)	13.80 (0.017)	10.90 (0.014)	**6.54** (0.016)
	mushrooms	9.68 (0.021)	9.16 (0.020)	7.69 (0.021)	8.32 (0.013)	**5.72** (0.026)
	shuttle	10.88 (0.020)	9.58 (0.019)	8.38 (0.011)	8.25 (0.006)	**5.39** (0.039)
	spambase	29.45 (0.018)	29.87 (0.020)	20.84 (0.031)	25.69 (0.017)	**22.39** (0.013)
	usps	13.08 (0.012)	11.11 (0.012)	11.10 (0.003)	10.93 (0.013)	**7.71** (0.012)
	landsat	10.28 (0.012)	10.08 (0.014)	9.38 (0.013)	9.24 (0.022)	**8.03** (0.015)

Table 3: Comparative results of *PULNS*, *uPU*, *nnPU*, *PUSB*, and *PUbN* on all benchmarks with γ =0.2, 0.4 and 0.6. For each cell 'x (y)', x, y denote the average error rate of binary classification in test data (%), and the standard deviation, respectively.

landsat, $|P|$ and $|U|$ are fixed at 400 and 800, respectively. For benchmarks CIFAR-10 and MNIST, $|P|$ and $|U|$ are fixed at 2000 and 4000, respectively. To better simulate the practical scenarios and to comprehensively evaluate all competing approaches, we use 3 different settings for $\gamma = \{0.2, 0.4, 0.6\}$ to resemble the different proportions of positive samples within the unlabeled samples.

For each benchmark, the validation dataset and the testing dataset are generated through the same sampling method for U in the training dataset with the same values for γ. For benchmarks mushrooms, shuttle, spambase, usps and landsat, the total numbers of samples in the validation set and the testing set are set to 100 and 1000, respectively. For benchmarks CIFAR-10 and MNIST, the total numbers of samples in the validation set and the testing set are set to 500 and 5000, respectively.

Evaluation Metric To assess the performance of *PULNS* and its competitors, we adopt the error rate of classification (*i.e.,* $1 - accuracy$) on testing dataset as our evaluation metric following the common practice in the literature (Kato, Teshima, and Honda 2019; Zhang, Hou, and Zhang 2020). Hence, in our experiments, the smaller the error rate, the better the performance. For each benchmark with a particular value of γ, 20 independent runs are performed for *PULNS* and its 4 competitors. The average error rate and the standard deviation across the 20 runs are reported.

Classifier Specification In PU learning, different classifiers are often adopted for dealing with different benchmarks. Therefore, following the work (Kato, Teshima, and Honda 2019), the following two classifiers are adopted: 1) for the CIFAR-10 benchmark, we adopt a convolutional neural network as the classifier; 2) for the remaining benchmarks, we adopt a multilayer perceptron (MLP) with a single-hidden-layer of 100 neurons as the classifier. Moreover, to ensure a fair comparison, for each benchmark, the same classifier is used by all the competing methods.

Experimental Results

We conduct extensive experiments on 7 application benchmarks to study the performance of *PULNS* and the effectiveness of its selector. Note that in all the experiments, the true value of class prior γ is assumed to be known. This value is needed for all the 4 state-of-the-art competitors but not our *PULNS* approach and has to be estimated in practice.

Comparing *PULNS* Against Competitors Table 3 reports the comparative results of *PULNS* and its 4 state-of-the-art competitors (*i.e., uPU*, *nnPU*, *PUSB* and *PUbN*) on 7 application benchmarks with 3 different values of γ (0.2, 0.4, and 0.6). In Table 3, for each benchmark with a particular γ, the best result across all five competing methods is indicated in **boldface**. From Table 3, *PULNS* stands out as the best method on all benchmarks with all γ values. In particular, on the MNIST benchmark with $\gamma = 0.2$, 0.4 and 0.6, *PULNS* achieves the average error rates of 3.63%, 5.27% and 6.54%, respectively, while the second best average error rates are 6.17% (*PUbN*), 8.10% (*PUbN*) and 10.37% (*nnPU*), respectively. Similarly, on the shuttle benchmark with $\gamma = 0.2$, 0.4 and 0.6, *PULNS* yields the average error rates of 2.74%, 4.43% and 5.39%, respectively, while the average error rates for the second best approach

γ	Method	CIFAR-10	MNIST	mushrooms	shuttle	spambase	usps	landsat
0.2	*PULNS_alt*	12.92	7.48	7.42	9.06	17.78	17.44	6.63
	PULNS	7.98	3.63	3.13	2.74	9.59	5.43	4.22
	Oracle	7.20	3.00	1.61	0.52	8.87	4.74	4.15
0.4	*PULNS_alt*	18.84	12.28	20.75	16.41	27.51	24.20	18.24
	PULNS	11.44	5.27	4.30	4.43	15.70	8.53	7.03
	Oracle	10.72	4.32	2.10	0.75	14.52	5.80	5.72
0.6	*PULNS_alt*	48.26	38.58	41.20	44.35	43.43	37.65	44.30
	PULNS	11.73	6.54	5.72	1.90	22.39	7.71	8.03
	Oracle	11.10	4.78	3.72	0.91	19.87	5.36	6.08

Table 4: Comparative results of *PULNS*, *PULNS_alt* and *Oracle* on all benchmarks with $\gamma = 0.2$, 0.4 and 0.6. In this table, the metric is the average error rate of binary classification on testing set (%).

