Cost-Sensitive Positive and Unlabeled Learning

Xiuhua Chen[a], Chen Gong[a,*], Jian Yang[a,b]

[a]*PCA Lab, the Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, School of Computer Science and Engineering, Nanjing University of Science and Technology, China*
[b]*Jiangsu Key Lab of Image and Video Understanding for Social Security*

Abstract

Positive and Unlabeled learning (PU learning) aims to train a binary classifier solely based on positively labeled and unlabeled data when negatively labeled data are absent or distributed too diversely. However, none of the existing PU learning methods takes the class imbalance problem into account, which significantly neglects the minority class and is likely to generate a biased classifier. Therefore, this paper proposes a novel algorithm termed "**C**ost-**S**ensitive **P**ositive and **U**nlabeled learning" (CSPU) which imposes different misclassification costs on different classes when conducting PU classification. Specifically, we assign distinct weights to the losses caused by false negative and false positive examples, and employ double hinge loss to build our CSPU algorithm under the framework of empirical risk minimization. Theoretically, we analyze the computational complexity, and also derive a generalization error bound of CSPU which guarantees the good performance of our algorithm on test data. Empirically, we compare CSPU with the state-of-the-art PU learning methods on synthetic dataset, OpenML benchmark datasets, and real-world datasets. The results clearly demonstrate the superiority of the proposed CSPU to other comparators in dealing with class imbalanced tasks.

Keywords: positive and unlabeled learning (PU learning), class imbalance, cost-sensitive learning, generalization bound

[*]Corresponding author
Email addresses: chenxh@njust.edu.cn (Xiuhua Chen), chen.gong@njust.edu.cn (Chen Gong), csjyang@njust.edu.cn (Jian Yang)

1. Introduction

Positive and Unlabeled learning (PU learning) [1] has gained increasing popularity in recent years due to its usefulness and effectiveness for practical applications, of which the target is to train a binary classifier from only positive and unlabeled data. Here the unlabeled data might be positive or negative, but the learning algorithm does not know their groundtruth labels during the training stage.

Since the training of a PU classifier does not depend on the explicit negative examples, it is preferred when the negative data are absent or distributed too diversely. For example, in information retrieval, the user-provided information constitutes the positive data, while the databases are regarded as unlabeled as they contain both similar and dissimilar information to the users query [2]. In this application, negative examples are unavailable, and thus PU learning can be utilized to find the users interest in the unlabeled set. In addition, in a remotely-sensed hyperspectral image, we may only be interested in identifying one specific land-cover type for certain use without considering other types [3]. In this case, we may directly treat the type-of-interest as positive and leave the remaining ones as negative, so PU learning can be employed to detect the image regions of positive land-cover type.

Existing PU learning algorithms can be mainly divided into three categories based on how the unlabeled data are treated. The first category [4, 5] initially identifies some reliable negative data in unlabeled data, and then invokes a traditional classifier to perform ordinary supervised learning. The result of such two-step framework is severely dependent on the precision of the identified negative data. That is, if the detection of negative data is inaccurate, the final outcome could be disastrous. To handle this shortcoming, the second category [6, 7, 8] directly treats all unlabeled data as negative and takes PU learning as a label noise learning problem (the definition of label noise learning can be found in [9]), among which the original positive examples are deemed as mislabeled as negative data. The last but also the most prevalent category [10, 11, 12] in recent years focuses on designing various unbiased risk estimators. The approaches of this category apply distinct loss functions that satisfy specific conditions to PU risk estimators, and result in various unbiased risk estimators. A breakthrough in this direction is [10] for proposing the first unbiased risk estimator with a nonconvex loss function $\ell(z)$ such that $\ell(z) + \ell(-z) = 1$ (e.g., ramp loss $\ell_{\mathrm{R}}(z) = \frac{1}{2}\max(0, \min(2, 1 - z)))$ with z being the variable. Furthermore, a more general and consistent unbiased estimator was proposed in [11] which advances a novel double hinge loss $\ell_{\mathrm{DH}}(z) = \max(-z, \max(0, \frac{1}{2} - \frac{1}{2}z))$ so that the composite loss $\tilde{\ell}(z) = \ell_{\mathrm{DH}}(z) - \ell_{\mathrm{DH}}(-z)$ satisfies $\tilde{\ell}(z) = -z$ after normalization. After that, a non-negative unbiased risk estimator suggested in [12] converts negative part of empirical risks in [11] into zero to avoid overfitting.

Although the methods mentioned above have received encouraging performances on various datasets or tasks, these methods would fail when encountering class imbalanced situations. In practical applications, the class imbalanced phenomena are prevalent, such as credit card fraud detection, disease diagnosis, and outlier detection, etc. In outlier detection, the very few outliers identified by the primitive detector constitute the positive set, and the remaining data points are deemed as unlabeled because some outliers are probably hidden among them. Moreover, the outliers usually occupy a small part of the entire dataset when compared with the inliers, which results in a class imbalanced PU learning problem. Unfortunately, none of the existing PU learning methods takes the class imbalance problem into consideration, so they are all likely to classify every example into the majority class (e.g., inlier) to acquire high classification accuracy. As a result, the influence of minority class (e.g., outlier) will be overwhelmed by the majority class [13] in deciding the decision function, and thus the biased classifier will be generated. This is obviously undesirable as the minority class usually contains our primary interest.

To enable PU learning to applicable to the imbalanced data, in this paper, we propose a novel algorithm dubbed "**C**ost-**S**ensitive **P**ositive and **U**nlabeled learning" (CSPU) which is convex and depends on the widely-used unbiased double hinge loss [11]. To be specific, we cast PU learning as an empirical risk minimization problem, in which the losses incurred by false negative and false positive examples are assigned distinct weights. As a result, the generated decision boundary can be calibrated to the potentially correct one. We show that our CSPU algorithm can be converted into a traditional Quadratic Programming (QP) problem, so it can be easily solved via off-the-shelf QP optimization toolbox. Theoretically, we analyze the computational complexity of our CSPU algorithm, and derive a generalization error bound of the algorithm based on its Rademacher complexity. Thorough experiments on various practical imbalanced datasets demonstrate that the proposed CSPU is superior to the state-of-the-art PU methods in terms of the F-measure metric [14, 15]. The main contributions of our work are summarized as follows:

- We propose a novel learning setting called "Cost-Sensitive PU learning" (CSPU) to model the practical problems where the absence of negative data and the class imbalance problem co-occur.
- We design a novel algorithm to address the CSPU learning problem, which introduces a convex empirical risk estimator with double hinge loss, and an efficient optimization method is also provided to solve our algorithm.
- We analyze the computational complexity of our algorithm, which takes $\mathcal{O}(9n^3 + 15n^2 + 7n + 1)$. We also derive a generalization error bound of the algorithm based on its Rademacher complexity, which reveals that the generalization error converges to the expected classification risk with the order of $\mathcal{O}(1/\sqrt{n_{\mathrm{p}}} + 1/\sqrt{n_{\mathrm{u}}} + 1/\sqrt{n})$ (n, n_{p}, and n_{u} are the amounts of training data, positive data, and unlabeled data correspondingly).
- We achieve the state-of-the-art results when compared with other PU learning methods in dealing with class imbalanced PU learning problem.

The rest of this paper is organized as follows. In Section 2, the related works of PU learning and imbalanced data learning are reviewed. Section 3 introduces the proposed CSPU algorithm. The optimal solution of our CSPU is given in Section 4. Section 5 studies the computational complexity and derives a generalization error bound of the proposed algorithm. The experimental results of our CSPU and other representative PU comparators are presented in Section 6. Finally, we draw a conclusion in Section 7.

2. Related Work

In this section, we review the representative works of PU learning and imbalanced data learning, as these two learning frameworks are very relevant to the topic of this paper.

2.1. PU Learning

PU learning has attracted a great amount of attention due to its practical value and has been well studied in recent years. The study of PU learning can be traced back to [4, 16], which follows a two-stage strategy. The first stage aims to identify a set of reliable negative examples from the unlabeled set and the second stage is to perform a traditional supervised learning (e.g., SVM) based on the detected negative examples as well as the original positive examples. In the first stage, Liu et al. [16] built a nave Bayesian classifier by taking the unlabeled data as negative, and then utilized the classifier to figure out the definite negative examples in the unlabeled data. Besides, Liu et al. [4] employed a "spy" technique that first puts a set of randomly selected positive examples to the unlabeled set and then determines a probability threshold to recognize possible negative examples from the unlabeled data. Moreover, Yu et al. [17] adopted the 1-DNF (Disjunctive Normal Form) technique to accomplish this stage for web page classification. The drawback of this two-stage strategy is that the detection of negative examples cannot always be accurate, which may lead to unsatisfactory performance.

Therefore, some researchers consider converting PU learning into one-sided label noise learning, in which the original positive examples in the unlabeled data are regarded as negatively labeled by mistake. For example, Lee and Liu [6] directly treated all unlabeled data as noisy negatively labeled data, and then performed Logistic regression after weighting the examples to learn a linear function. Differently, Shi et al. [7] explicitly modeled the label noise in the negative set (i.e., the original unlabeled set) by decomposing the hinge loss function, and shown that the adverse effect caused by label noise can be reduced by estimating the centroid of negative examples. Based on this algorithm, Gong et al. [18] proposed a kernelized algorithm to enable their method to tackle the non-linear cases.

However, the performances of abovementioned methods [6, 7, 18] are heavily influenced by the number of positive examples hidden in the unlabeled set. If the unlabeled set contains many positive examples, these methods [6, 7, 18] may not work well. Therefore, the recent state-of-the-art methods are centered on finding suitable risk estimators for PU learning. For instance, Elkan and Noto [19] first trained a nontraditional classifier on positive data and unlabeled data to estimate the weights of examples on a validation set, and then utilized a weighted SVM to estimate the positive class probability. In addition, Du Plessis et al. [10] developed a nonconvex ramp loss, with which the learning under the PU setting would give the same classification boundary as in the ordinary supervised setting. Since training a classifier with a nonconvex loss is usually difficult in practice, Du Plessis et al. [11] devised a convex unbiased formulation for PU classification, of which the key idea is to use an ordinary convex loss function for unlabeled data and a composite convex loss function for positive examples. Since the hinge loss is not permissible here, the double hinge loss is presented as an alternative. Unfortunately, the unbiased risk estimators will generate negative empirical risks, which may result in serious overfitting problem. To alleviate this phenomenon, Kiryo et al. [12] suggested a novel nonnegative risk estimator which lower-bounds the double hinge loss mentioned above. Since these methods need to estimate class prior, Zhang et al. [20] recently proposed a new PU learning strategy by formulating PU as an optimization problem without pre-estimating the class prior in an isolated step.

Apart from above methods, other representative PU methods include large-margin theory based PUSVM [21], multi-manifold based PU approach [22], the scalable kernelized PU algorithm [2], and PU learning via label disambiguation [23].

2.2. Imbalanced Data Learning

Generally, previous works on learning with imbalanced data mainly fall into two categories: data re-sampling [24, 25] and classifier design [26, 27]. The former category aims to alter the original data distribution by either over-sampling the minority class or under-sampling the majority class to make the re-sampled data distribution balanced. The latter category, instead of manipulating the examples at data level, operates at the classifier (algorithmic) level by directly modifying the learning procedure to improve the sensitivity of the classifier towards minority class.

