MIXPUL: CONSISTENCY-BASED AUGMENTATION FOR POSITIVE AND UNLABELED LEARNING

A PREPRINT

Tong Wei
Nanjing University, China
weit@lamda.nju.edu.cn

Feng Shi
Nanjing University, China
shif@lamda.nju.edu.cn

Hai Wang
4Paradigm Inc., China
wanghai.ha@gmail.com

Wei-Wei Tu
4Paradigm Inc., China
tuww.cn@gmail.com

Yu-Feng Li
Nanjing University, China
liyf@lamda.nju.edu.cn

April 21, 2020

ABSTRACT

Learning from positive and unlabeled data (PU learning) is prevalent in practical applications where only a couple of examples are positively labeled. Previous PU learning studies typically rely on existing samples such that the data distribution is not extensively explored. In this work, we propose a simple yet effective data augmentation method, coined MIXPUL, based on *consistency regularization* which provides a new perspective of using PU data. In particular, the proposed MIXPUL incorporates supervised and unsupervised consistency training to generate augmented data. To facilitate supervised consistency, reliable negative examples are mined from unlabeled data due to the absence of negative samples. Unsupervised consistency is further encouraged between unlabeled datapoints. In addition, MIXPUL reduces margin loss between positive and unlabeled pairs, which explicitly optimizes AUC and yields faster convergence. Finally, we conduct a series of studies to demonstrate the effectiveness of consistency regularization. We examined three kinds of reliable negative mining methods. We show that MIXPUL achieves an averaged improvement of classification error from 16.49 to 13.09 on the CIFAR-10 dataset across different positive data amount.

Keywords PU Learning · Consistency Regularization · Deep Neural Networks

1 Introduction

Positive and Unlabeled learning (PU learning) is emerging in real-world applications since labeling large amounts of data is often prohibitive due to time, financial, and expertise constraints. PU learning typically deals with binary classification and has been applied to novelty or outlier detection [1], software clone detection [2], and disease gene identification [3].

Given a large number of application scenarios, PU learning has been well studied in recent decades. Previous literature can be divided into two categories based on how unlabeled data is handled. The first line of research is called problem transformation. Through identifying reliable negative examples from unlabeled data, PU learning is transformed into supervised learning [4, 5]. Some other work regards unlabeled data directly as negative and considers hidden positive examples among unlabeled data as mislabeled examples. The PU learning problem is transformed into label noise learning [6, 7]. The second line of research is developing unbiased PU learning risk estimators. This type of research can be seen as cost-sensitive classification [1, 8, 9, 10]. These unbiased risk estimators typically rely on the knowledge of class-prior which is usually unavailable in real-world problems. Although several approaches have been proposed to estimate the class-prior from PU data [11, 12, 13, 14], inaccurate estimation usually results in severe performance degeneration as illustrated in Figure 1.

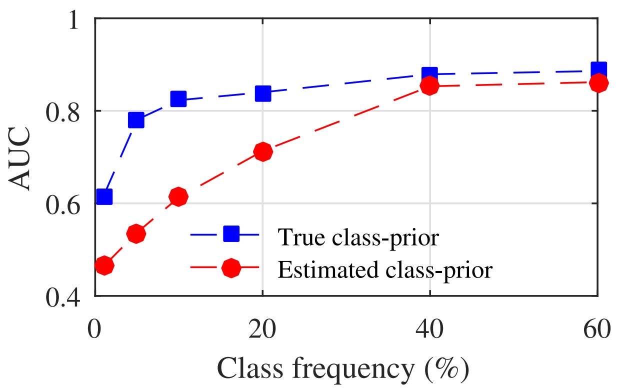


Figure 1: The performance comparison (AUC) using nnPU with true and estimated class-frequency is demonstrated. We vary the class-frequency $c = |\mathcal{P}|/(|\mathcal{P}| + \pi|\mathcal{U}|)$ to simulate PU problems in the wild where π is class-prior.

It is worth noting that although deep learning achieves excellent performance in semi-supervised learning tasks [15], it has not been fully applied to PU learning. Moreover, in much recent work, many effective strategies have been proposed for the training of deep neural networks leveraging unlabeled data, such as *consistency regularization* [16, 17] and *mixup* [18]. It is demonstrated that these approaches help enhance the performance for semi-supervised learning with a large margin on various problems.

In this paper, we introduce MIXPUL, a new consistency-based data augmentation algorithm. Unlike previous approaches, MIXPUL does not require the knowledge of class-prior. We introduce a unified loss term for PU data that seamlessly improves AUC while encouraging consistency between datapoints. By using mixup, it interpolates pairs of datapoints and their corresponding class labels. The network is then regularized to minimize the distance between its output and the interpolated class labels. It is observed that mixup can move the decision boundary to low-density regions of the data distribution [19] and encourage the model to generalize better to unseen data. Due to the absence of negative samples for training, we propose to mine reliable negative examples from unlabeled data to facilitate supervised consistency loss. In extensive ablation studies, we show the effectiveness of consistency regularization and negative example mining techniques.