γ	Selector	mushrooms		shuttle		spambase		landsat		usps	
		NPV	TNR	NPV	TNR	NPV	TNR	NPV	TNR	NPV	TNR
0.2	*Selector_1*	96.71	73.44	81.71	89.38	91.74	62.50	95.31	92.03	95.50	89.53
	Selector_2	95.88	90.94	91.83	93.12	83.00	94.58	95.73	94.69	95.39	90.62
	Selector_3	91.50	84.06	87.12	88.75	92.09	89.79	95.80	92.66	96.30	85.47
	Selector_4	91.52	64.06	87.80	90.00	77.99	88.59	91.73	39.84	91.98	64.53
	Selector_5	97.34	85.62	84.79	94.06	89.96	93.33	97.24	93.44	95.87	91.88
	Selector_PULNS	**98.26**	**97.03**	**97.03**	**96.88**	**92.17**	**95.63**	**98.23**	**95.47**	**96.41**	**96.56**
0.4	*Selector_1*	89.80	73.33	80.04	95.21	92.42	76.25	86.41	88.75	89.42	89.79
	Selector_2	90.31	85.42	82.01	96.88	88.76	82.29	90.58	87.19	91.98	90.83
	Selector_3	80.51	86.04	74.17	92.71	82.58	85.94	82.42	85.00	90.17	86.04
	Selector_4	77.60	90.94	71.63	96.25	76.32	81.56	83.28	79.38	84.47	64.58
	Selector_5	83.13	86.25	83.67	91.25	87.23	91.04	88.60	92.29	88.62	92.50
	Selector_PULNS	**90.32**	**93.33**	**91.39**	**97.29**	**92.48**	**92.29**	**95.54**	**93.75**	**92.07**	**98.54**
0.6	*Selector_1*	77.16	55.94	63.45	94.38	67.81	80.31	79.33	88.75	75.46	89.38
	Selector_2	78.01	83.12	79.33	88.75	63.36	83.75	75.46	89.38	79.33	88.75
	Selector_3	66.28	88.44	75.46	89.38	55.89	87.50	78.14	89.38	63.36	83.75
	Selector_4	53.96	96.50	67.81	80.31	64.65	86.88	75.14	81.25	70.43	81.88
	Selector_5	78.90	85.31	80.71	85.42	67.55	87.19	83.28	88.75	81.82	84.38
	Selector_PULNS	**80.56**	**97.03**	**85.08**	**96.25**	**73.20**	**92.19**	**90.68**	**91.25**	**92.24**	**92.81**

Table 5: Comparative results of our selector underlying *PULNS* and 5 representative, effective selectors on all benchmarks with $\gamma = 0.2$, 0.4 and 0.6. In this table, the metrics are NPV and TNR of binary classification on U data (%).

(*i.e., PUbN*) are 4.11%, 5.78% and 8.25%, respectively. The experimental results in Table 3 clearly present the superiority of *PULNS* over state-of-the-art competitors on those 7 application benchmarks, which indicates that *PULNS* might bring benefits in practice.

Effectiveness of the Selector Underlying *PULNS* To demonstrate the effectiveness of the selector underlying *PULNS*, we modify *PULNS* by removing our proposed selector, resulting in an alternative version called *PULNS_alt*. We conduct experiments to directly compare *PULNS* against *PULNS_alt* on all benchmarks with $\gamma = 0.2$, 0.4 and 0.6, and summarize the results in Table 4. According to Table 4, it is apparent that *PULNS* can achieve significantly lower error rate than *PULNS_alt*. Moreover, as γ becomes larger, the performance of *PULNS_alt* decreases significantly. In contrast, *PULNS* can still maintain reasonably good performance for large γ. All these evidences clearly demonstrate the effectiveness and the crucial role of our selector for handling the label noise introduced by unlabeled data.

Also, based on *PULNS*, we replace our selector with a (virtual) ideal selector, which can perfectly select negative samples from U data, to form an ideal approach named *Oracle*. The results achieved by *Oracle* can then be treated as the empirical performance upper bound of *PULNS*. We demonstrate the experimental results of *PULNS* and *Oracle* in Table 4. From Table 4, it is clear that, on all benchmarks with all γ settings, the average error rates achieved by *PULNS* are quite close to those achieved by *Oracle*, confirming the effectiveness of the selector underlying *PULNS*.

