SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2022, Vol. 65 212101:1–212101:13
https://doi.org/10.1007/s11432-020-3167-1

Robust model selection for positive and unlabeled learning with constraints

Tong WEI[1], Hai WANG[2], Weiwei TU[2] & Yufeng LI[1*]

[1]*Department of Computer Science and Technology, Nanjing University, Nanjing* 210023*, China;*
[2]*4Paradigm Inc., Beijing* 100089*, China*

Received 17 July 2020/Accepted 29 September 2020/Published online 25 October 2022

Abstract Positive and unlabeled (PU) learning is the problem in which training data contains only PU samples. Although PU learning is widely used in real-world applications, its model selection remains challenging. Specifically, traditional model selection methods are often highly sensitive to the class prior as well as the data size, resulting in human overhead in hyperparameter optimization. In this paper, we present a method called ODE (robust model selection) for robust model selection in PU learning. Two novel model evaluators based on the integral probability metric and area under the curve, which are free of the class prior, are introduced, and a variance reduction method is further employed to improve the quality of model selection. In addition, we perform model selection under user-defined constraints and propose a fast halving-style searching algorithm to efficiently identify the most promising model configuration. Extensive empirical studies demonstrate that our proposed method performs more robustly and is more computationally efficient than many state-of-the-art methods.

Keywords positive and unlabeled learning, model selection, automated machine learning

Citation Wei T, Wang H, Tu W W, et al. Robust model selection for positive and unlabeled learning with constraints. Sci China Inf Sci, 2022, 65(11): 212101, https://doi.org/10.1007/s11432-020-3167-1

1 Introduction

Positive and unlabeled (PU) learning is the problem in which only positive and unlabeled data are available. PU learning has attracted increasing attention in the machine learning literature, as such data naturally arises in real applications. For example, in recommender systems [1, 2], purely positive feedback from users is collected while other feedback is unknown. In webpage categorization [3], only a subset of documents of interest are annotated while other documents are unlabeled. Similarly, many other applications involve PU data, such as software clone detection [4], time-series data [5], and disease gene identification [6].

Many PU learning methods have been developed, which can be roughly divided into two categories. (i) The first line of research is based on problem transformation. One direct approach is to identify reliable negative samples from unlabeled data, then PU learning is transformed into supervised learning [7–10], and solved by employing standard supervised learning algorithms. Another approach is to treat unlabeled data as negative and consider hidden positive samples among unlabeled data as mislabeled samples. The PU learning problem can thus be transformed to the label noise learning [11, 12]. (ii) The second line of research involves developing unbiased risk estimators. This type of research can be considered as cost-sensitive classification [13–16]. For unlabeled samples, different costs are compelled when it is predicted as positive or negative. These approaches typically achieve better performance than approaches based on problem transformation. Nevertheless, they rely on knowledge of the class prior which is usually inaccessible in real-world scenarios.

Although many PU learning algorithms have been proposed, it remains difficult to apply standard model selection for a specific PU task, which requires a large amount of human overhead. Many model

* Corresponding author (email: liyf@nju.edu.cn)

selection methods [17–20] in supervised learning and semi-supervised learning have been studied to select the best configuration; however, it is nontrivial to directly apply them to PU learning. The difficulties are mainly threefold: (1) supervised model selection methods can easily lead to overfitting when applied to PU learning because labeled data is biased and insufficient; (2) existing performance evaluators [15,16] in PU learning are highly sensitive to the class prior and often lead to inferior performance if the class prior is inaccurate, while robustness is essential in weakly supervised learning [21–24]; and (3) model selection methods must be implemented under given resource constraints, which requires an efficient and controllable algorithm.