In summary, our contributions are:

- We apply consistency regularization to PU learning to yield a simple yet effective approach which does not need the knowledge of class-prior compared with existing state of the art.
- We examine three reliable negative mining methods and show that the randomized technique works best.
- We conduct experiments showing that applying consistency regularization can yield substantial improvements over prior state of the art. For example, the proposed method improves classification error from 16.49 to 13.09 on average over the CIFAR-10 dataset.

The rest of this paper is arranged as follows. We start by a brief introduction to the problem setting and consistency regularization. Next, we present the proposed algorithm. After that, experimental results are reported followed by the conclusion of this work.

2 Preliminaries

2.1 Problem Setup

Given N samples $\mathcal{D} = \{\boldsymbol{x}_i, s_i\}_{i=1}^{N}$ where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $s_i \in \{0, 1\}$. $\boldsymbol{x}_i$ is regarded as a positive example if $s_i = 1$, otherwise an unlabeled example. We denote the set of positive examples as $\mathcal{P}$ and unlabeled set as $\mathcal{U} = \mathcal{D} \setminus \mathcal{P}$. The positive set $\mathcal{P}$ of data is sampled independently from the underlying joint density $p(\boldsymbol{x}|y = 1)$ and $\mathcal{U}$ is sampled from a mixture density $p(\boldsymbol{x}) = \pi p(\boldsymbol{x}|y = 1) + (1 - \pi)p(\boldsymbol{x}|y = 0)$ where π indicates the *class-prior probability* and y represents the true class label of instance $\boldsymbol{x}$. The same as conventional supervised learning, PU learning aims to learn a classifier $p(y = 1|\boldsymbol{x})$ which distinguishes positive and negative data.

2.2 Interpolation Consistency Regularization

The mixup operator was first introduced by [18] for supervised learning and can be defined as:

$$\mathrm{Mix}_\lambda(\boldsymbol{x}_a, \boldsymbol{x}_b) = \lambda \cdot \boldsymbol{x}_a + (1-\lambda) \cdot \boldsymbol{x}_b,$$

where $\boldsymbol{x}_a, \boldsymbol{x}_b$ are two labeled examples and the coefficient λ is sampled from the Beta distribution. Later, [20] adapts mixup to semi-supervised learning and applies interpolation between unlabeled datapoints. Interpolation-based consistency trains a prediction model f_θ to provide consistent predictions at interpolations of unlabeled points:

$$f_\theta\left(\mathrm{Mix}_\lambda\left(\boldsymbol{x}_j, \boldsymbol{x}_k\right)\right) \approx \mathrm{Mix}_\lambda\left(f_{\theta'}\left(\boldsymbol{x}_j\right), f_{\theta'}\left(\boldsymbol{x}_k\right)\right),$$

where $\boldsymbol{x}_j, \boldsymbol{x}_k$ are a pair of unlabeled examples and θ' is a moving average of the network parameter θ. The interpolation-based consistency can be seen as encouraging the model to have strictly linear behavior “between” examples, by requiring that the model's output for a convex combination of two inputs is close to the convex combination of the output for each individual input. The mixup regularizer and consistency loss have not been previously investigated in PU learning and it is interesting to investigate its efficiency.

3 Method

3.1 Consistency Regularization for PU Learning

During training phase of neural networks f_θ, given a batch $\mathcal{X}_\mathcal{P} \subset \mathcal{P}$ of positively labeled examples with corresponding targets and an equally-sized batch $\mathcal{X}_\mathcal{U} \subset \mathcal{U}$ of unlabeled examples, MIXPUL produces a processed batch of augmented unlabeled examples $\mathcal{X}'_\mathcal{U}$ with “soft” labels using mixup. $\mathcal{X}'_\mathcal{U}$ is then used in computing the unsupervised consistency loss term. For each pair of two unlabeled samples and their “soft” labels $(\boldsymbol{x}_j, \hat{y}_j), (\boldsymbol{x}_k, \hat{y}_k)$ where $\hat{y}_j = f_{\theta'}\left(\boldsymbol{x}_j\right), \hat{y}_k = f_{\theta'}\left(\boldsymbol{x}_k\right)$, an augmented unlabeled datapoint $(\boldsymbol{x}', \hat{y}')$ is obtained as follows using mixup operator:

$$\boldsymbol{x}' = \lambda \boldsymbol{x}_j + (1-\lambda)\boldsymbol{x}_k \tag{1}$$

and

$$\hat{y}' = \lambda \hat{y}_j + (1-\lambda)\hat{y}_k, \tag{2}$$

where

$$\lambda \sim \mathrm{Beta}(\alpha, \alpha) \tag{3}$$

and α is a hyperparameter of the Beta distribution. In our implementations, we first collect unlabeled examples with their guessed labels into:

$$\mathcal{X}_\mathcal{U} = \{(\boldsymbol{x}_1, \hat{y}_1), \ldots, (\boldsymbol{x}_B, \hat{y}_B)\}. \tag{4}$$

and after performing mixup we get:

$$\mathcal{X}'_\mathcal{U} = \left\{\left(\mathrm{Mix}\left(\boldsymbol{x}_i, \boldsymbol{x}_{r_i}\right), \mathrm{Mix}\left(\hat{y}_i, \hat{y}_{r_i}\right)\right)\right\}_{i=1}^{B}, \tag{5}$$

where r is a random permutation of $[B]$. Notably, instead of predicting soft labels $\hat{y}_j$ and $\hat{y}_k$ in Equation (2) using network f_θ, we maintain a moving average θ' of parameter θ following [16, 19] and set $\hat{y}_j = f_{\theta'}(\boldsymbol{x}_j)$. Then, we perform $\mathcal{W} =$ Shuffle$(\mathcal{X}_\mathcal{U})$ which will serve as a data source for mixup. For each the i-th example and label pair $(\boldsymbol{x}_i, \hat{y}_i) \in \mathcal{X}_\mathcal{U}$ and $(\boldsymbol{x}_{r_i}, \hat{y}_{r_i}) \in \mathcal{W}$, we apply the mixup operator and add the result to the collection $\mathcal{X}'_\mathcal{U}$. Note that the interpolation is only applied between unlabeled datapoints so far and on each mini-batch we sample a random λ from $\mathrm{Beta}(\alpha, \alpha)$ for mixup. To summarize, by using mixup, $\mathcal{X}_\mathcal{U}$ is transformed into $\mathcal{X}'_\mathcal{U}$, a collection of multiple augmentations of each unlabeled example with corresponding “soft” label.

3.2 Reliable Negative Mining

It is noteworthy that applying mixup requires reasonably good “soft” labels, which is realized by training networks on labeled data in semi-supervised learning. In PU learning, it is unrealistic to train the networks by feeding only positive data. We alleviate this problem by identifying a subset of reliable negative (RN) examples from the unlabeled set. In this work, three different types of methods are considered.

- *Rand:* We construct a set of “pseudo” negative examples by randomly downsampling the unlabeled set.
- *Dist:* Unlabeled instances with the farthest averaged distance from positive data are selected as negative.
- *NTC:* An Non-Traditional Classifier (NTC) is trained to discriminate $\mathcal{P}$ and $\mathcal{U}$. Instances with the smallest prediction scores are selected as negative.

The positive set $\mathcal{P}$ and the selected reliable negative samples $\mathcal{N} \subset \mathcal{U}$ are used to compute supervised consistency loss. Even when combining a positive sample and a false negative sample the loss computed can still be useful as the positive sample contains the true label of the other one. It is noteworthy that MIXPUL does not reduce cross-entropy loss in case of overfitting. By leveraging consistency training, it can better explore the data space even when labeled data is scarce. We compare three RN mining methods in the experiments.

3.3 Objective Function

We describe each part of MIXPUL's objective function in the following. Given batch $\mathcal{X}_\mathcal{P}$, $\mathcal{X}_\mathcal{N}$, and $\mathcal{X}_\mathcal{U}$, we construct $\mathcal{X}'_{\mathcal{PN}}$ by applying mixup operator on $\mathcal{X}_\mathcal{P}$ and $\mathcal{X}_\mathcal{N}$. Similarly, $\mathcal{X}'_\mathcal{U}$ is formed by mixing unlabeled data $\mathcal{X}_\mathcal{U}$. We then combine the consistency loss and the margin loss. More formally, the combined loss $\mathcal{L}$ for our proposed MIXPUL is computed as Equation (6):

$$\mathcal{L} = \mathcal{L}_{\mathcal{PN}} + \beta \mathcal{L}_\mathcal{U} + \gamma \mathcal{L}_{\mathcal{PU}}. \tag{6}$$

The first two terms respectively represent the supervised and unsupervised interpolation-based consistency loss and can be written as:

$$\mathcal{L}_{\mathcal{PN}} = \frac{1}{|\mathcal{X}'_{\mathcal{PN}}|} \sum_{\boldsymbol{x}, \hat{y} \in \mathcal{X}'_{\mathcal{PN}}} \|\hat{y} - f_\theta(\boldsymbol{x})\|_2^2 \tag{7}$$

$$\mathcal{L}_{\mathcal{U}} = \frac{1}{|\mathcal{X}'_{\mathcal{U}}|} \sum_{\boldsymbol{x}, \hat{y} \in \mathcal{X}'_{\mathcal{U}}} \|\hat{y} - f_\theta(\boldsymbol{x})\|_2^2 \tag{8}$$

By imposing consistency loss, it regularizes the network to have strictly linear behavior. Using "soft" labels can also alleviate the problem of absence of negative data.