Comparing Selector Underlying *PULNS* Against Existing Selectors Here we would like to verify whether the trained selector underlying *PULNS* can accurately select negative samples from unlabeled samples. We conduct extensive experiments to directly compare the selec-

tor underlying *PULNS*, which we name *Selector_PULNS*, against five effective selectors chosen from literature for classifying unlabeled samples in U on 5 application benchmarks (*i.e.,* `mushrooms`, `shuttle`, `spambase`, `usps` and `landsat`). These 5 selectors are named *Selector_1* (Liu et al. 2002), *Selector_2* (Li and Liu 2003), *Selector_3* (Li, Liu, and Ng 2010), *Selector_4* (Chaudhari and Shevade 2012) and *Selector_5* (Zhang and Zuo 2009). Both `CIFAR-10` and `MNIST` are excluded from experiments because all competing selectors do not support images as their inputs. In this experiment, we adopt the metrics of Negative Predictive Value (denoted by 'NPV') and True Negative Rate (denoted by 'TNR'), which are calculated as follows:

$$\text{NPV} = \frac{\text{\#Selected\_N}}{\text{\#Selected}}, \quad \text{TNR} = \frac{\text{\#Selected\_N}}{\text{\#N}}$$

where '#Selected_N' denotes the number of negative samples among all selected samples, '#Selected' denotes the total number of selected samples, and '#N' denotes the total number of negative samples in U. Since the role of the selector in PU learning is to select negative samples from unlabeled samples, NPV provides a good measure on the accuracy of the selector, and TNR allows us to check how complete the selector can select negative samples. For both metrics, the larger the values, the better the performance.

The related experimental results are shown in Table 5, with NPV and TNR as the evaluation criteria. As can be seen from Table 5, *Selector_PULNS* can achieve much higher precision and recall than all the 5 competing selectors, which confirms that training process of *PULNS* is capable of optimizing the underlying selector to accurately select negative samples from the unlabeled data.

Conclusion

In this paper, we propose a novel PU learning approach dubbed *PULNS*, which is equipped with an effective negative sample selector. We utilize reinforcement learning, along with reward shaping strategy, to optimize the negative sample selector and train the whole framework through an end-to-end manner. We conduct extensive experiments on seven public application benchmarks, and our results confirm the effectiveness of *PULNS* over the current state-of-the-art approaches in PU learning. More encouragingly, we have applied our *PULNS* approach to the project of failure prediction in cloud systems (including Microsoft Azure and Microsoft 365's cloud system), in order to address the label noise issue. In practice, after the deployment of *PULNS*, the accuracy of failure prediction has been significantly improved. Besides, with the help of proactive mitigation actions, our *PULNS* approach can help considerably enhance the service reliability of Microsoft Azure and Microsoft 365's cloud system.

For future work, we plan to explore the extension of the proposed approach to semi-supervised learning.

Acknowledgments

We express our deep gratitude to Dongmei Zhang, Girish Bablani, Robert Gu, Jim Kleewein, Yingnong Dang, Murali Chintalapati, Thomas Moscibroda, Melur Raghuraman, Yogesh Bansal, Marcus Fontoura and Andrew Zhou for their great support and sponsorship. Furthermore, we would like to thank Susy Yi and Paul Wang for the collaboration on the project of failure prediction in Microsoft 365, and to thank Youjiang Wu, Xukun Li, and Sebastien Levy for the collaboration from Microsoft Azure.

References

Bekker, J.; and Davis, J. 2018. Estimating the Class Prior in Positive and Unlabeled Data Through Decision Tree Induction. In *Proceedings of AAAI 2018*, 2712–2719.

Blanchard, G.; Lee, G.; and Scott, C. 2010. Semi-Supervised Novelty Detection. *Journal of Machine Learning Research* 11: 2973–3009.

Chaudhari, S.; and Shevade, S. K. 2012. Learning from Positive and Unlabelled Examples Using Maximum Margin Clustering. In *Proceedings of ICONIP 2012*, 465–473.

du Plessis, M. C.; Niu, G.; and Sugiyama, M. 2014. Analysis of Learning from Positive and Unlabeled Data. In *Proceedings of NIPS 2014*, 703–711.

du Plessis, M. C.; Niu, G.; and Sugiyama, M. 2015. Convex Formulation for Learning from Positive and Unlabeled Data. In *Proceedings ICML 2015*, 1386–1394.

du Plessis, M. C.; and Sugiyama, M. 2014. Class Prior Estimation from Positive and Unlabeled Data. *IEICE Transactions on Information & Systems* 97-D(5): 1358–1362.