Among early data level methods, a well-known issue with replication-based random over-sampling is its tendency to cause overfitting. More radically, it expands the training set without increasing any additional information. To remedy this situation, an advanced sampling approach called SMOTE (Synthetic Minority Over-sampling TEchnique) [24, 28] was proposed to generate new non-replicated examples by interpolating the minority class examples in their neighboring areas. Some extensions of this technique are also proposed. For example, safe-level SMOTE [29] carefully sampled minority examples with different weight degrees. Local neighborhood SMOTE [30] exploited the information from the local neighborhood of the considered examples. k-means based SMOTE [31] avoided the generation of noise by effectively overcoming the data imbalance between and within classes. However, the broadened decision regions generated by these methods are still error-prone due to the synthesizing of noisy and borderline examples. Therefore, under-sampling is preferred in some situations [32, 33], although potentially valuable information may be removed. Furthermore, the combination of over- and under-sampling procedures [25, 34] to balance the training data has also gained favorable performances. However, above methods are not applicable to PU learning as they all need complete supervised information to over-sample the minority class or under-sample the majority class, which cannot be accommodated to PU learning that works under partial supervision.

The classifier level approaches include thresholding, cost-sensitive learning, and ensemble learning, etc. Thresholding adjusts the decision threshold of a classifier in the test phase and thus changing the output class probabilities. Similarly, Domingos [35] re-labeled the classes of the training examples according to a modified threshold and then used the relabeled training examples to build a traditional supervised classifier. Zhou and Liu [36] first generated real-valued label outputs with a trained neural network, and then multiplied every output by the costs of misclassifying the corresponding class to other classes to obtain the calibrated classifier. Differently, cost-sensitive learning methods directly assign distinct costs (i.e., weights) to misclassification of examples from different classes. For example, Zadrozny et al. [26] fed the cost-proportionate based weights to classification algorithm to enhance the performance. Masnadi-Shirazi and Vasconcelos [37] extended the hinge loss to the cost-sensitive setting, and derived the optimal cost-sensitive learning algorithm to find the minimizer of the associated risk. Liu et al. [14] decomposed F-measure optimization into a series of cost-sensitive classification problems, and investigated the cost-sensitive feature selection by generating and assigning different costs to each class. However, the threshold determination and weight estimation inherited by above methods are difficult for real-world situations, so ensemble learning based methods are developed which train multiple individual sub-classifiers, and then use voting or other integrating techniques to achieve better results. For instance, Sun et al. [27] first divided the imbalanced dataset into smaller balanced subsets and then utilized a classification algorithm to build classifiers on these subsets. Finally, the classification results of these classifiers for new data are combined by a specific ensemble rule. Recently, Yin et al. [38] proposed a novel model for imbalanced data classification by integrating sampling, data space construction, cost-sensitive learning, and ensemble learning in a principle way. In addition to the aforementioned approaches, other recent classifier-level approaches include [39, 40, 41].

From above review, we see that none of the existing PU methods is equipped with the cost-sensitive strategy to address the class imbalance problem. Therefore, we first favor the marriage of PU learning and cost-sensitive learning in this paper and propose a novel cost-sensitive PU learning paradigm.

3. The Proposed Algorithm

The target of PU learning is to train a binary classifier from only positive and unlabeled data. Our proposed algorithm aims to address the situations where the absence of negative training data and the class imbalance problem co-occur. These phenomena are prevalent in many real-world cases, such as outlier detection. In this section, we first provide the formal setting for the PU learning problem, and then propose our CSPU classification algorithm.

3.1. Problem Setting

We denote $\boldsymbol{X} \in \mathbb{R}^d$ ($d \in \mathbb{N}$ is data dimensionality) and $Y \in \{+1, -1\}$ as the input and output random variables, respectively. Besides, let $p(\mathbf{x}, y)$ be the *underlying joint density* of $(\boldsymbol{X}, Y)$, based on which the *class-conditional densities* regarding positive class and negative class can be written as $p_{\mathrm{p}}(\mathbf{x}) = p(\mathbf{x}|y = +1)$ and $p_{\mathrm{n}}(\mathbf{x}) = p(\mathbf{x}|y = -1)$ correspondingly, and $p(\mathbf{x})$ is the *marginal density* regarding the unlabeled data.

Furthermore, given $\pi = p(y = +1)$ as the *class-prior probability*, then we have $p(y = -1) = 1 - \pi$. Since an unlabeled set consists of positive and negative examples, we know that $p(\mathbf{x}) = \pi p(\mathbf{x}|y = +1) + (1 - \pi)p(\mathbf{x}|y = -1)$.

Suppose we have a dataset $\mathcal{X}$ that consists of a positive set $\mathcal{X}_{\mathrm{p}}$ and an unlabeled set $\mathcal{X}_{\mathrm{u}}$ (i.e., $\mathcal{X} = \mathcal{X}_{\mathrm{p}} \cup \mathcal{X}_{\mathrm{u}}$). $\mathcal{X}_{\mathrm{p}}$ and $\mathcal{X}_{\mathrm{u}}$ are independent and identically drawn as $\mathcal{X}_{\mathrm{p}} = \{\mathbf{x}_i^{\mathrm{p}}\}_{i=1}^{n_{\mathrm{p}}} \sim p_{\mathrm{p}}(\mathbf{x})$ and $\mathcal{X}_{\mathrm{u}} = \{\mathbf{x}_i^{\mathrm{u}}\}_{i=1}^{n_{\mathrm{u}}} \sim p(\mathbf{x})$ where n_{p} and n_{u} are the amounts of positive examples and unlabeled data. Furthermore, we denote n'_{p} and n'_{n} as the real numbers of positive and negative examples in $\mathcal{X}$ according to their groundtruth labels which are unknown to the classifier, and thus the class imbalance problem discussed in this paper refers to the situation $n'_{\mathrm{p}} \ll n'_{\mathrm{n}}$. Therefore, given the hypothesis space as $\mathcal{F}$, the target of PU learning is to find a suitable decision function $f \in \mathcal{F} : \mathbb{R}^d \to \mathbb{R}$ on $\mathcal{X}$ that assigns a label $\hat{y}$ to an unseen test example $\mathbf{x}$ as $\hat{y} = \mathrm{sign}(f(\mathbf{x})) \in \{+1, -1\}$. It has been widely acknowledged that the Bayes optimal classifier $f(\boldsymbol{X})$ can be obtained by minimizing the following classification risk [11], namely

$$\begin{aligned} R(f) &= \mathbb{E}_{(\boldsymbol{X},Y)\sim p(\mathbf{x},y)}[\ell_{0-1}(Yf(\boldsymbol{X}))] \\ &= \pi\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(f(\boldsymbol{X}))] + (1-\pi)\mathbb{E}_{\mathrm{n}}[\ell_{0-1}(-f(\boldsymbol{X}))], \end{aligned} \tag{1}$$

where $\mathbb{E}_{\mathrm{p}}[\cdot] = \mathbb{E}_{\boldsymbol{X}\sim p_{\mathrm{p}}}[\cdot]$, $\mathbb{E}_{\mathrm{n}}[\cdot] = \mathbb{E}_{\boldsymbol{X}\sim p_{\mathrm{n}}}[\cdot]$, and $\ell_{0-1}(\cdot)$ is the zero-one loss formulated as $\ell_{0-1}(z) = \frac{1}{2} - \frac{1}{2}\mathrm{sign}(z)$.

215 *3.2. PU Classification*

In the ordinary classification setting, thanks to the availability of positive and negative examples, the expectations $\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(f(\boldsymbol{X}))]$ and $\mathbb{E}_{\mathrm{n}}[\ell_{0-1}(-f(\boldsymbol{X}))]$ in (1) can be estimated by the corresponding example averages [11], namely

$$\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(f(\boldsymbol{X}))] = \frac{1}{n_{\mathrm{p}}}\sum_{i=1}^{n_{\mathrm{p}}}\ell_{0-1}(f(\mathbf{x}_i^{\mathrm{p}})) \tag{2}$$

and

$$\mathbb{E}_{\mathrm{n}}[\ell_{0-1}(-f(\boldsymbol{X}))] = \frac{1}{n_{\mathrm{n}}}\sum_{i=1}^{n_{\mathrm{n}}}\ell_{0-1}(-f(\mathbf{x}_i^{\mathrm{n}})), \tag{3}$$

where $\{\mathbf{x}_i^{\mathrm{n}}\}_{i=1}^{n_{\mathrm{n}}} \sim p_{\mathrm{n}}(\mathbf{x})$ and n_{n} is the number of negative examples. However, in the PU classification setting, no labeled negative examples are available for classifier training and therefore $\mathbb{E}_{\mathrm{n}}[\ell_{0-1}(-f(\boldsymbol{X}))]$ cannot be estimated directly. In spite of this, $\mathbb{E}_{\mathrm{n}}[\ell_{0-1}(-f(\boldsymbol{X}))]$ can be indirectly approximated according to [10, 11]. As $p(\mathbf{x}) = \pi p_{\mathrm{p}}(\mathbf{x}) + (1-\pi)p_{\mathrm{n}}(\mathbf{x})$, we can obtain that

$$\mathbb{E}_{\mathrm{u}}[\ell_{0-1}(-f(\boldsymbol{X}))] = \pi\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(-f(\boldsymbol{X}))] + (1-\pi)\mathbb{E}_{\mathrm{n}}[\ell_{0-1}(-f(\boldsymbol{X}))], \tag{4}$$

where $\mathbb{E}_{\mathrm{u}}[\cdot]$ denotes the expectation over $p(\mathbf{x})$. Therefore, the risk $R(f)$ can be approximated by

$$\begin{aligned} R(f) &= \pi\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(f(\boldsymbol{X}))] + \mathbb{E}_{\mathrm{u}}[\ell_{0-1}(-f(\boldsymbol{X}))] - \pi\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(-f(\boldsymbol{X}))] \\ &= \pi\mathbb{E}_{\mathrm{p}}[\underbrace{\ell_{0-1}(f(\boldsymbol{X})) - \ell_{0-1}(-f(\boldsymbol{X}))}_{\text{Composite loss}}] + \mathbb{E}_{\mathrm{u}}[\ell_{0-1}(-f(\boldsymbol{X}))]. \end{aligned} \tag{5}$$

225 From Eq. (5), we can see that the risk of PU learning consists of a composite loss for positive examples and an ordinary loss for unlabeled data.