Since the consistency terms function as regularizers, we introduces a risk function, i.e., margin loss, between pairs of positive and unlabeled samples, which takes the following form:

$$\mathcal{L}_{\mathcal{PU}} = \frac{1}{|\mathcal{X}_\mathcal{P}| \cdot |\mathcal{X}_\mathcal{U}|} \sum_{\boldsymbol{x}_p \in \mathcal{X}_\mathcal{P}, \boldsymbol{x}_u \in \mathcal{X}_\mathcal{U}} |f_\theta(\boldsymbol{x}_u) - f_\theta(\boldsymbol{x}_p) + \eta|_+ , \tag{9}$$

where $|z|_+$ returns z if $z > 0$, otherwise 0. η is the margin parameter. By imposing $\mathcal{L}_{\mathcal{PU}}$, MIXPUL is desired to produce higher prediction score for positive samples than unlabeled samples. It is shown that $\mathcal{L}_{\mathcal{PU}}$ can be viewed as an estimation of its supervised counterpart.

Finally, we use hyperparameters β and γ to trade-off these three terms. When optimizing Problem (6), we compute the gradient $\nabla_\theta \mathcal{L}$ and update θ using standard SGD or Adam. Then we update the exponential moving average θ' of network parameter θ following [16].

3.4 Theoretical Interpretation of Equation (6)

We further explain our objective function from the perspective of empirical risk minimization. If we regard the consistency loss $\mathcal{L}_{\mathcal{PN}}$ and $\mathcal{L}_\mathcal{U}$ in Equation (6) as regularizations, the last term $\mathcal{L}_{\mathcal{PU}}$ can be interpreted as a risk function. The consistency loss can move the decision boundary to low-density regions of the data distribution [19]. The pairwise ranking loss is designated for PU-AUC risk minimization. Inspired by [21], the risk function in AUC optimization from PU data is equivalent to the risk in supervised AUC optimization. Particularly, let $\ell_{01}(z)$ denote the zero-one loss which returns 1 if $z < 0$, 0.5 if $z = 0$, and 0 otherwise. Supposing that unlabeled data is sampled from a mixture of class distribution $P(\boldsymbol{x}|y = 0)$ and $P(\boldsymbol{x}|y = 1)$ completely at random, we show that PU-AUC risk R_{PU} is an equivalent estimation of PN-AUC risk R_{PN} as follows.

$$\begin{aligned}
R_{\mathrm{PU}} =& \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_\mathrm{P}} \mathop{\mathbb{E}}_{\boldsymbol{x}' \sim \mathcal{X}_\mathrm{U}} \ell_{01}\big(f(\boldsymbol{x}) - f(\boldsymbol{x}')\big) \\
=& \mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathcal{X}_\mathrm{P}} [\pi \mathop{\mathbb{E}}_{\overline{\boldsymbol{x}} \sim \mathcal{X}_\mathrm{P}} \ell_{01}\big(f(\boldsymbol{x}) - f(\overline{\boldsymbol{x}})\big) \\
& + (1-\pi) \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathcal{X}_\mathrm{N}} \ell_{01}\big(f(\boldsymbol{x}) - f(\hat{\boldsymbol{x}})\big)] \\
=& \pi \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_\mathrm{P}} \mathop{\mathbb{E}}_{\overline{\boldsymbol{x}} \sim \mathcal{X}_\mathrm{P}} \ell_{01}\big(f(\boldsymbol{x}) - f(\overline{\boldsymbol{x}})\big) \\
& + (1-\pi) \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_\mathrm{P}} \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathcal{X}_\mathrm{N}} \ell_{01}\big(f(\boldsymbol{x}) - f(\hat{\boldsymbol{x}})\big)
\end{aligned}$$

The above equation holds thanks to the linearity of expectation. Note that the first term at the right hand side of the equation is a constant which equals to $\frac{\pi}{2}$. Therefore, it can be omitted in the optimization. Surprisingly, the second term coincides with PN-AUC risk R_{PN}. In other words, we get:

$$R_{\mathrm{PU}} = (1-\pi)R_{\mathrm{PN}} + \frac{\pi}{2}$$

It is effortless to see that R_{PU} is a linear transformation of R_{PN}. During training, the zero-one loss is usually replaced with a surrogate loss for the convenience of optimization. In our implementation, we substitute it with margin loss defined in Equation (9) which is enough for AUC risk optimization when it gets minimized.

4 Experiments

To validate the superiority of MIXPUL, we conduct experiments on the MNIST[1], CIFAR-10[2], and UCI datasets [3]. The comprehensive statistics of used datasets are listed in Table 1. Notably, the class-ratio of each dataset is the percentage of positive examples among training data.