To address these challenges, in this paper, we develop a robust PU learning algorithm under user-defined budgets called ODE (robust model selection) that attempts to identify the best PU model. First, we propose two criteria for model selection in PU learning based on the integral probability metric (IPM) [25] and area under the curve (AUC) [26]. We demonstrate that the IPM and AUC score are equivalent to unbiased performance estimators when used for PU model selection. The two proposed selection criteria are free of the class prior, thereby mitigating the problem of existing methods' sensitivity to the class prior. We further attempt to reduce the estimation variance by a control variate method. Then, we formulate the robust PU learning model selection procedure as a constrained optimization problem under user-defined budgets. Two types of resource budgets are considered for the efficiency of ODE: the total running time and the number of training samples used. A halving-style searching algorithm is subsequently developed to efficiently identify promising configurations. Extensive empirical studies on several benchmark and real-world datasets reveal the robustness of the proposed method under varying scenarios.

The contributions of this paper are summarized as follows.

- We propose two unbiased model selection criteria without the need for the class prior. We further reduce the estimation variance by the control variate, which is novel in PU learning.
- We formulate robust model selection as a constrained optimization problem and develop a halving-style algorithm to efficiently identify the most promising configuration.
- We apply the proposed ODE to real-world problems in recommender systems, and it achieves more stable performance than the state-of-the-art methods.

The remainder of this paper is arranged as follows. We start with a brief review of related work. Then, we introduce unbiased risk evaluators and present the proposed robust PU model selection method. Thereafter, we present empirical results followed by the conclusion.

2 Related work

2.1 Model selection in PU learning

Assessing the performance of a learned model constitutes a crucial part of machine learning. However, in some domains, only PU samples are available, which prohibits the use of most standard evaluation metrics.

Previous literature on PU model selection is mainly based on the random selection assumption, that is, observed positive samples are selected from the underlying positive class distribution completely at random. Therefore, the empirical risk of classifiers for negative samples can be approximated by risks of PU data. Based on this assumption, several unbiased risk estimators have been proposed [13,15,16,26,27]. Additionally, Ref. [28] proposed using a non-convex surrogate loss function in PU learning to avoid a biased solution. Furthermore, Ref. [29] proposed an approach to estimate any metric based on contingency tables. This approach can essentially be reduced to estimating the fraction of (latent) positive samples in the unlabeled set. In addition, Ref. [30] demonstrated that many standard evaluation metrics can be correctly recovered with knowledge of the class prior. Nonetheless, their method is restricted to non-traditional classifiers, that is, it directly treats unlabeled data as negative. The above-mentioned estimation methods can facilitate theoretical analysis; however, their main shortcoming is that they require the class prior, which is usually unavailable in real-world applications.

2.2 Hyperparameter optimization

Hyperparameter optimization is a central part of automated machine learning, which is becoming an emerging area in both industry and academia. Frequently used hyperparameter optimization strategies include simple search and optimization from samples [19,31].

Grid search and random search [32] are two common simple search approaches, with the former being the most traditional method of hyperparameter tuning. To obtain the optimal hyperparameter settings, grid search must enumerate every possible configuration in searching space. Discretization is necessary when the search space is continuous. Random search randomly samples configurations in the searching space. As its name suggests, random search picks one configuration completely at random for evaluation at each step and updates the best hyperparameter setting. On the aspect of optimization from samples, it iteratively generates new configurations based on previously evaluated sampled configurations. Bayesian optimization [33–35] is the most popular approach. This approach develops a probabilistic model, which maps configurations to their expected performance with uncertainty and defines an acquisition function to balance exploration and exploitation during the search. At each iteration, a new sample is generated by optimizing the acquisition function and used to update the probabilistic model once it is evaluated. However, most hyperparameter search approaches require both positive and negative data for model selection. Therefore, existing model selection methods cannot be directly applied to PU learning due to the absence of negative samples.

3 Proposed PU risk estimator

To alleviate the problem of the absence of negative samples, it is necessary to develop alternative performance evaluation metrics. In this section, we first provide the basic setting of PU learning. Then, we introduce two representative unbiased risk estimators from existing literature, i.e., uPU [15] and nnPU [16]. Finally, we propose two equivalent performance evaluators: the $\mathrm{IPM}_{\mathrm{PU}}$ and $\mathrm{AUC}_{\mathrm{PU}}$.