3.3. Cost-Sensitive PU Classification

In the traditional cost-sensitive classification setting, distinct weights are assigned to the losses incurred by false negative and false positive examples, which means

$$R(f) = \pi c_1\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(f(\boldsymbol{X}))] + (1-\pi)c_{-1}\mathbb{E}_{\mathrm{n}}[\ell_{0-1}(-f(\boldsymbol{X}))], \tag{6}$$

where c_1 is the false negative cost, c_{-1} is the false positive cost, and π is the *class-prior probability* defined in Section 3.1. Following [10, 11, 21], the *class-prior probability* can be estimated from positive and unlabeled data according to the approaches suggested in [42, 43] in real-world situations. In practical applications, we usually take the minority class as positive and the majority class as negative (i.e., $n'_{\mathrm{p}} \ll n'_{\mathrm{n}}$). Therefore, the cost of false negative is more expensive than that of false positive, that is, $c_1 > c_{-1} > 0$ and ideally $\dfrac{c_1}{c_{-1}} = \dfrac{n'_{\mathrm{n}}}{n'_{\mathrm{p}}}$. Consequently, similar to the derivations for Eq. (5), the proposed CSPU classification risk can be expressed as

$$\begin{aligned} R(f) &= \pi c_1\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(f(\boldsymbol{X}))] + c_{-1}\mathbb{E}_{\mathrm{u}}[\ell_{0-1}(-f(\boldsymbol{X}))] - \pi c_{-1}\mathbb{E}_{\mathrm{p}}[\ell_{0-1}(-f(\boldsymbol{X}))] \\ &= \pi\mathbb{E}_{\mathrm{p}}[c_1\ell_{0-1}(f(\boldsymbol{X})) - c_{-1}\ell_{0-1}(-f(\boldsymbol{X}))] + c_{-1}\mathbb{E}_{\mathrm{u}}[\ell_{0-1}(-f(\boldsymbol{X}))]. \end{aligned} \tag{7}$$

Practically, it is intractable to optimize the discrete zero-one loss, so we usually employ a continuous surrogate loss function $\ell(z)$ to replace the original zero-one loss, which leads to

$$R(f) = \pi\mathbb{E}_{\mathrm{p}}[\underbrace{c_1\ell(f(\boldsymbol{X})) - c_{-1}\ell(-f(\boldsymbol{X}))}_{\text{Weighted composite loss}}] + c_{-1}\mathbb{E}_{\mathrm{u}}[\ell(-f(\boldsymbol{X}))]. \tag{8}$$

Note that the first term in Eq. (8) is a weighted composite loss $\hat{\ell}(z) = c_1\ell(z) - c_{-1}\ell(-z)$, and it will degenerate to the conventional composite loss $\hat{\ell}_0(z) = \ell(z) - \ell(-z)$ in [11] when $c_1 = c_{-1} = 1$. In [11], the authors proposed to employ double hinge loss as $\ell(z)$ as it makes the conventional composite loss $\hat{\ell}_0(z) = \ell(z) - \ell(-z)$ to be convex, so the entire objective function can be convex. The commonly adopted hinge loss is not applicable as it will lead to the non-convexity of $\hat{\ell}_0(z)$, which is undesirable for the subsequent optimization. In this paper, we also adopt the double hinge loss as our loss function. Therefore, here we need to prove that the double hinge loss also makes the weighted composite loss $\hat{\ell}(z) = c_1\ell(z) - c_{-1}\ell(-z)$ to be convex. The related theorem is presented below:

Theorem 1. *If the convex loss $\ell(z)$ makes the conventional composite loss $\hat{\ell}_0(z) = \ell(z) - \ell(-z)$ convex, the weighted composite loss $\hat{\ell}(z) = c_1\ell(z) - c_{-1}\ell(-z)$ is also convex.*

Proof. By computing the second order derivative of weighted composite loss $\hat{\ell}(z)$ to z, we have

$$\frac{\mathrm{d}^2}{\mathrm{d}z^2}\hat{\ell}(z) = c_1\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(z) - c_{-1}\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(-z). \tag{9}$$

Since $c_1 > c_{-1}$, we write $c_1 = c_{-1} + a$ where $a > 0$. Then, Eq. (9) is equivalent to

$$\begin{aligned}\frac{\mathrm{d}^2}{\mathrm{d}z^2}\hat{\ell}(z) &= (c_{-1} + a)\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(z) - c_{-1}\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(-z)\\ &= a\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(z) + c_{-1}(\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(z) - \frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(-z)).\end{aligned} \tag{10}$$

If a convex $\ell(z)$ enables $\hat{\ell}_0(z) = \ell(z) - \ell(-z)$ to be convex, we have that $\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(z) \geq 0$ and $\frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(z) - \frac{\mathrm{d}^2}{\mathrm{d}z^2}\ell(-z) \geq 0$ hold for all z. Besides, due to the fact that $a > 0$ and $c_{-1} > 0$, we can obviously obtain that $\frac{\mathrm{d}^2}{\mathrm{d}z^2}\hat{\ell}(z) \geq 0$, which implies that $\hat{\ell}(z)$ is also a convex function. □

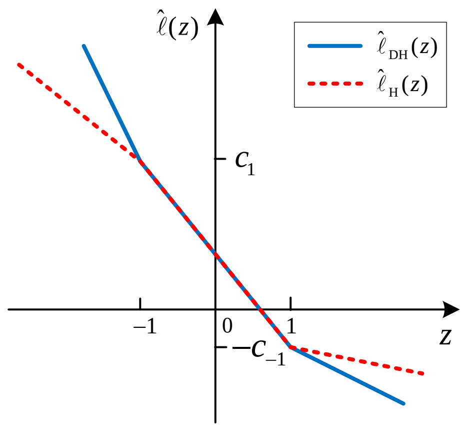


Figure 1: The weighted composite loss constructed by using the double hinge loss $\hat{\ell}_{\mathrm{DH}}(z)$ and the traditional hinge loss $\hat{\ell}_{\mathrm{H}}(z)$.

In addition, we illustrate the weighted composite loss $\hat{\ell}(z)$ constructed by employing the double hinge loss (denoted as $\hat{\ell}_{\mathrm{DH}}(z)$) and the hinge loss (denoted as $\hat{\ell}_{\mathrm{H}}(z)$) in Figure 1. From Figure 1, we can intuitively see that the weighted composite loss function regarding the double hinge loss is convex, while that regarding the traditional hinge loss is nonconvex. Therefore, it is reasonable to deploy the double hinge loss as our loss function.

Now, we are going to minimize the classification risk in Eq. (8). Empirically, given the training examples $\{\mathbf{x}_i^{\mathrm{p}}\}_{i=1}^{n_{\mathrm{p}}} \cup \{\mathbf{x}_i^{\mathrm{u}}\}_{i=1}^{n_{\mathrm{u}}}$, a decision function $f(\boldsymbol{X})$, and the double hinge loss $\ell_{\mathrm{DH}}(z) = \max(-z, \max(0, \frac{1}{2} - \frac{1}{2}z))$ [11], the above risk minimization process can be formulated as

$$\begin{aligned}\min_f \frac{\pi}{n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} (c_1 \max(-f(\mathbf{x}_i^{\mathrm{p}}), \max(0, \frac{1}{2} - \frac{1}{2} f(\mathbf{x}_i^{\mathrm{p}}))) \\ - c_{-1} \max(f(\mathbf{x}_i^{\mathrm{p}}), \max(0, \frac{1}{2} + \frac{1}{2} f(\mathbf{x}_i^{\mathrm{p}})))) \\ + \frac{c_{-1}}{n_{\mathrm{u}}} \sum_{i=1}^{n_{\mathrm{u}}} \max(f(\mathbf{x}_i^{\mathrm{u}}), \max(0, \frac{1}{2} + \frac{1}{2} f(\mathbf{x}_i^{\mathrm{u}}))). \qquad (11)\end{aligned}$$

To enable our algorithm to handle non-linear cases, we build our algorithm in the Reproducing Kernel Hilbert Space (RKHS). An RKHS $\mathcal{H}_K$ is a Hilbert space $\mathcal{H}$ of functions on a set $\mathcal{D}$ with the property that for all $x \in \mathcal{D}$ and $f \in \mathcal{H}$, the point evaluations $f \to f(x)$ are continuous linear functionals [44]. Consequently, according to the Moore-Aronszajn theorem [45], there exists a unique positive definite kernel $K(\cdot,\cdot)$ on $\mathcal{D} \times \mathcal{D}$ which has an important property that $\forall x_1, x_2 \in \mathcal{D}$, $K(x_1, x_2) = \langle K(\cdot, x_1), K(\cdot, x_2)\rangle_{\mathcal{H}}$. We further introduce the regularizer $\|f\|_{\mathcal{H}}^2$ to (11) to prevent overfitting, and then the optimization algorithm is expressed as

$$\begin{aligned}\min_{f \in \mathcal{H}_K} \frac{\pi}{n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} &(c_1 \max(-f(\mathbf{x}_i^{\mathrm{p}}), \max(0, \frac{1}{2} - \frac{1}{2} f(\mathbf{x}_i^{\mathrm{p}}))) \\ &- c_{-1} \max(f(\mathbf{x}_i^{\mathrm{p}}), \max(0, \frac{1}{2} + \frac{1}{2} f(\mathbf{x}_i^{\mathrm{p}})))) \\ &+ \frac{c_{-1}}{n_{\mathrm{u}}} \sum_{i=1}^{n_{\mathrm{u}}} \max(f(\mathbf{x}_i^{\mathrm{u}}), \max(0, \frac{1}{2} + \frac{1}{2} f(\mathbf{x}_i^{\mathrm{u}}))) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2, \end{aligned} \tag{12}$$

255 where λ is the non-negative trade-off parameter.

According to the representer theorem [46], the minimizer of (12) can be written as the expansion of kernel functions on all $n = n_{\mathrm{p}} + n_{\mathrm{u}}$ training examples, namely

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}, \mathbf{c}_i) + b, \tag{13}$$

where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^\top$ denotes the coefficient vector and b is the biased term. In this work, we employ the traditional Gaussian kernel, namely, $K(\mathbf{x}, \mathbf{c}_i) = \exp(-\|\mathbf{x} - \mathbf{c}_i\|^2/(2\sigma^2))$, where $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\} = \{\mathbf{x}_1^{\mathrm{p}}, \ldots, \mathbf{x}_{n_{\mathrm{p}}}^{\mathrm{p}}, \mathbf{x}_1^{\mathrm{u}}, \ldots, \mathbf{x}_{n_{\mathrm{u}}}^{\mathrm{u}}\}$ with $n = n_{\mathrm{p}} + n_{\mathrm{u}}$. Then we can get the kernel matrix $\mathbf{K}$ where the (i,j)-th element of $\mathbf{K}$ refers to $K(\mathbf{x}_i, \mathbf{c}_j)$. Alternatively, linear and polynomial functions can also be used as kernel functions, but this is not the focus of this paper, so here we only consider the Gaussian kernel. By incorporating b into $\boldsymbol{\alpha}$ as $\boldsymbol{\alpha} = (\boldsymbol{\alpha}^\top \ b)^\top$ and augmenting $\mathbf{K}$ as $\mathbf{K} = (\mathbf{K} \ \mathbf{1})$ with $\mathbf{1}$ being the all-one column vector, (13) can be written in a concise form as $f(\mathbf{x}) = \mathbf{K}\boldsymbol{\alpha}$. As a result, by introducing the slack variables $\boldsymbol{\eta}$ and $\boldsymbol{\xi}$, the primal problem equivalent to (12) is

$$\min_{\boldsymbol{\alpha},\boldsymbol{\eta},\boldsymbol{\xi}} \frac{\pi}{n_{\mathrm{p}}}\mathbf{1}^\top\boldsymbol{\eta} + \frac{c_{-1}}{n_{\mathrm{u}}}\mathbf{1}^\top\boldsymbol{\xi} + \frac{\lambda}{2}\boldsymbol{\alpha}^\top\mathbf{K}^\top\mathbf{K}\boldsymbol{\alpha} \tag{14}$$

$$\text{s.t.} \quad \boldsymbol{\eta} \geq -c_1\boldsymbol{\Phi}_{\mathrm{p}}\boldsymbol{\alpha}, \tag{15}$$

$$\boldsymbol{\eta} \geq -\frac{c_1+c_{-1}}{2}\boldsymbol{\Phi}_{\mathrm{p}}\boldsymbol{\alpha} + \frac{c_1-c_{-1}}{2}\mathbf{1}, \tag{16}$$

$$\boldsymbol{\eta} \geq -c_{-1}\boldsymbol{\Phi}_{\mathrm{p}}\boldsymbol{\alpha}, \tag{17}$$

$$\boldsymbol{\xi} \geq \mathbf{0}, \tag{18}$$

$$\boldsymbol{\xi} \geq \frac{1}{2}\boldsymbol{\Phi}_{\mathrm{u}}\boldsymbol{\alpha} + \frac{1}{2}\mathbf{1}, \tag{19}$$

$$\boldsymbol{\xi} \geq \boldsymbol{\Phi}_{\mathrm{u}}\boldsymbol{\alpha}, \tag{20}$$

where $\boldsymbol{\Phi}_{\mathrm{p}}$ is the first n_{p} rows of $\mathbf{K}$, $\boldsymbol{\Phi}_{\mathrm{u}}$ is the remaining n_{u} rows of $\mathbf{K}$, and "$\geq$" is applied element-wise on vectors.