Dataset	#Train	#Test	#Feature	Class-ratio
ethn	1,840	790	30	0.50
krvskp	2,237	959	36	0.49
titanic	1,540	661	3	0.32
spambase	3,220	1,381	57	0.40
MNIST	60,000	10,000	784	0.49
CIFAR-10	50,000	10,000	3,072	0.40

Table 1: Dataset statistics

4.1 Implementation Details

Unless otherwise noted, in all experiments we use the multilayer perceptron. We simply evaluate models using an exponential moving average of their parameters with a decay rate of 0.999. We find in practice that most of MIXPUL's hyperparameters can be fixed and do not need to be tuned on a per-experiment or per-dataset basis. Specifically, for all experiments, we respectively set the hyperparameters β and γ the objective function of MIXPUL to 1 and 1 for simplicity. Further, we only change and α on a per-dataset basis; we found that $\alpha = 1$ are good starting points for tuning. We used the SGD with nesterov momentum optimizer for all of our experiments. For the experiments in Table 1 and Table 2, we run the experiments for 200 epochs. The initial learning rate was set to 10^{-5} on CIFAR-10 and 10^{-3} for other datasets. The momentum parameter was set to 0.9. We used a L_2 regularization coefficient 10^{-4} and a batch-size of 128 in our experiments. All the experiments were done with Pytorch[4].

4.2 Competing Methods

The following methods are compared:

- *Supervised*: This method trains a supervised classifier with lightGBM [22]. It treats unlabeled data as negative and uses hyperopt [23] for hyperparameter optimization.
- *WSVM*: The method of [1]. This method treats each unlabeled instance as a combination of positive and negative examples.
- *Ramp*: The method of [8] through optimizing ramp loss. This method is used for comparison with MIX-PUL on MNIST dataset.
- *uPU*: The method of [9] using unbiased PU learning risk estimator.
- *nnPU*: The method of [10] using non-negative unbiased PU learning risk estimator. It is an improved version of uPU which usually overfits because the value of uPU loss can become negative.
- *PNU*: The method of [24] which explicitly optimizes AUC.

[1]http://yann.lecun.com/exdb/mnist/
[2]https://www.cs.toronto.edu/ kriz/cifar.html
[3]https://archive.ics.uci.edu/ml/datasets.php
[4]https://pytorch.org/

Setting	Method	1,200	2,400	3,600
0 vs. 1	Ramp	3.36	4.85	5.48
	MIXPUL	**0.24**	**0.33**	**0.33**
0 vs. 2	Ramp	5.15	6.96	7.22
	MIXPUL	**4.57**	**2.83**	**2.14**
0 vs. 3	Ramp	3.49	4.72	5.02
	MIXPUL	**3.02**	**2.41**	**2.46**
0 vs. 4	Ramp	1.68	2.05	2.21
	MIXPUL	**0.76**	**0.66**	**0.46**
0 vs. 5	Ramp	5.21	7.22	**7.46**
	MIXPUL	**1.60**	**3.63**	9.62
0 vs. 6	Ramp	11.47	19.87	22.58
	MIXPUL	**8.57**	**5.31**	**3.51**
0 vs. 7	Ramp	1.89	2.55	2.64
	MIXPUL	**1.64**	**1.20**	**1.15**
0 vs. 8	Ramp	3.98	4.81	4.75
	MIXPUL	**3.58**	**2.81**	**2.41**
0 vs. 9	Ramp	1.22	1.60	**1.73**
	MIXPUL	**1.21**	**0.96**	3.02

Table 2: Misclassification rate (in percent) of MIXPUL and Ramp on MNIST dataset. We set the amount of positive data $|\mathcal{P}|$ from $\{1200, 2400, 3600\}$. The best results are in bold.

4.3 Results on MNIST

The model for MNIST is a 3-layer multilayer perceptron (MLP) with ReLU activation function. MNIST has 10 classes originally, and we constructed the $\mathcal{P}$ and $\mathcal{N}$ classes from them as follows: MNIST was preprocessed in such a way that 0 constitute the positive class, while one of {1, 2, 3, 4, 5, 6, 7, 8, 9} constitutes the negative class separately in each experimental setting. Subsequently, we randomly sample part of $\mathcal{P}$ which is denoted by $\mathcal{P}'$. We set $\mathcal{P} = \mathcal{P} \setminus \mathcal{P}'$ and $\mathcal{U} = \mathcal{N} \cup \mathcal{P}'$ to form a PU dataset. We compare Ramp [8] which optimizes ramp loss calculated on positive and unlabeled data with the knowledge of true class-prior. The comparison results are reported in Table 2. It is noted that MIXPUL achieves lowest misclassification rate in comparison with Ramp [8] in most settings we studied. Specifically, MIXPUL significantly reduces the classification error rate from 5.48 to 0.33 in 0 vs. 1 task and from 22.58 to 3.51 in 0 vs. 6 task. It indicates that the model initialization for MIXPUL produces considerably accurate "soft" labels facilitating the consistency loss. In summary, the empirical studies demonstrate that MIXPUL is insensitive to class-prior and consistently outperforms Ramp which is one of the representative approaches based on unbiased risk minimization.