3.1 Problem setting

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 sample if $s_i = 1$; otherwise it is regarded as an unlabeled sample. We denote the set of positive samples $\mathcal{P}$ and unlabeled set $\mathcal{U} = \mathcal{D} - \mathcal{P}$. The positive set $\mathcal{P}$ is sampled independently from the underlying joint density $p(\boldsymbol{x} \,|\, y = 1)$, and the unlabeled set $\mathcal{U}$ is sampled from a mixture density $p(\boldsymbol{x}) = \pi p(\boldsymbol{x} \,|\, y = 1) + (1-\pi)p(\boldsymbol{x} \,|\, y = 0)$, where π represents the class prior probability and y represents the true class label of instance $\boldsymbol{x}$. As in conventional supervised learning, PU learning aims to learn a classifier $p(y = 1 \,|\, \boldsymbol{x})$ that distinguishes positive and negative data.

3.2 Unbiased PU risk estimator

Evaluating PU models using only PU data is very challenging, thus, Ref. [15] proposed an unbiased risk estimator called uPU. Let $R_{\mathrm{p}}^{+}(f) = \mathbb{E}_{\mathrm{p}}[\ell(f(\boldsymbol{x}), +1)]$ and $R_{\mathrm{n}}^{-}(f) = \mathbb{E}_{\mathrm{n}}[\ell(f(\boldsymbol{x}), -1)]$ represent the empirical risk of positive and negative samples, respectively. For brevity, we use supervised (PN) learning as the short for supervised learning in the rest of this paper. In PN learning, owing to the availability of positive and negative samples, empirical risk of classifier f, i.e., $R(f)$, can be directly approximated by

$$\widehat{R}_{\mathrm{pn}}(f) = \pi \widehat{R}_{\mathrm{p}}^{+}(f) + (1-\pi)\widehat{R}_{\mathrm{n}}^{-}(f). \tag{1}$$

In PU learning, although $\widehat{R}_{\mathrm{n}}^{-}(f)$ is unavailable, it can be indirectly approximated owing to the property $(1-\pi)R_{\mathrm{n}}^{-}(f) = R_{\mathrm{u}}^{-}(f) - \pi R_{\mathrm{p}}^{-}(f)$, and we can obtain the approximation of the PU learning risk as follows:

$$\widehat{R}_{\mathrm{pu}}(f) = \pi \widehat{R}_{\mathrm{p}}^{+}(f) - \pi \widehat{R}_{\mathrm{p}}^{-}(f) + \widehat{R}_{\mathrm{u}}^{-}(f), \tag{2}$$

which can be calculated from PU data.

3.3 Non-negative PU risk estimator

Based on the observation that the value of (2) can be negative which often results in overfitting, Ref. [16] proposed a non-negative risk estimator for PU learning called nnPU:

$$\widetilde{R}_{\mathrm{pu}}(f) = \pi \widehat{R}_{\mathrm{p}}^{+}(f) + \max\left\{0, \widehat{R}_{\mathrm{u}}^{-}(f) - \pi \widehat{R}_{\mathrm{p}}^{-}(f)\right\}. \tag{3}$$

Briefly, nnPU works by explicitly constraining the training risk of uPU to be non-negative. This estimator helps evaluate candidate models more precisely. However, both uPU and nnPU rely on knowledge

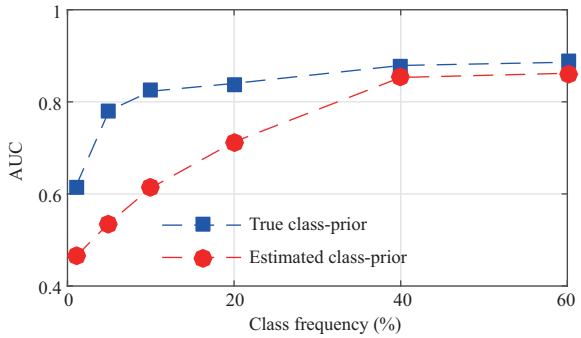

Figure 1 (Color online) AUC of nnPU [16] with true and estimated value of the class prior.