4. Optimization

In this section, we solve our algorithm presented in (14)-(20), which falls into scope of Quadratic Programming (QP) that has the form

$$\begin{aligned} \min_{\boldsymbol{\gamma}} \ & \frac{1}{2}\boldsymbol{\gamma}^\top\mathbf{H}\boldsymbol{\gamma} + \mathbf{f}^\top\boldsymbol{\gamma} \\ \text{s.t.} \ & \mathbf{L}\boldsymbol{\gamma} \leq \mathbf{k}, \\ & \mathbf{q} \leq \boldsymbol{\gamma}. \end{aligned} \tag{21}$$

In our algorithm, we let

$$\boldsymbol{\gamma} = \begin{bmatrix}\boldsymbol{\alpha}^\top_{(n+1)\times 1} & \boldsymbol{\eta}^\top_{n_{\mathrm{p}}\times 1} & \boldsymbol{\xi}^\top_{n_{\mathrm{u}}\times 1}\end{bmatrix}^\top. \tag{22}$$

Then $\mathbf{H}$ is defined as

$$\mathbf{H} = \begin{bmatrix} \lambda\mathbf{K}^\top\mathbf{K} & \mathbf{O}_{(n+1)\times n_{\mathrm{p}}} & \mathbf{O}_{(n+1)\times n_{\mathrm{u}}} \\ \mathbf{O}_{n_{\mathrm{p}}\times(n+1)} & \mathbf{O}_{n_{\mathrm{p}}\times n_{\mathrm{p}}} & \mathbf{O}_{n_{\mathrm{p}}\times n_{\mathrm{u}}} \\ \mathbf{O}_{n_{\mathrm{u}}\times(n+1)} & \mathbf{O}_{n_{\mathrm{u}}\times n_{\mathrm{p}}} & \mathbf{O}_{n_{\mathrm{u}}\times n_{\mathrm{u}}} \end{bmatrix}, \tag{23}$$

where the $\mathbf{O}_{(n+1)\times n_{\mathrm{p}}}$ is a zero matrix of the size $(n+1)\times n_{\mathrm{p}}$. Accordingly, the coefficient $\mathbf{f}$ in (14) is constituted of

$$\mathbf{f} = \begin{bmatrix}\mathbf{0}^\top_{(n+1)\times 1} & \frac{\pi}{n_{\mathrm{p}}}\mathbf{1}^\top_{n_{\mathrm{p}}\times 1} & \frac{c_{-1}}{n_{\mathrm{u}}}\mathbf{1}^\top_{n_{\mathrm{u}}\times 1}\end{bmatrix}^\top. \tag{24}$$

Similarly, the $\mathbf{q}$ in the constraint of (14) is

$$\mathbf{q} = \begin{bmatrix} -\boldsymbol{\infty}_{(n+1)\times 1}^{\top} & -\boldsymbol{\infty}_{n_{\mathrm{p}}\times 1}^{\top} & \mathbf{0}_{n_{\mathrm{u}}\times 1}^{\top} \end{bmatrix}^{\top}. \tag{25}$$

As a consequence, the five linear constraints in (15)-(17) and (19)-(20) are transformed as

$$\begin{aligned} -c_1\boldsymbol{\Phi}_{\mathrm{p}}\boldsymbol{\alpha} - \boldsymbol{\eta} &\leq \mathbf{0}_{n_{\mathrm{p}}\times 1}, \\ -\frac{c_1 + c_{-1}}{2}\boldsymbol{\Phi}_{\mathrm{p}}\boldsymbol{\alpha} - \boldsymbol{\eta} &\leq -\frac{c_1 - c_{-1}}{2}\mathbf{1}_{n_{\mathrm{p}}\times 1}, \\ -c_{-1}\boldsymbol{\Phi}_{\mathrm{p}}\boldsymbol{\alpha} - \boldsymbol{\eta} &\leq \mathbf{0}_{n_{\mathrm{p}}\times 1}, \\ \frac{1}{2}\boldsymbol{\Phi}_{\mathrm{u}}\boldsymbol{\alpha} - \boldsymbol{\xi} &\leq -\frac{1}{2}\mathbf{1}_{n_{\mathrm{u}}\times 1}, \\ \boldsymbol{\Phi}_{\mathrm{u}}\boldsymbol{\alpha} - \boldsymbol{\xi} &\leq \mathbf{0}_{n_{\mathrm{u}}\times 1}. \end{aligned}$$

Then we can get

$$\mathbf{L} = \begin{bmatrix} -c_1\boldsymbol{\Phi}_{\mathrm{p}} & -\mathbf{I}_{n_{\mathrm{p}}\times n_{\mathrm{p}}} & \mathbf{O}_{n_{\mathrm{p}}\times n_{\mathrm{u}}} \\ -\frac{c_1+c_{-1}}{2}\boldsymbol{\Phi}_{\mathrm{p}} & -\mathbf{I}_{n_{\mathrm{p}}\times n_{\mathrm{p}}} & \mathbf{O}_{n_{\mathrm{p}}\times n_{\mathrm{u}}} \\ -c_{-1}\boldsymbol{\Phi}_{\mathrm{p}} & -\mathbf{I}_{n_{\mathrm{p}}\times n_{\mathrm{p}}} & \mathbf{O}_{n_{\mathrm{p}}\times n_{\mathrm{u}}} \\ \frac{1}{2}\boldsymbol{\Phi}_{\mathrm{u}} & \mathbf{O}_{n_{\mathrm{u}}\times n_{\mathrm{p}}} & -\mathbf{I}_{n_{\mathrm{u}}\times n_{\mathrm{u}}} \\ \boldsymbol{\Phi}_{\mathrm{u}} & \mathbf{O}_{n_{\mathrm{u}}\times n_{\mathrm{p}}} & -\mathbf{I}_{n_{\mathrm{u}}\times n_{\mathrm{u}}} \end{bmatrix} \tag{26}$$

and

$$\mathbf{k} = \begin{bmatrix} \mathbf{0}_{n_{\mathrm{p}}\times 1}^{\top} & -\frac{c_1-c_{-1}}{2}\mathbf{1}_{n_{\mathrm{p}}\times 1}^{\top} & \mathbf{0}_{n_{\mathrm{p}}\times 1}^{\top} & -\frac{1}{2}\mathbf{1}_{n_{\mathrm{u}}\times 1}^{\top}, & \mathbf{0}_{n_{\mathrm{u}}\times 1}^{\top} \end{bmatrix}^{\top}. \tag{27}$$

Since we have the specific formations of $\mathbf{H}$, $\mathbf{f}$, $\mathbf{q}$, $\mathbf{L}$, and $\mathbf{k}$ in our case, we can use any off-the-shelf QP optimization toolbox [47, 48] to solve the optimization problem (14)-(20), such as the "quadprog" command in Matlab [47]. The detailed procedure of above optimization process is presented in Algorithm 1.

5. Theoretical Analyses

This section provides the theoretical analyses on CSPU. We firstly analyze the computational complexity of Algorithm 1, and then theoretically derive a generalization error bound of CSPU.

Algorithm 1 **C**ost-**S**ensitive **P**ositive and **U**nlabeled learning (CSPU) algorithm

```
Input: positive set $\mathcal{X}_{\rm p} = \{\mathbf{x}_i^{\rm p}\}_{i=1}^{n_{\rm p}}$, unlabeled set $\mathcal{X}_{\rm u} = \{\mathbf{x}_i^{\rm u}\}_{i=1}^{n_{\rm u}}$; trade-off parameters $c_1$, $c_{-1}$, and $\lambda$;
Output: the optimal classifier parameter $\boldsymbol{\alpha}$;
1: Compute kernel matrix $\mathbf{K}$;
2: Compute matrix $\mathbf{H}$ via (23);
3: Compute vector $\mathbf{f}$ via (24);
4: Compute vector $\mathbf{q}$ via (25);
5: Compute matrix $\mathbf{L}$ via (26);
6: Compute vector $\mathbf{k}$ via (27);
7: Find the solution $\boldsymbol{\gamma}$ of problem (21) via existing QP solver;
return $\boldsymbol{\alpha}$, namely the first $n+1$ elements of $\boldsymbol{\gamma}$ according to (22).
```

5.1. Computational Complexity Analysis

In this section, we analyze the computational complexity of Algorithm 1. The kernel matrix computation in Line 1 takes the complexity of $\mathcal{O}(n^2)$. In Line 2, it involves the computation of $\mathbf{K}^\top\mathbf{K}$ where $\mathbf{K} = (\mathbf{K}\ \mathbf{1})$ is an $n \times (n+1)$ augmented kernel matrix, so the complexity is $\mathcal{O}((n+1)^2 n)$. Besides, note that Line 7 is accomplished by using the "quadprog command in Matlab, which utilizes the interior point method and the complexity is $\mathcal{O}((2n+1)^3)$ [49]. Therefore, the total complexity of our proposed CSPU algorithm is $\mathcal{O}(9n^3 + 15n^2 + 7n + 1)$.

5.2. Generalizability Analysis

In this section, we study the generalizability of the proposed learning algorithm. We show that the empirical classification risk of any classifier learned by the proposed algorithm converges to its expected classification risk when the amounts of both positive and unlabeled data are sufficiently large. Since we utilize the kernel trick in our algorithm, the generalizability analysis presented below are discussed in the Hilbert space.

Specifically, let $\phi : \mathcal{X} \to \mathcal{H}$ be a mapping of input features to a Hilbert space $\mathcal{H}$ with inner product $\langle \cdot, \cdot \rangle$. A vector $\boldsymbol{\alpha}_{\mathcal{H}} \in \mathcal{H}$ can be used to predict the label of $\mathbf{x}$ as

$\hat{y} = \mathrm{sign}(\langle \boldsymbol{\alpha}_{\mathcal{H}}, \phi(\mathbf{x}) \rangle)$. First of all, we define the expected classification error of a classifier $\boldsymbol{\alpha}_{\mathcal{H}}$ by

$$R(\boldsymbol{\alpha}_{\mathcal{H}}) = \mathbb{E}[\mathbb{1}_{Y \boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\boldsymbol{X}) \leq 0}], \tag{28}$$

295 where $\mathbb{1}_{\{\cdot\}}$ is the indicator function for representing the zero-one loss.