4.4 Results on CIFAR-10

We compare our method with state-of-the-art PU learning algorithms on CIFAR-10 dataset. We use the same architecture for all methods as specified in [10]. CIFAR-10 has 10 classes originally, and we construct the *positive* class and *negative* class as follows. The *positive* class is formed by "airplane", "automobile", "ship", and "truck", and the *negative* class is formed by "bird", "cat", "deer", "dog", "frog", and "horse". The results are reported in Table 3. We find that *uPU* is very prone to overfitting and we therefore use a small number of epoch (less than 10). When only 100 positive examples are available, both *nnPU* and *uPU* tend to treat all unlabeled data as negative, which prevent the model from training. Our consistency-regularized model achieve the lowest (best) classification error in 5 out of 6 settings. The comparison result is especially encouraging, considering that *nnPU* and *uPU* use the knowledge of class-prior.

Method	100	500	1k	2k	4k	10k
nnPU	40.00	15.55	13.32	11.32	9.90	**8.87**
uPU	40.00	26.01	19.95	14.38	12.08	9.90
MIXPUL	**22.63**	**14.00**	**12.21**	**11.10**	**9.77**	8.88

Table 3: Error rate on *CIFAR-10* dataset with varying number of positively labeled data.

Dataset	Method	0.01	0.05	0.1	0.2	0.4
ethn	Supervised	0.50±0.00	0.69±0.10	0.89±0.02	0.92±0.01	0.96±0.00
	WSVM	0.56±0.20	**0.96±0.01**	**0.98±0.00**	**0.99±0.00**	**0.99±0.00**
	uPU	0.66±0.07	0.84±0.04	0.94±0.01	0.96±0.00	0.97±0.00
	nnPU	0.63±0.07	0.80±0.09	0.82±0.03	0.86±0.03	0.86±0.01
	PNU	0.71±0.04	0.92±0.01	0.94±0.01	0.95±0.01	0.97±0.00
	MIXPUL	**0.73±0.04**	0.92±0.01	0.97±0.00	0.98±0.00	0.94±0.00
krvskp	Supervised	0.50±0.00	0.81±0.06	0.87±0.07	**0.97±0.01**	**0.98±0.00**
	WSVM	0.61±0.07	0.77±0.06	0.81±0.05	0.85±0.04	0.88±0.05
	uPU	0.72±0.08	0.85±0.06	0.89±0.04	0.95±0.02	0.96±0.02
	nnPU	0.62±0.09	0.78±0.03	0.82±0.06	0.84±0.02	0.88±0.04
	PNU	**0.72±0.07**	0.88±0.03	0.91±0.03	0.95±0.09	0.96±0.00
	MIXPUL	0.70±0.08	**0.90±0.03**	**0.93±0.02**	0.96±0.00	0.97±0.00
titanic	Supervised	0.50±0.00	0.50±0.00	0.69±0.02	0.72±0.01	**0.71±0.00**
	WSVM	0.35±0.04	0.68±0.05	**0.73±0.01**	**0.73±0.02**	0.71±0.02
	uPU	0.64±0.09	**0.70±0.01**	0.71±0.00	0.71±0.00	**0.71±0.00**
	nnPU	0.63±0.05	0.70±0.03	0.70±0.02	0.71±0.01	0.71±0.02
	PNU	0.63±0.08	0.69±0.00	0.70±0.02	0.70±0.00	0.70±0.00
	MIXPUL	**0.67±0.01**	0.68±0.01	0.70±0.01	0.71±0.01	**0.71±0.00**
spambase	Supervised	0.50±0.00	0.89±0.02	0.90±0.01	0.93±0.02	0.95±0.01
	WSVM	0.36±0.01	0.58±0.00	0.72±0.00	0.79±0.00	0.85±0.00
	uPU	0.87±0.05	0.91±0.01	0.93±0.00	0.93±0.01	0.94±0.00
	nnPU	0.77±0.07	0.87±0.01	0.90±0.00	0.91±0.01	0.92±0.00
	PNU	0.76±0.07	0.87±0.01	0.91±0.01	0.93±0.00	0.94±0.00
	MIXPUL	**0.89±0.01**	**0.92±0.01**	**0.94±0.01**	**0.94±0.01**	**0.96±0.00**

Table 4: Experimental comparisons on benchmark datasets with varying class-frequency. On each dataset, 10 test runs were conducted. The average AUC and standard deviation are presented. The true value of class-prior is used in uPU and nnPU. The best results in each setting are in bold. MIXPUL (ours) achieves competitive performance.