of the class prior π, which is usually unknown in practical applications [36]. Although several approaches have been proposed to estimate π from PU data [13, 36–39], inaccurate estimation usually results in severe performance degradation. In Figure 1, we illustrate the performance of nnPU with the true and estimated value of class prior. As the class-frequency $c = \frac{|\mathcal{P}|}{|\mathcal{P}|+\pi|\mathcal{U}|}$ varies, nnPU with the true value of π consistently outperforms nnPU with the estimated value of π, especially when the cardinality of the positive set $\mathcal{P}$ is small. We state this empirical finding in Observation 1.

Observation 1. Inaccurate estimation of the class prior degrades performance particularly when the number of positive samples is limited, thus it is critical to study model evaluators that are free of the class prior.

3.4 Proposed equivalent PU risk estimators

Although uPU and nnPU can be used for performance evaluation, these unbiased evaluators are susceptible to the estimation of class prior π in real-world problems. In what follows, two equivalent model selection criteria, namely, the IPM and AUC, are described in a unified form without knowledge of π. We first introduce the IPM and AUC score in PN learning and then extend them to the PU setting.

3.4.1 *Integral probability metric (PM)*

We first introduce the IPM [25], which is a widely used metric between two probability measures P and Q defined on $\mathcal{X}$ as follows:

$$\mathrm{IPM}(P, Q; \mathcal{F}) := \sup_{f \in \mathcal{F}} \left| \int_{\mathcal{X}} f(x) \mathrm{d}P(x) - \int_{\mathcal{X}} f(x) \mathrm{d}Q(x) \right|, \tag{4}$$

where $\mathcal{F}$ is a set of bounded measurable functions. In PN learning, Lemma 1 provides an important theoretical result connecting the IPM with empirical risk minimization [40].

Lemma 1 (Relationship between L-risk and IPM)**.** Let $\mathcal{F}$ be a symmetric hypothesis space and $L(y, \alpha) = \frac{2y\alpha}{y(1-2t)-1}$ where $0 < t < 1$ and $\alpha \in \mathbb{R}$. Then, we have

$$\inf_{f \in \mathcal{F}} R_L(f) = 1 - \mathrm{IPM}\left(p(\mathcal{X}|y=1), p(\mathcal{X}|y=-1); \mathcal{F}\right). \tag{5}$$

The IPM used for model selection can be defined as follows:

$$\mathrm{IPM} = \max_{f \sim \mathcal{F}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_\mathrm{P}} \left[f(\boldsymbol{x})\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}' \in \mathcal{X}_\mathrm{N}} \left[f(\boldsymbol{x}')\right], \tag{6}$$

where $\mathbb{E}_{\boldsymbol{x} \in \mathcal{X}_\mathrm{P}} \left[f(\boldsymbol{x})\right]$ and $\mathbb{E}_{\boldsymbol{x}' \in \mathcal{X}_\mathrm{N}} \left[f(\boldsymbol{x}')\right]$ represent the expected prediction score of positive and negative samples, respectively. The IPM can be intuitively explained as maximizing the margin between the expected score over positive and negative samples. It should be noted that Eq. (6) can be easily rewritten as follows:

$$\mathrm{IPM} = \max_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_\mathrm{P}} \left[\mathop{\mathbb{E}}_{\boldsymbol{x}' \sim \mathcal{X}_\mathrm{N}} \left[f(\boldsymbol{x}) - f(\boldsymbol{x}')\right] \right]. \tag{7}$$