Since there are only positive and unlabeled data for training, we need to rewrite the expected risk. We set

$$R_{\mathrm{p}}(\boldsymbol{\alpha}_{\mathcal{H}}) = \int p(\phi(\boldsymbol{X})|Y = +1) \mathbb{1}_{\boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\boldsymbol{X}) \leq 0} d\phi(\boldsymbol{X}) \tag{29}$$

and

$$R_{\mathrm{n}}(\boldsymbol{\alpha}_{\mathcal{H}}) = \int p(\phi(\boldsymbol{X})|Y = -1) \mathbb{1}_{\boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\boldsymbol{X}) \geq 0} d\phi(\boldsymbol{X}), \tag{30}$$

which denote the false negative risk and false positive risk, respectively, and then we 300 have

$$R(\boldsymbol{\alpha}_{\mathcal{H}}) = \mathbb{E}[\mathbb{1}_{Y \boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\boldsymbol{X}) \leq 0}] = \pi R_{\mathrm{p}}(\boldsymbol{\alpha}_{\mathcal{H}}) + (1 - \pi) R_{\mathrm{n}}(\boldsymbol{\alpha}_{\mathcal{H}}). \tag{31}$$

Moreover, we define the risk on unlabeled set as

$$\begin{aligned} R_{\mathrm{u}}(\boldsymbol{\alpha}_{\mathcal{H}}) = \mathbb{E}[\mathbb{1}_{\boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\boldsymbol{X}) \geq 0}] &= \int p(\phi(\boldsymbol{X})) \mathbb{1}_{\boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\boldsymbol{X}) \geq 0} d\phi(\boldsymbol{X}) \\ &= \int (p(\phi(\boldsymbol{X})|Y = +1) p(Y = +1) \\ &\qquad + p(\phi(\boldsymbol{X})|Y = -1) p(Y = -1)) \mathbb{1}_{\boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\boldsymbol{X}) \geq 0} d\phi(\boldsymbol{X}) \\ &= \pi(1 - R_{\mathrm{p}}(\boldsymbol{\alpha}_{\mathcal{H}})) + (1 - \pi) R_{\mathrm{n}}(\boldsymbol{\alpha}_{\mathcal{H}}). \end{aligned} \tag{32}$$

By combining Eqs. (31) and (32), we can obtain the excepted risk on PU datasets, namely

$$R(\boldsymbol{\alpha}_{\mathcal{H}}) = 2\pi R_{\mathrm{p}}(\boldsymbol{\alpha}_{\mathcal{H}}) + R_{\mathrm{u}}(\boldsymbol{\alpha}_{\mathcal{H}}) - \pi. \tag{33}$$

Accordingly, the empirical margin error of the classifier $\boldsymbol{\alpha}_{\mathcal{H}}$ can be defined as

$$\hat{R}_{\rho}(\boldsymbol{\alpha}_{\mathcal{H}}) = \frac{2\pi}{n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} \mathbb{1}_{\boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\mathbf{x}_i^{\mathrm{p}}) \leq \rho} + \frac{1}{n_{\mathrm{u}}} \sum_{i=1}^{n_{\mathrm{u}}} \mathbb{1}_{-\boldsymbol{\alpha}_{\mathcal{H}}^{\top} \phi(\mathbf{x}_i^{\mathrm{u}}) \leq \rho} - \pi, \tag{34}$$

where ρ is the margin. Note that when $\rho \to 0$, we have $\mathbb{E}[\hat{R}_{\rho}(\boldsymbol{\alpha}_{\mathcal{H}})] \to R(\boldsymbol{\alpha}_{\mathcal{H}})$.

305 Let $\hat{\boldsymbol{\alpha}}_{\mathcal{H}}$ be any learned classifier output by the proposed learning algorithm. Our target is to upper bound the generalization error $R(\hat{\boldsymbol{\alpha}}_{\mathcal{H}}) - \hat{R}_{\rho}(\hat{\boldsymbol{\alpha}}_{\mathcal{H}})$.

Theorem 2. *Assume that* $\forall \mathbf{x} \in \mathcal{X}$*,* $\|\phi(\mathbf{x})\| \leq B$*. Let* $\hat{\boldsymbol{\alpha}}_{\mathcal{H}}$ *be the classifier parameter learned by the proposed algorithm. For any* $\delta > 0$*, with probability at least* $1-\delta$*, we have*

$$R(\hat{\boldsymbol{\alpha}}_{\mathcal{H}}) - \hat{R}_{\rho}(\hat{\boldsymbol{\alpha}}_{\mathcal{H}}) \leq \frac{2B\sqrt{\pi c_1 + (1-\pi)c_{-1}}}{\rho\sqrt{\lambda n_{\mathrm{p}}}} + \frac{2B\sqrt{\pi c_1 + (1-\pi)c_{-1}}}{\rho\sqrt{\lambda n_{\mathrm{u}}}} + \sqrt{\frac{\ln(1/\delta)}{2n}}. \tag{35}$$

Before proving this theorem, we first present some useful definitions and lemmas.

Definition 3. ***(Rademacher Complexity, [50])*** *Let* $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_n\}$ *be a set of independent Rademacher variables which are uniformly sampled from* $\{-1, 1\}$*. Let* $v_1, \ldots, v_n$ *be an independent distributed sample set and* $\mathcal{F}$ *be a function class, then the Rademacher complexity is defined as*

$$\mathfrak{R}_n(\mathcal{F}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(v_i)\right]. \tag{36}$$

The Rademacher complexity is often used to derive the dimensionality independent generalization error bound of a decision function f, which is

Lemma 4. ***(Generalization Bound, [50])*** *Let* $\mathcal{F}$ *be a* $[0,1]$*-valued function class on* $\mathcal{X}$ *and* $f \in \mathcal{F}$*. Given* $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ *as i.i.d. variables, then for any* $\delta > 0$*, with probability at least* $1-\delta$*, we have*

$$\sup_{f\in\mathcal{F}}\left(\mathbb{E}f\left(\boldsymbol{X}\right) - \frac{1}{n}\sum_{i=1}^{n} f\left(\mathbf{x}_i\right)\right) \leq 2\mathfrak{R}_n(\mathcal{F}) + \sqrt{\frac{\ln(1/\delta)}{2n}}. \tag{37}$$

In addition, the Rademacher complexity appeared in (37) can be usually upper bounded by the following lemma, which is

Lemma 5. ***(Talagrand Contraction Lemma, [50])*** *If* $\ell : \mathbb{R} \to \mathbb{R}$ *is Lipschitz continuous with constant L and satisfies* $\ell(0) = 0$*, then*

$$\mathfrak{R}_n(\ell \circ \mathcal{F}) \leq L\mathfrak{R}_n(\mathcal{F}), \tag{38}$$

where "$\circ$" represents the composition of two functions.

Now, we present the formal proof for Theorem 2.

Proof. Firstly, we introduce the following function

$$\ell_\rho(z) = \begin{cases} 0 & \text{if } z > \rho, \\ 1 - z/\rho & \text{if } 0 \leq z \leq \rho, \\ 1 & \text{if } z < 0, \end{cases} \tag{39}$$

where ρ is the margin as defined in (34). Let

$$R(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) = \mathbb{E}[\ell_\rho(Y \boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\boldsymbol{X}))] = 2\pi R_{\mathrm{p}}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) + R_{\mathrm{u}}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) - \pi, \tag{40}$$

$$\hat{R}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) = \frac{2\pi}{n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} \ell_\rho(\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{p}})) + \frac{1}{n_{\mathrm{u}}} \sum_{i=1}^{n_{\mathrm{u}}} \ell_\rho(-\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{u}})) - \pi, \tag{41}$$

where $\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}$ represents the composite function, and

$$R_{\mathrm{p}}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) = \int p(\phi(\boldsymbol{X})|Y = +1) \ell_\rho(\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\boldsymbol{X})) d\phi(\boldsymbol{X}) \tag{42}$$

330 and

$$R_{\mathrm{u}}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) = \int p(\phi(\boldsymbol{X})|Y = -1) \ell_\rho(-\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\boldsymbol{X})) d\phi(\boldsymbol{X}). \tag{43}$$

It can be easily verified that $R(\boldsymbol{\alpha}_{\mathcal{H}}) \leq R(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}})$ and $\hat{R}_\rho(\boldsymbol{\alpha}_{\mathcal{H}}) \geq \hat{R}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}})$, which imply that

$$R(\boldsymbol{\alpha}_{\mathcal{H}}) - \hat{R}_\rho(\boldsymbol{\alpha}_{\mathcal{H}}) \leq R(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) - \hat{R}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}). \tag{44}$$

Let $\mathcal{A}$ be the set of all possible learned classifiers, then we have

$$R(\hat{\boldsymbol{\alpha}}_{\mathcal{H}}) - \hat{R}_\rho(\hat{\boldsymbol{\alpha}}_{\mathcal{H}}) \leq \sup_{\boldsymbol{\alpha}_{\mathcal{H}} \in \mathcal{A}} (R(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) - \hat{R}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}})). \tag{45}$$

We are now going to upper bound the defect $R(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) - \hat{R}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}})$. According 335 to Lemma 4, with probability at least $1 - \delta$, we have

$$\sup_{\boldsymbol{\alpha}_{\mathcal{H}} \in \mathcal{A}} (R(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}}) - \hat{R}(\ell_\rho \circ \boldsymbol{\alpha}_{\mathcal{H}})) \leq 2\mathfrak{R}_{\mathrm{p}}(\ell_\rho \circ \mathcal{A}) + 2\mathfrak{R}_{\mathrm{u}}(\ell_\rho \circ \mathcal{A}) + \sqrt{\frac{\ln(1/\delta)}{2n}}, \tag{46}$$

where

$$\mathfrak{R}_{\mathrm{p}}(\ell_\rho \circ \mathcal{A}) = \mathbb{E}\left[\sup_{\boldsymbol{\alpha}_{\mathcal{H}} \in \mathcal{A}} \frac{1}{n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} \sigma_i \ell_\rho(\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{p}}))\right] \tag{47}$$

and

$$\mathfrak{R}_{\mathrm{u}}(\ell_\rho \circ \mathcal{A}) = \mathbb{E}\left[\sup_{\boldsymbol{\alpha}_{\mathcal{H}} \in \mathcal{A}} \frac{1}{n_{\mathrm{u}}} \sum_{i=1}^{n_{\mathrm{u}}} \sigma_i \ell_\rho(\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{u}}))\right]. \tag{48}$$

To obtain the upper bounds of the Rademacher complexities in (47) and (48), we first derive an upper bound for the classifiers in $\mathcal{A}$. Specifically, due to the optimality of any $\boldsymbol{\alpha}_{\mathcal{H}} \in \mathcal{A}$, we have

$$\begin{aligned}
&\frac{\lambda}{2}\|\boldsymbol{\alpha}_{\mathcal{H}}\|^2 + \frac{\pi}{n_{\mathrm{p}}}\sum_{i=1}^{n_{\mathrm{p}}}(c_1 \max(-\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{p}}), \max(0, \frac{1}{2} - \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{p}})))\\
&\qquad\qquad - c_{-1} \max(\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{p}}), \max(0, \frac{1}{2} + \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{p}}))))\\
&\quad + \frac{c_{-1}}{n_{\mathrm{u}}}\sum_{i=1}^{n_{\mathrm{u}}} \max(\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{u}}), \max(0, \frac{1}{2} + \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{u}})))\\
&\le \frac{\lambda}{2}\|\mathbf{0}\|^2 + \frac{\pi}{n_{\mathrm{p}}}\sum_{i=1}^{n_{\mathrm{p}}}(c_1 \max(-\mathbf{0}^\top \phi(\mathbf{x}_i^{\mathrm{p}}), \max(0, \frac{1}{2} - \frac{1}{2}\mathbf{0}^\top \phi(\mathbf{x}_i^{\mathrm{p}})))\\
&\qquad\qquad - c_{-1} \max(\mathbf{0}^\top \phi(\mathbf{x}_i^{\mathrm{p}}), \max(0, \frac{1}{2} + \frac{1}{2}\mathbf{0}^\top \phi(\mathbf{x}_i^{\mathrm{p}}))))\\
&\quad + \frac{c_{-1}}{n_{\mathrm{u}}}\sum_{i=1}^{n_{\mathrm{u}}} \max(\mathbf{0}^\top \phi(\mathbf{x}_i^{\mathrm{u}}), \max(0, \frac{1}{2} + \frac{1}{2}\mathbf{0}^\top \phi(\mathbf{x}_i^{\mathrm{u}})))\\
&= \frac{\pi c_1 + (1-\pi)c_{-1}}{2}. \qquad (49)
\end{aligned}$$

This implies that $\|\boldsymbol{\alpha}_{\mathcal{H}}\|^2 \le \frac{\pi c_1 + (1-\pi)c_{-1}}{\lambda}$.