4.5 Results on UCI Datasets

To simulate PU learning problems in the wild, we construct PU data with varying class-frequency c. More specifically, we run all competing methods by setting class-frequency c to $c' \in \{0.01, 0.05, 0.1, 0.2, 0.4\}$ through randomly downsampling positive examples and appending them into the unlabeled set $\mathcal{U}$. For each c', we repeat the experiment 10 times and report the average performance. The comparison results in terms of AUC are shown in Table 4, where means and standard deviations of testing performance based on 10 random samplings are reported. It is effortless to see that the supervised baseline performs dreadfully with a modest set of positive examples. When more and more positive examples are observed, it achieves competing results because sampled unlabeled examples are becoming more likely to be negative. This validates that the supervised baseline is considerably good and should be compared in PU learning literature. We implement WSVM using Gaussian kernel and it fits the data very well in most cases except when the number of positive examples is extremely limited. Since deep neural networks are used in nnPU, it is no surprise that its performance on small datasets (e.g., ethn, krvskp) is usually worse than other algorithms owing to the lack of labeled examples. It is interesting to observe that the AUC score of comparison methods is approaching 1.0 even when $c = 0.01$ on *spambase*, which indicates that this dataset is relatively easier to deal with. It is noteworthy that MIXPUL is able to achieve superior or comparable results with uPU and nnPU even though they use the true value of class-prior π especially when $|\mathcal{P}|$ is small. In summary, our MIXPUL can readily adapted across many practical tasks without the knowledge of class-prior and assumptions over the data distribution.

4.6 Ablation Studies and Discussion

In the following, we provide an analysis of the effects of different parts of objective function and RN mining methods.

Method	5	25	50	100
Rand	**0.67±0.01**	**0.68±0.01**	**0.70±0.01**	**0.71±0.01**
Dist	0.48±0.14	0.53±0.16	0.66±0.01	0.47±0.15
NTC	0.56±0.11	0.68±0.02	**0.70±0.01**	0.69±0.00

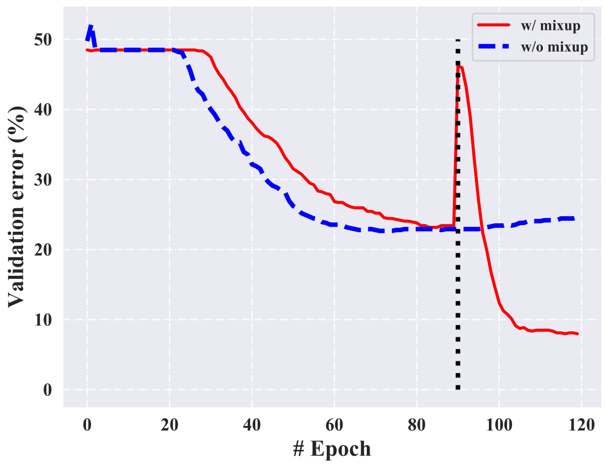
Table 5: A comparison between negative mining methods on *titanic* dataset with different amount of positive data $|\mathcal{P}| \in \{5, 25, 50, 100\}$.

4.6.1 How Does the Type of RN Mining Affect Results?

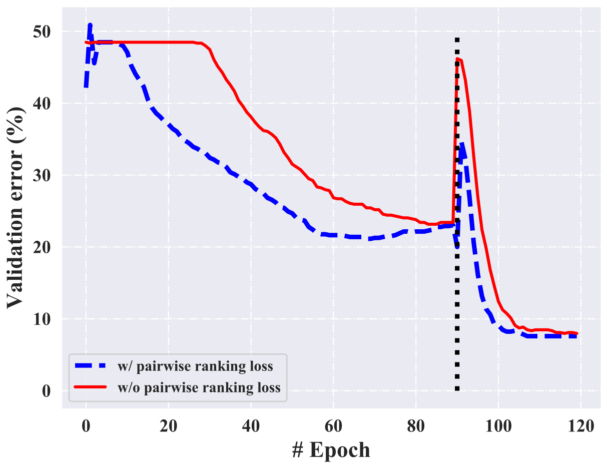
We report numerical results of employing three different reliable negative mining methods by fixing other components of the networks in Table 5. Euclidean distance is used in *Dist* method. For *NTC* method, we train a random forest classifier. It is effortless to observe that *Dist* has the worst performance. This indicates that it is unsafe to use distance-based classifiers for unknown data distribution. Therefore, we use *Rand* in all experiments for its observed good performance. We also tried to train the networks without RN examples. Since only positive examples are fed, the networks suffer from overfitting after a few epochs.

4.6.2 How Much Does Unsupervised Mixup Matter By Itself?

We study the effect of mixup by training the networks with and without unsupervised interpolation-based consistency loss on one of the image datasets, *ethn*. As shown in Figure 2a, the vertical dashed line indicates the iteration where interpolation training begins. The red and blue lines respectively demonstrate the classification error with and without unsupervised mixup. If we apply the mixup operator on unlabeled data, the misclassification rate initially increases very fast because augmented data is generated which the networks have never seen. After a few iterations, the error rate decreases to less than 10% which is far smaller than the number without using unsupervised mixup. The results demonstrate the effectiveness of mixup and provide another way of employing unlabeled data for PU learning.



(a) Ablations on mixup



(b) Ablations on margin loss

Figure 2: Ablation studies on unsupervised mixup (left) and margin loss (right).