Elkan, C.; and Noto, K. 2008. Learning Classifiers from Only Positive and Unlabeled Data. In *Proceedings of KDD 2008*, 213–220.

Fei, G.; and Liu, B. 2015. Social Media Text Classification under Negative Covariate Shift. In *Proceedings of EMNLP 2015*, 2347–2356.

Gong, T.; Wang, G.; Ye, J.; Xu, Z.; and Lin, M. 2018. Margin Based PU Learning. In *Proceedings of AAAI 2018*, 3037–3044.

Hsieh, Y.; Niu, G.; and Sugiyama, M. 2019. Classification from Positive, Unlabeled and Biased Negative Data. In *Proceedings of ICML 2019*, 2820–2829.

Jain, S.; Delano, J.; Sharma, H.; and Radivojac, P. 2020. Class Prior Estimation with Biased Positives and Unlabeled Examples. In *Proceedings of AAAI 2020*, 4255–4263.

Jain, S.; White, M.; and Radivojac, P. 2016. Estimating the Class Prior and Posterior from Noisy Positives and Unlabeled Data. In *Proceedings of NIPS 2016*, 2685–2693.

Kato, M.; Teshima, T.; and Honda, J. 2019. Learning from Positive and Unlabeled Data with a Selection Bias. In *Proceedings of ICLR 2019*.

Kiryo, R.; Niu, G.; du Plessis, M. C.; and Sugiyama, M. 2017. Positive-Unlabeled Learning with Non-Negative Risk Estimator. In *Proceedings of NIPS 2017*, 1675–1685.

Li, T.; Wang, C.; Ma, Y.; Ortal, P.; Zhao, Q.; Stenger, B.; and Hirate, Y. 2019. Learning Classifiers on Positive and Unlabeled Data with Policy Gradient. In *Proceedings of ICDM 2019*, 399–408.

Li, X.; and Liu, B. 2003. Learning to Classify Texts Using Positive and Unlabeled Data. In *Proceedings of IJCAI 2003*, 587–594.

Li, X.; Liu, B.; and Ng, S. 2010. Negative Training Data Can be Harmful to Text Classification. In *Proceedings of EMNLP 2010*, 218–228.

Liu, B.; Lee, W. S.; Yu, P. S.; and Li, X. 2002. Partially Supervised Classification of Text Documents. In *Proceedings of ICML 2002*, 387–394.

Ng, A. Y.; Harada, D.; and Russell, S. J. 1999. Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping. In *Proceedings of ICML 1999*, 278–287.

Nguyen, M. N.; Li, X.; and Ng, S. 2012. Ensemble Based Positive Unlabeled Learning for Time Series Classification. In *Proceedings of DASFAA 2012*, 243–257.

Perini, L.; Vercruyssen, V.; and Davis, J. 2020. Class Prior Estimation in Active Positive and Unlabeled Learning. In *Proceedings of IJCAI 2020*, 2915–2921.

Pocius, R.; Isele, D.; Roberts, M.; and Aha, D. W. 2018. Comparing Reward Shaping, Visual Hints, and Curriculum Learning. In *Proceedings of AAAI 2018*, 8135–8136.

Ramaswamy, H. G.; Scott, C.; and Tewari, A. 2016. Mixture Proportion Estimation via Kernel Embeddings of Distributions. In *Proceedings of ICML 2016*, 2052–2060.

Riedmiller, M. A.; Hafner, R.; Lampe, T.; Neunert, M.; Degrave, J.; de Wiele, T. V.; Mnih, V.; Heess, N.; and Springenberg, J. T. 2018. Learning by Playing Solving Sparse Reward Tasks from Scratch. In *Proceedings of ICML 2018*, 4341–4350.

Schnabel, T.; Swaminathan, A.; Singh, A.; Chandak, N.; and Joachims, T. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *Proceedings of ICML 2016*, 1670–1679.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Proceedings of NIPS*, 1057–1063.

Yu, H.; Han, J.; and Chang, K. C. 2004. PEBL: Web Page Classification without Negative Examples. *IEEE Transactions on Knowledge and Data Engineering* 16(1): 70–81.

Zeiberg, D.; Jain, S.; and Radivojac, P. 2020. Fast Nonparametric Estimation of Class Proportions in the Positive-Unlabeled Classification Setting. In *Proceedings of AAAI 2020*, 6729–6736.

Zhang, B.; and Zuo, W. 2009. Reliable Negative Extracting Based on kNN for Learning from Positive and Unlabeled Examples. *Journal of Computers* 4(1): 94–101.

Zhang, C.; Hou, Y.; and Zhang, Y. 2020. Learning from Positive and Unlabeled Data without Explicit Estimation of Class Prior. In *Proceedings of AAAI 2020*, 6762–6769.