Now, we are going to upper bound $\mathfrak{R}_{\mathrm{p}}(\ell_\rho \circ \mathcal{A})$ and $\mathfrak{R}_{\mathrm{u}}(\ell_\rho \circ \mathcal{A})$. Specially, since the function $\ell_\rho(z)$ is $1/\rho$-Lipschtiz, by using Lemma 5, we have

$$\begin{aligned}
\mathfrak{R}_{\mathrm{p}}(\ell_\rho \circ \mathcal{A}) &\leq \frac{1}{\rho}\mathfrak{R}_{\mathrm{p}}(\mathcal{A}) \\
&= \frac{1}{\rho}\mathbb{E}\left[\sup_{\boldsymbol{\alpha}_{\mathcal{H}}\in\mathcal{A}} \frac{1}{n_{\mathrm{p}}}\sum_{i=1}^{n_{\mathrm{p}}} \sigma_i \boldsymbol{\alpha}_{\mathcal{H}}^\top \phi(\mathbf{x}_i^{\mathrm{p}})\right] \\
&= \frac{1}{\rho}\mathbb{E}\left[\sup_{\boldsymbol{\alpha}_{\mathcal{H}}\in\mathcal{A}} \left\langle \boldsymbol{\alpha}_{\mathcal{H}}, \frac{1}{n_{\mathrm{p}}}\sum_{i=1}^{n_{\mathrm{p}}} \sigma_i \phi(\mathbf{x}_i^{\mathrm{p}})\right\rangle\right] \\
&\overset{1}{\leq} \frac{1}{\rho n_{\mathrm{p}}}\mathbb{E}\left[\sup_{\boldsymbol{\alpha}\in\mathcal{A}} \|\boldsymbol{\alpha}_{\mathcal{H}}\| \left\|\sum_{i=1}^{n_{\mathrm{p}}} \sigma_i \phi(\mathbf{x}_i^{\mathrm{p}})\right\|\right] \\
&\leq \frac{\sqrt{\pi c_1 + (1-\pi)c_{-1}}}{\sqrt{\lambda}\rho n_{\mathrm{p}}}\mathbb{E}\left[\left\|\sum_{i=1}^{n_{\mathrm{p}}} \sigma_i \phi(\mathbf{x}_i^{\mathrm{p}})\right\|\right] \\
&\overset{2}{\leq} \frac{\sqrt{\pi c_1 + (1-\pi)c_{-1}}}{\sqrt{\lambda}\rho n_{\mathrm{p}}}\sqrt{\sum_{i=1}^{n_{\mathrm{p}}} \mathbb{E}[\|\phi(\mathbf{x}_i^{\mathrm{p}})\|^2]} \\
&\leq \frac{B\sqrt{\pi c_1 + (1-\pi)c_{-1}}}{\rho\sqrt{\lambda n_{\mathrm{p}}}},
\end{aligned} \tag{50}$$

where the Inequality 1 holds because of the CauchySchwarz inequality, and the In-
340 equality 2 holds due to the Jensens inequality.

Similarly, we have

$$\mathfrak{R}_{\mathrm{u}}(\ell_\rho \circ \mathcal{A}) \leq \frac{B\sqrt{\pi c_1 + (1-\pi)c_{-1}}}{\rho\sqrt{\lambda n_{\mathrm{u}}}}. \tag{51}$$

By combining inequalities (45), (46), (50), and (51), we know that for any learned $\hat{\boldsymbol{\alpha}}_{\mathcal{H}}$, with probability at least $1-\rho$, we have

$$R(\hat{\boldsymbol{\alpha}}_{\mathcal{H}}) - \hat{R}_\rho(\hat{\boldsymbol{\alpha}}_{\mathcal{H}}) \leq \frac{2B\sqrt{\pi c_1 + (1-\pi)c_{-1}}}{\rho\sqrt{\lambda n_{\mathrm{p}}}} + \frac{2B\sqrt{\pi c_1 + (1-\pi)c_{-1}}}{\rho\sqrt{\lambda n_{\mathrm{u}}}} + \sqrt{\frac{\ln(1/\delta)}{2n}}, \tag{52}$$

which concludes the proof of Theorem 2. □

345 Theorem 2 not only shows that the generalization error will converge to the expected classification risk when the amounts of both positive and unlabeled data get large, but also reveals that the convergence rate is at the order of $\mathcal{O}(1/\sqrt{n_{\mathrm{p}}}+1/\sqrt{n_{\mathrm{u}}}+1/\sqrt{n})$. Besides, we see that in the cost-sensitive setting, the value of π is small and $1-\pi$ is large. Then in order to make the bound at the right-hand side of inequality (52) small,

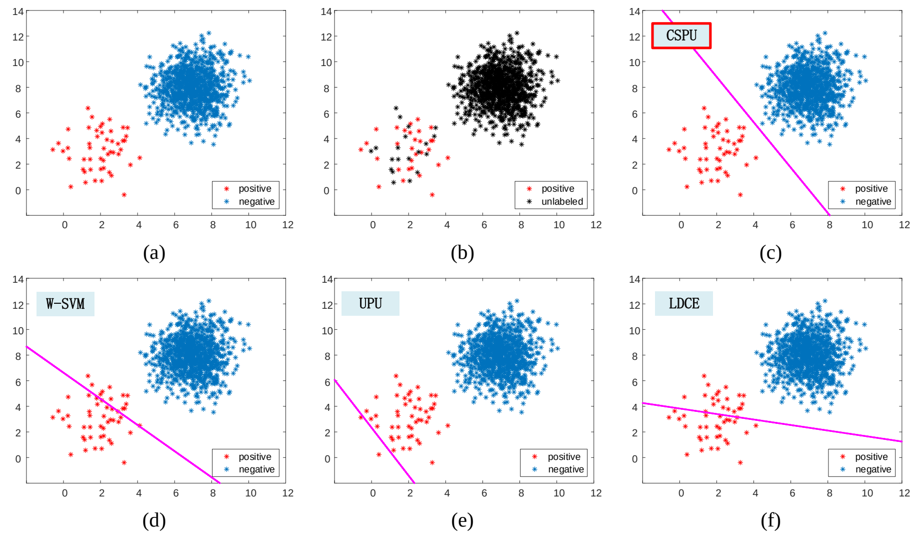


Figure 2: The performances of various PU methods on synthetic dataset. (a) The real positive and negative examples (50 vs. 1000); (b) The positive and unlabeled data for training; (c)-(f) The decision boundaries generated by CSPU, W-SVM, UPU, and LDCE, respectively.

the value of c_{-1} (i.e., the weight on the false positive error) should be tuned to a small value, which is also consistent with our general understanding.

6. Experiments

In this section, we test the performance of our proposed CSPU by performing exhaustive experiments on one synthetic dataset, four publicly available benchmark datasets, and two real-world datasets. To demonstrate the superiority of CSPU, we compare it with several state-of-the-art PU learning algorithms including Weighted SVM (W-SVM) [19], Unbiased PU learning (UPU) [11], Multi-Layer Perceptron with Non-Negative PU risk estimator (NNPU-MLP) [12], Linear classifier with Non-Negative PU risk estimator (NNPU-Linear) [12], and Loss Decomposition and Centroid Estimation (LDCE) [7].

Table 1: Overview of the adopted datasets. $n = n'_{\mathrm{p}} + n'_{\mathrm{n}}$ is the total number of examples, where n'_{p} and n'_{n} are the real number of positive and negative examples, respectively. d denotes the feature dimensionality, and p_+ refers to the proportion of positive examples.

Datasets	(n, d)	n'_{p}	n'_{n}	p_+
German	(1000,24)	300	700	0.30
OVA_Breast	(1545,10936)	344	1201	0.22
Wilt	(4839,5)	261	4578	0.05
Satellite	(5100,36)	75	5025	0.01

6.1. Synthetic Dataset

First of all, we create a two-dimensional toy dataset composed of two Gaussian clusters as shown in Figure 2(a). This dataset contains 50 positive examples and 1000 negative examples, and each class corresponds to a Gaussian. The proportion of positive examples in the whole dataset is approximately 0.05, resulting in a highly imbalanced case. Then, we randomly pick 40% of the original positive examples along with all negative examples to be unlabeled (see Figure 2(b)), and examine whether different PU learning methods can accurately find the proper decision boundary for separating positive and negative examples. The results of these methods are shown in Figure 2(c)-(f), revealing that only the decision boundary generated by our CSPU can separate positive and negative examples perfectly. Note that NNPU-MLP and NNPU-Linear are not compared here as they do not generate explicit decision boundaries. Specially, we notice that W-SVM, UPU, and LDCE misclassify positive examples (the minority) as negative ones (the majority) to some degree, which means that they cannot address the problem caused by the dominance of negative class. This also demonstrates that our CSPU has the capability to calibrate the improper decision boundary to the potentially correct one in dealing with class imbalanced situation.

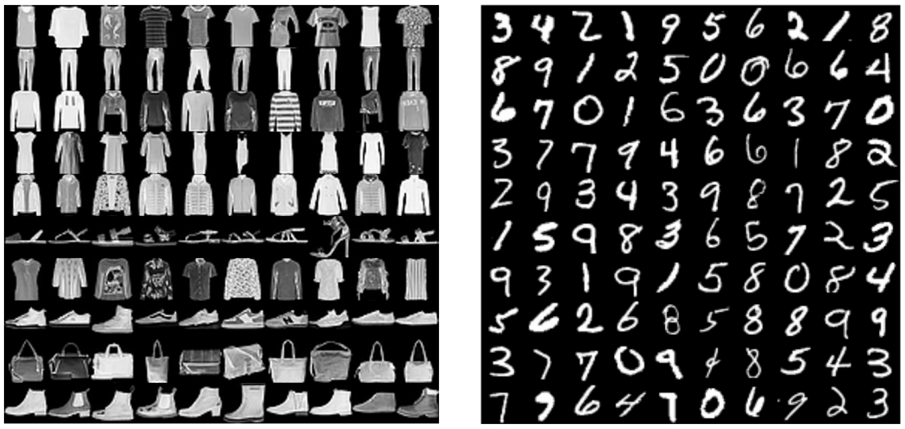
6.2. Benchmark Datasets

In this section, we conduct extensive experiments on four imbalanced OpenML benchmark datasets including *German*, *OVA_Breast*, *Wilt*, and *Satellite*. Their configurations are summarized in Table 1. By comparing n'_{p} and n'_{n}, we see that these datasets are highly imbalanced. For each dataset, we randomly pick up 80% of the whole data for training and leave the rest 20% examples for testing. Then, we randomly select $r = 20\%$, 30%, 40% positive training examples and combine them with the original negative ones to compose the unlabeled set. Note that the division of positive set and unlabeled set under each r is kept identical for all compared methods, and all data features have been normalized in advance. In our experiments, to tune the model parameters for every compared method, we conduct 5-fold cross validation within the training set. Specifically, the regularization parameter λ for CSPU is tuned by searching the grid $\{10^{-4}, \ldots, 10^{1}\}$, and c_{-1} is set to 1 while c_1 is tuned around $\frac{n'_{\mathrm{n}}}{n'_{\mathrm{p}}}$. For W-SVM, we tune the parameter γ from $\{0.1, 0.2, \ldots, 0.9\}$. In UPU, the regularization parameter λ is selected from $\{10^{-3}, \ldots, 10^{1}\}$. For NNPU, the parameter β is tuned to the default value 0 on all benchmark datasets. As for LDCE, we choose regularization parameter λ from $\{2^{-4}, \ldots, 2^{4}\}$ and β from $\{0.1, 0.2, \ldots, 0.9\}$, respectively. To evaluate the performances of various methods, we report the mean F-measures on test set over five independent trials, and the results are shown in Table 2. Furthermore, we also apply the *t*-test with significant level 0.05 to investigate whether our CSPU is significantly better than other methods. Note that LDCE is not compared on *OVA_Breast* dataset as it is not scalable to this large dataset.