4.6.3 How Does Margin Loss Affect Results?

In our objective function, we use margin loss between positive and unlabeled example pairs guiding the model to give higher prediction scores for observed positive examples than unlabeled ones. As shown in Figure 2b, it results in faster convergence by incorporating margin loss. We also observe that the model has a more stable performance when the mixup is applied as illustrated by the vertical dashed line. The results coincide with our theoretical analysis that the margin loss leads to AUC risk minimization.

In a brief summary, we provide an analysis of the effects of mixup, margin loss, and the negative example mining methods for researchers and practitioners. We find that mixup is more effective on image datasets than other types of data and it also provides a new way of using unlabeled instances for PU learning. Further, by incorporating margin loss, it leads to faster convergence for the networks and makes the model performs more stably when applying mixup. Finally, randomly downsampling unlabeled data as negative is an effective RN mining method for PU learning.

5 Conclusion

In this work, we introduce MIXPUL which applies interpolation-based consistency regularization to PU learning. MIXPUL has two advantages over previous PU learning approaches. First, it does not require the knowledge of class-prior, which otherwise hinders the applicability of the algorithm. Second, through extensive experiments, we find that MIXPUL exhibited significantly performance improvements over prior state of the art. Besides, we empirically observe that the proposed negative mining techniques are considerably effective, without which the model corrupts and is unable to incorporate consistency regularization. Specifically, different negative example mining techniques are further investigated and we find the randomized method work very well. We also conduct thorough ablation studies on the consistency regularization and the margin loss. We hope that the proposed consistency regularization will become a standard element in PU learning, and that it will make things easier and simpler for researchers and practitioners.

References

[1] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV*, pages 213–220, 2008.

[2] Huihui Wei and Ming Li. Positive and unlabeled learning for detecting software functional clones with adversarial training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden*, pages 2840–2846, 2018.

[3] Peng Yang, Xiaoli Li, Hon-Nian Chua, Chee-Keong Kwoh, and See-Kiong Ng. Ensemble positive unlabeled learning for disease gene identification. *PloS one*, 9(5), 2014.

[4] Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. Partially supervised classification of text documents. In *Proceedings of the 19th International Conference on Machine Learning, Sydney, Australia*, pages 387–394, 2002.

[5] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico*, pages 587–592, 2003.

[6] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *Machine Learning, Proceedings of the 30th International Conference, Washington, DC*, pages 448–455, 2003.

[7] Hong Shi, Shaojun Pan, Jian Yang, and Chen Gong. Positive and unlabeled learning via loss decomposition and centroid estimation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden*, pages 2689–2695, 2018.

[8] Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Advances in Neural Information Processing Systems 27, Montreal, Canada*, pages 703–711, 2014.

[9] Marthinus Christoffel du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *Proceedings of the 32nd International Conference on Machine Learning, Lille, France*, pages 1386–1394, 2015.

[10] Ryuichi Kiryo, Gang Niu, Marthinus Christoffel du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Advances in Neural Information Processing Systems 30, Long Beach, CA*, pages 1674–1684, 2017.

[11] Aditya Krishna Menon, Brendan van Rooyen, Cheng Soon Ong, and Bob Williamson. Learning from corrupted binary labels via class-probability estimation. In *Proceedings of the 32nd International Conference on Machine Learning, Lille, France*, volume 37, pages 125–134.

[12] Harish G. Ramaswamy, Clayton Scott, and Ambuj Tewari. Mixture proportion estimation via kernel embeddings of distributions. In *Proceedings of the 33rd International Conference on Machine Learning, New York City, NY*, pages 2052–2060, 2016.

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

[14] Jessa Bekker and Jesse Davis. Estimating the class prior in positive and unlabeled data through decision tree induction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA*, pages 2712–2719, 2018.

[15] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems 29, Barcelona, Spain*, pages 1163–1171, 2016.

[16] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in Neural Information Processing Systems 30, Long Beach, CA*, pages 1195–1204, 2017.

[17] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2018.

[18] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of the 6th International Conference on Learning Representations, Vancouver, Canada*, 2018.

[19] Vikas Verma, Alex Lamb, Juho Kannala, Yoshua Bengio, and David Lopez-Paz. Interpolation consistency training for semi-supervised learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China*, pages 3635–3641, 2019.

[20] David Berthelot, Nicholas Carlini, Ian J. Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *CoRR*, abs/1905.02249, 2019.

[21] Zheng Xie and Ming Li. Semi-supervised AUC optimization without guessing labels of unlabeled data. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA*, pages 4310–4317, 2018.

[22] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30, Long Beach, CA*, pages 3149–3157, 2017.

[23] Brent Komer, James Bergstra, and Chris Eliasmith. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In *International Conference on Machine Learning Workshop on AutoML*, volume 9, 2014.

[24] Tomoya Sakai, Gang Niu, and Masashi Sugiyama. Semi-supervised auc optimization based on positive-unlabeled learning. *Machine Learning*, 107(4):767–794, 2018.