From the mean test F-measures reported in Table 2, we can see that our CSPU generally achieves the highest record in terms of F-measure when compared with other methods. Specifically, it is easy to observe that CSPU significantly outperforms UPU and LDCE in all cases, and UPU performs unsatisfactorily on *Wilt* dataset as it mistakenly classifies all data into negative. Moreover, out of the 57 cases (5 compared methods, 4 datasets, and 3 different settings of r, and subtracting 3 cases of LDCE on the *OVA_Breast* dataset), CSPU significantly outperforms all the compared methods in 91.2% cases, and achieves competitive performance in 8.8% cases. In summary, these experimental results on the four imbalanced benchmark datasets demonstrate the effectiveness of our proposed CSPU in handling the data imbalanced problem.

Table 2: The F-measures (mean±std) of various PU methods on adopted datasets when $r = 20\%$, 30%, and 40%. The best record under each r is marked in **bold**. "$\surd$" indicates that CSPU is significantly better than the corresponding methods via paired t-test with significance level 0.05.

Datasets	Methods	$r = 20\%$	$r = 30\%$	$r = 40\%$
German	W-SVM [19]	0.481 ± 0.042 $\surd$	0.462 ± 0.001 $\surd$	0.463 ± 0.002 $\surd$
	UPU [11]	0.007 ± 0.015 $\surd$	0.007 ± 0.015 $\surd$	0.018 ± 0.041 $\surd$
	NNPU-MLP [12]	0.379 ± 0.027 $\surd$	0.427 ± 0.066 $\surd$	0.428 ± 0.103 $\surd$
	NNPU-Linear [12]	0.461 ± 0.049 $\surd$	0.400 ± 0.032 $\surd$	0.388 ± 0.069 $\surd$
	LDCE [7]	0.460 ± 0.006 $\surd$	0.456 ± 0.000 $\surd$	0.462 ± 0.001 $\surd$
	CSPU (ours)	**0.634 ± 0.029**	**0.633 ± 0.027**	**0.630 ± 0.031**
OVA_Breast	W-SVM [19]	0.821 ± 0.019 $\surd$	0.725 ± 0.041 $\surd$	0.640 ± 0.083 $\surd$
	UPU [11]	0.892 ± 0.029 $\surd$	0.912 ± 0.028 $\surd$	0.904 ± 0.013 $\surd$
	NNPU-MLP [12]	0.775 ± 0.088 $\surd$	0.807 ± 0.103 $\surd$	0.825 ± 0.068 $\surd$
	NNPU-Linear [12]	0.924 ± 0.008	0.918 ± 0.015 $\surd$	0.918 ± 0.013 $\surd$
	LDCE [7]	-	-	-
	CSPU (ours)	**0.934 ± 0.012**	**0.934 ± 0.017**	**0.933 ± 0.015**
Wilt	W-SVM [19]	0.717 ± 0.036 $\surd$	0.701 ± 0.062	0.674 ± 0.041 $\surd$
	UPU [11]	0.000 ± 0.000 $\surd$	0.000 ± 0.000 $\surd$	0.000 ± 0.000 $\surd$
	NNPU-MLP [12]	0.741 ± 0.042	0.722 ± 0.038	0.720 ± 0.033
	NNPU-Linear [12]	0.125 ± 0.050 $\surd$	0.187 ± 0.099 $\surd$	0.142 ± 0.035 $\surd$
	LDCE [7]	0.108 ± 0.004 $\surd$	0.107 ± 0.002 $\surd$	0.106 ± 0.002 $\surd$
	CSPU (ours)	**0.757 ± 0.016**	**0.730 ± 0.045**	**0.726 ± 0.014**
Satellite	W-SVM [19]	0.633 ± 0.080 $\surd$	0.632 ± 0.085 $\surd$	0.597 ± 0.036 $\surd$
	UPU [11]	0.476 ± 0.130 $\surd$	0.589 ± 0.124 $\surd$	0.620 ± 0.078 $\surd$
	NNPU-MLP [12]	0.598 ± 0.109 $\surd$	0.620 ± 0.125 $\surd$	0.635 ± 0.073 $\surd$
	NNPU-Linear [12]	0.078 ± 0.015 $\surd$	0.072 ± 0.004 $\surd$	0.071 ± 0.007 $\surd$
	LDCE [7]	0.029 ± 0.001 $\surd$	0.029 ± 0.002 $\surd$	0.029 ± 0.001 $\surd$
	CSPU (ours)	**0.774 ± 0.045**	**0.754 ± 0.074**	**0.750 ± 0.055**



(a) (b)

Figure 3: Example images from the adopted datasets. (a) *FashionMnist* dataset; (b) *USPS* dataset.

6.3. Real-world Datasets

Here we investigate the performances of W-SVM, UPU, NNPU-MLP, NNPU-Linear, LDCE, and CSPU on the practical image classification tasks such as *FashionMnist* and *USPS* datasets. *FashionMnist* consists of 70000 28×28 grayscale images in 10 classes with each class containing 7000 images, and *USPS* contains 9298 16×16 handwritten digit images belonging to 10 classes, i.e., the digits "0"-"9". Some example images are presented in Figure 3.

In terms of *FashionMnist*, we extract the 512-dimensional GIST features for each image, then choose the images of the first class (i.e., "T-shirt") as positive, and regard the remaining images as negative. Therefore, we get 7000 positive examples and 63000 negative examples for *FashionMnist*. For *USPS*, we directly adopt the 256-dimensional pixel-wise features, in which each dimension represents the gray value of corresponding pixel. We also select the digit images of the first class (i.e., "0") as positive, and the rest images are treated as negative. As a result, there are 1553 positive examples and 7745 negative examples for *USPS*. The proportions of positive examples in these two datasets are 0.10 and 0.17, respectively, which poses a great difficulty for the compared

Table 3: The test F-measures of various PU methods on the adopted real-world datasets. The best record under each r is marked in **bold**.

Datasets	Methods	$r = 20\%$	$r = 30\%$	$r = 40\%$
FashionMnist	W-SVM [19]	0.821	0.809	0.781
	UPU [11]	0.611	0.668	0.724
	NNPU-MLP [12]	0.686	0.454	0.118
	NNPU-Linear [12]	0.753	0.754	0.766
	LDCE [7]	0.189	0.188	0.186
	CSPU (ours)	**0.824**	**0.816**	**0.795**
USPS	W-SVM [19]	0.922	0.904	0.831
	UPU [11]	0.794	0.870	0.891
	NNPU-MLP [12]	0.925	0.905	0.863
	NNPU-Linear [12]	0.756	0.755	0.753
	LDCE [7]	0.312	0.310	0.307
	CSPU (ours)	**0.937**	**0.934**	**0.922**

methods to establish unbiased classifiers. Note that the training set and the test set are split in advance with 60000 training examples and 10000 test examples for *FashionMnist*, and 7291 training examples and 2007 test examples for *USPS*.

Similar to the previous experiments, for each image dataset, the situations of $r = 20\%$, 30%, and 40% are studied. The parameters of every method have been carefully tuned to achieve the best performance. For *FashionMnist*, in CSPU, we set $\lambda = 10^{-3}$ under each r. In W-SVM, the parameter γ is set to 0.4 for every r. The parameter λ of UPU is automatically determined by the algorithm as 0.001, 0.060, and 0.022 when $r = 20\%$, $r = 30\%$, and $r = 40\%$, respectively. In NNPU-MLP and NNPU-Linear, we set the parameters β and γ to the default value 0 and 1, respectively. The parameters of LDCE are set as $\lambda = 5$ and $\beta = 0.9$ for every r. For *USPS*, the parameter λ of CSPU is set to 0.008 when $r = 20\%$ and $r = 30\%$, and when r is equal to 40%, the parameter λ is tuned to 0.02. In W-SVM, we tune γ as 0.2 for all situations of r. For UPU, the parameter λ is adaptively decided as 0.001 for $r = 20\%$, 0.003 for $r = 30\%$, and 0.022 for $r = 40\%$. The parameters of NNPU-MLP and NNPU-Linear are set as $\beta = 0$ and $\gamma = 1$. In LDCE, the parameters are optimally tuned as $\lambda = 1$ and $\beta = 0.7$

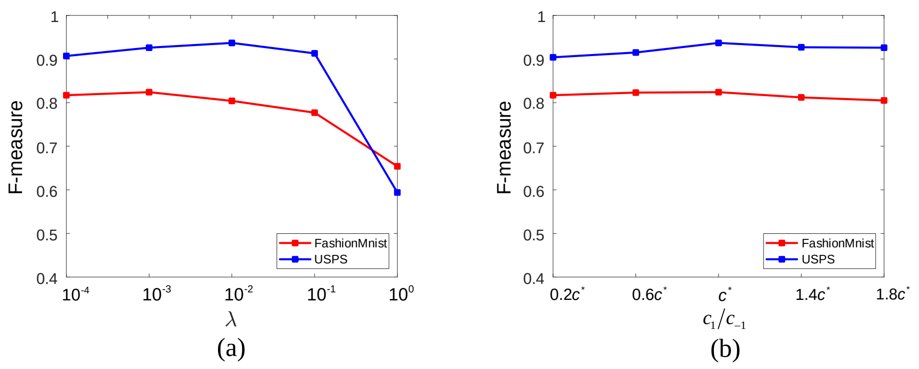


Figure 4: The parametric sensitivity of (a) λ and (b) c_1/c_{-1} of our CSPU on *FashionMnist* and *USPS* datasets.

under $r = 20\%$, $\lambda = 1$ and $\beta = 0.5$ when $r = 30\%$ and $r = 40\%$.

The F-measures achieved by the compared methods on the two image datasets are represented in Table 3, where we can clearly see that our CSPU achieves the highest F-measure among all comparators on both *FashionMnist* and *USPS*. Therefore, the proposed CSPU is effective in dealing with imbalanced real-world data.

6.4. Parametric Sensitivity

There are three trade-off parameters c_1, c_{-1}, and λ for our CSPU algorithm (14)-(20), which should be pre-tuned manually. Hence this section studies the parametric sensitivity. To investigate the influence of λ to algorithm output, we fix c_1 and c_{-1} and vary λ from 10^{-4} to 10^{0} to see the generated test F-measures of CSPU. Noting that the ratio of c_1 and c_{-1} matters in our algorithm, we study the performance variation of CSPU when $\dfrac{c_1}{c_{-1}}$ gradually deviates from the real value $c^* = \dfrac{n'_{\mathrm{n}}}{n'_{\mathrm{p}}}$. To be specific, the cases of $\dfrac{c_1}{c_{-1}} = \{0.2c^*, 0.6c^*, c^*, 1.4c^*, 1.8c^*\}$ are explored on the two real-world datasets *FashionMnist* and *USPS* when $r = 20\%$. The results are shown in Figure 4. From Figure 4(a), we observe that the change of λ from 10^{-4} to 10^{-1} will not severely influence the model performance. However, the F-measure will drop significantly if λ is larger than 0.1, so this parameter is suggested to be set to a small value above zero.

From Figure 4(b), we see that the F-measure is generally stable when $\frac{c_1}{c_{-1}}$ is around c^*, so slightly inaccurate settings of $\frac{c_1}{c_{-1}}$ will not cause dramatic model corruption for the practical implementation of our CSPU.

7. Conclusion

In this paper, we propose a novel PU learning algorithm to deal with class imbalance problem named "Cost-Sensitive PU learning" (CSPU) which imposes distinct weights on the losses regarding false negative and false positive examples. Then the PU learning is formulated as an empirical risk minimization problem with respect to the unbiased double hinge loss that makes the empirical risk to be convex. The proposed algorithm can be easily solved via off-the-shelf quadratic programming optimization toolbox. Moreover, its computational complexity and generalization bound has also been theoretically analyzed. By comparing CSPU with representative state-of-the-art PU learning methods, the superiority of the developed CSPU can be clearly observed.

Since the performance of CSPU is dependent on the weights c_1 and c_{-1} that should be pre-tuned manually, in the future, it is worthwhile to find an effective way to adaptively determine c_1 and c_{-1} instead of tuning them manually.

Acknowledgments

We would like to thank Dr. Tongliang Liu from the University of Sydney for helping proofread this paper and all anonymous reviewers for the valuable comments to improve our paper.

References

[1] J. Bekker, J. Davis, Learning from positive and unlabeled data: A survey, Machine Learning 109 (4) (2020) 719–760. doi:10.1007/s10994-020-05877-5.

[2] E. Sansone, F. G. De Natale, Z.-H. Zhou, Efficient training for positive unlabeled learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (11) (2018) 2584–2598. doi:10.1109/TPAMI.2018.2860995.

[3] W. Li, Q. Guo, C. Elkan, A positive and unlabeled learning algorithm for one-class classification of remote-sensing data, IEEE Transactions on Geoscience and Remote Sensing 49 (2) (2010) 717–725. `doi:10.1109/TGRS.2010.2058578`.

[4] B. Liu, W. S. Lee, P. S. Yu, X. Li, Partially supervised classification of text documents, in: International Conference on Machine Learning, Vol. 2, 2002, pp. 387–394. `doi:10.1385/1-59259-358-5:387`.

[5] X. Li, B. Liu, Learning to classify texts using positive and unlabeled data, in: International Joint Conference on Artificial Intelligence, Vol. 3, 2003, pp. 587–592.

[6] W. S. Lee, B. Liu, Learning with positive and unlabeled examples using weighted logistic regression, in: International Conference on Machine Learning, Vol. 3, 2003, pp. 448–455.

[7] H. Shi, S. Pan, J. Yang, C. Gong, Positive and unlabeled learning via loss decomposition and centroid estimation, in: International Joint Conference on Artificial Intelligence, 2018, pp. 2689–2695. `doi:10.24963/ijcai.2018/373`.

[8] F. He, T. Liu, G. I. Webb, D. Tao, Instance-dependent PU learning by bayesian optimal relabeling, arXiv preprint arXiv:1808.02180 (2018).

[9] B. Frénay, M. Verleysen, Classification in the presence of label noise: A survey, IEEE Transactions on Neural Networks and Learning Systems 25 (5) (2014) 845–869. `doi:10.1109/TNNLS.2013.2292894`.

[10] M. C. Du Plessis, G. Niu, M. Sugiyama, Analysis of learning from positive and unlabeled data, in: Advances in Neural Information Processing Systems, 2014, pp. 703–711.

[11] M. Du Plessis, G. Niu, M. Sugiyama, Convex formulation for learning from positive and unlabeled data, in: International Conference on Machine Learning, 2015, pp. 1386–1394.

[12] R. Kiryo, G. Niu, M. C. Du Plessis, M. Sugiyama, Positive-unlabeled learning with non-negative risk estimator, in: Advances in Neural Information Processing Systems, 2017, pp. 1675–1685.

[13] N. V. Chawla, N. Japkowicz, A. Kotcz, Special issue on learning from imbalanced data sets, ACM SIGKDD Explorations Newsletter 6 (1) (2004) 1–6. doi:10.1145/1007730.1007733.

[14] M. Liu, C. Xu, Y. Luo, C. Xu, Y. Wen, D. Tao, Cost-sensitive feature selection by optimizing F-measures, IEEE Transactions on Image Processing 27 (3) (2017) 1323–1335. doi:10.1109/TIP.2017.2781298.

[15] S. P. Parambath, N. Usunier, Y. Grandvalet, Optimizing F-measures by cost-sensitive classification, in: Advances in Neural Information Processing Systems, 2014, pp. 2123–2131.

[16] B. Liu, Y. Dai, X. Li, W. S. Lee, S. Y. Philip, Building text classifiers using positive and unlabeled examples, in: IEEE International Conference on Data Mining, Vol. 3, 2003, pp. 179–188. doi:10.1109/ICDM.2003.1250918.

[17] H. Yu, J. Han, K. C.-C. Chang, PEBL: Positive Example Based Learning for web page classification using SVM, in: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 239–248. doi:10.1145/775047.775083.

[18] C. Gong, H. Shi, T. Liu, C. Zhang, J. Yang, D. Tao, Loss decomposition and centroid estimation for positive and unlabeled learning, IEEE Transactions on Pattern Analysis and Machine Intelligence (2019). doi:10.1109/TPAMI.2019.2941684.

[19] C. Elkan, K. Noto, Learning classifiers from only positive and unlabeled data, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 213–220. doi:10.1145/1401890.1401920.

[20] C. Zhang, Y. Hou, Y. Zhang, Learning from positive and unlabeled data without explicit estimation of class prior, in: Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020.

[21] C. Gong, T. Liu, J. Yang, D. Tao, Large-margin label-calibrated support vector machines for positive and unlabeled learning, IEEE Transactions on Neural Networks and Learning Systems 30 (11) (2019) 3471–3483. doi:10.1109/TNNLS.2019.2892403.

[22] C. Gong, H. Shi, J. Yang, J. Yanga, Multi-manifold positive and unlabeled learning for visual analysis, IEEE Transactions on Circuits and Systems for Video Technology 30 (5) (2019) 1396–1409. doi:10.1109/tcsvt.2019.2903563.

[23] C. Zhang, D. Ren, T. Liu, J. Yang, C. Gong, Positive and unlabeled learning with label disambiguation, in: International Joint Conference on Artificial Intelligence, 2019, pp. 4250–4256. doi:10.24963/ijcai.2019/590.

[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling TEchnique, Journal of Artificial Intelligence Research 16 (2002) 321–357. doi:10.1613/jair.953.

[25] G. Y. Wong, F. H. Leung, S.-H. Ling, A hybrid evolutionary preprocessing method for imbalanced datasets, Information Sciences 454 (2018) 161–177. doi:10.1016/j.ins.2018.04.068.

[26] B. Zadrozny, J. Langford, N. Abe, Cost-sensitive learning by cost-proportionate example weighting, in: International Conference on Machine Learning, Vol. 3, 2003, pp. 435. doi:10.1109/ICDM.2003.1250950.

[27] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, Y. Zhou, A novel ensemble method for classifying imbalanced data, Pattern Recognition 48 (5) (2015) 1623–1637. doi:10.1016/j.patcog.2014.11.014.

[28] D. Elreedy, A. F. Atiya, A comprehensive analysis of Synthetic Minority Oversampling TEchnique (SMOTE) for handling class imbalance, Information Sciences 505 (2019) 32–64. doi:10.1016/j.ins.2019.07.070.

[29] C. Bunkhumpornpat, K. Sinapiromsaran, C. Lursinsap, Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2009, pp. 475–482. doi:10.1007/978-3-642-01307-2_43.

[30] T. Maciejewski, J. Stefanowski, Local neighbourhood extension of SMOTE for mining imbalanced data, in: IEEE Symposium on Computational Intelligence and Data Mining, 2011, pp. 104–111. doi:10.1109/CIDM.2011.5949434.

[31] G. Douzas, F. Bacao, F. Last, Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE, Information Sciences 465 (2018) 1–20. doi:10.1016/j.ins.2018.06.056.

[32] C.-F. Tsai, W.-C. Lin, Y.-H. Hu, G.-T. Yao, Under-sampling class imbalanced datasets by combining clustering analysis and instance selection, Information Sciences 477 (2019) 47–54. doi:10.1016/j.ins.2018.10.029.

[33] P. Vuttipittayamongkol, E. Elyan, Neighbourhood-based undersampling approach for handling imbalanced and overlapped data, Information Sciences 509 (2020) 47–70. doi:10.1016/j.ins.2019.08.062.

[34] E. Ramentol, Y. Caballero, R. Bello, F. Herrera, SMOTE-RSB*: A hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory, Knowledge and Information Systems 33 (2) (2012) 245–265. doi:10.1007/s10115-011-0465-6.

[35] P. Domingos, Metacost: A general method for making classifiers cost-sensitive, in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Vol. 99, 1999, pp. 155–164. doi:10.1145/312129.312220.

[36] Z.-H. Zhou, X.-Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Transactions on Knowledge & Data Engineering (1) (2006) 63–77. `doi:10.1109/tkde.2006.17`.

[37] H. Masnadi-Shirazi, N. Vasconcelos, Risk minimization, probability elicitation, and cost-sensitive SVMs, in: International Conference on Machine Learning, 2010, pp. 759–766. `doi:10.1.1.170.1360`.

[38] J. Yin, C. Gan, K. Zhao, X. Lin, Z. Quan, Z.-J. Wang, A novel model for imbalanced data classification, in: Thirty-Fourth AAAI Conference on Artificial Intelligence, 2020, pp. 6680–6687.

[39] H. Li, L. Zhang, B. Huang, X. Zhou, Cost-sensitive dual-bidirectional linear discriminant analysis, Information Sciences 510 (2020) 283–303. `doi:10.1016/j.ins.2019.09.032`.

[40] K. Cao, C. Wei, A. Gaidon, N. Arechiga, T. Ma, Learning imbalanced datasets with label-distribution-aware margin loss, Advances in Neural Information Processing Systems (2019) 1565–1576.

[41] A. Krishnamurthy, A. Agarwal, T.-K. Huang, H. Daumé III, J. Langford, Active learning for cost-sensitive classification, Journal of Machine Learning Research 20 (65) (2019) 1–50.

[42] M. Christoffel, G. Niu, M. Sugiyama, Class-prior estimation for learning from positive and unlabeled data, in: Asian Conference on Machine Learning, 2016, pp. 221–236. `doi:10.1007/s10994-016-5604-6`.

[43] J. Bekker, J. Davis, Estimating the class prior in positive and unlabeled data through decision tree induction, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[44] T. Hofmann, B. Schölkopf, A. J. Smola, A tutorial review of RKHS methods in machine learning, in: Technical Report, 2005.

[45] N. Aronszajn, Theory of reproducing kernels, Transactions of the American Mathematical Society 68 (3) (1950) 337–404. doi:10.2307/1990404.

[46] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, Journal of Machine Learning Research 7 (Nov) (2006) 2399–2434. doi:10.1007/s10846-006-9077-x.

[47] J. Lofberg, YALMIP: A toolbox for modeling and optimization in MATLAB, in: IEEE International Conference on Robotics and Automation, 2004, pp. 284–289.

[48] M. Andersen, J. Dahl, L. Vandenberghe, Cvxopt: A python package for convex optimization, (2013).

[49] Y. Ye, E. Tse, An extension of karmarkar's projective algorithm for convex quadratic programming, Mathematical programming 44 (1-3) (1989) 157–179. doi:10.1007/BF01587086.

[50] P. L. Bartlett, S. Mendelson, Rademacher and gaussian complexities: Risk bounds and structural results, Journal of Machine Learning Research 3 (Nov) (2002) 463–482. doi:10.1007/3-540-44581-1_15.