3.4.2 *AUC score*

The AUC is one of the most frequently used evaluation metrics in practice, and is defined as

$$\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}_{\mathrm{P}}}\Big[\mathop{\mathbb{E}}_{\boldsymbol{x}'\sim\mathcal{X}_{\mathrm{N}}}[\ell(f(\boldsymbol{x})-f(\boldsymbol{x}'))]\Big], \tag{8}$$

where ℓ indicates ℓ_{0-1}, which is defined as

$$\ell_{0-1}(z)=\begin{cases}1, & z>0,\\ 1/2, & z=0,\\ 0, & z<0.\end{cases} \tag{9}$$

It should be noted that maximizing equation (8) in the hypothesis space is identical to the IPM when $\ell(z)=z$. Given the definition of the IPM and AUC, we now present our main results that both the IPM and AUC score serve as tailored model evaluators for PU learning. The result is stated in Theorem 1.

Theorem 1 (Equivalent PU model evaluator)**.** Given PU dataset $\mathcal{P}\cup\mathcal{U}$, the IPM and AUC score calculated for $\mathcal{P}$ and $\mathcal{U}$ are equivalent estimations in model selection to their supervised counterparts with a linear transformation.

Proof. According to the definition of our evaluators and the linearity of the expectation, we have

$$\begin{aligned}&\frac{1}{1-\pi}\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}_{\mathrm{P}}}\mathop{\mathbb{E}}_{\boldsymbol{x}'\sim\mathcal{X}_{\mathrm{U}}}\ell(f(\boldsymbol{x})-f(\boldsymbol{x}'))\\&=\frac{1}{1-\pi}\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}_{\mathrm{P}}}\Big[\pi\mathop{\mathbb{E}}_{\overline{\boldsymbol{x}}\sim\mathcal{X}_{\mathrm{P}}}\ell(f(\boldsymbol{x})-f(\overline{\boldsymbol{x}}))+(1-\pi)\mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathcal{X}_{\mathrm{N}}}\ell(f(\boldsymbol{x})-f(\hat{\boldsymbol{x}}))\Big]\\&=\frac{\pi}{1-\pi}\underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}_{\mathrm{P}}}\mathop{\mathbb{E}}_{\overline{\boldsymbol{x}}\sim\mathcal{X}_{\mathrm{P}}}\ell(f(\boldsymbol{x})-f(\overline{\boldsymbol{x}}))}_{I}+\underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}_{\mathrm{P}}}\mathop{\mathbb{E}}_{\hat{\boldsymbol{x}}\sim\mathcal{X}_{\mathrm{N}}}\ell(f(\boldsymbol{x})-f(\hat{\boldsymbol{x}}))}_{II}.\end{aligned}$$

The first term marked by I describes the measurement between $\mathcal{P}$ and hidden positive samples in $\mathcal{U}$. Term I can be estimated via $\mathcal{P}$ since $\forall\boldsymbol{x}\in\mathcal{P}$ are obtained from the underlying distribution of positive samples completely at random. Term II is the measurement between $\mathcal{P}$ and hidden negative samples in $\mathcal{U}$, which is the supervised counterpart of the IPM and AUC. Letting $\ell(z)=z$ be the identity function, we obtain

$$\mathrm{IPM}_{\mathrm{PN}}=\frac{1}{1-\pi}\mathrm{IPM}_{\mathrm{PU}},$$

where $\mathrm{IPM}_{\mathrm{PU}}$ and $\mathrm{IPM}_{\mathrm{PN}}$ represent the IPM in PU learning and PN learning, respectively. Letting $\ell(\cdot)$ be ℓ_{0-1}, $\mathrm{AUC}_{\mathrm{PU}}$ is an equivalent estimation of $\mathrm{AUC}_{\mathrm{PN}}$ and we obtain

$$\mathrm{AUC}_{\mathrm{PN}}=\frac{1}{1-\pi}\mathrm{AUC}_{\mathrm{PU}}-\frac{\pi}{2(1-\pi)},$$

where $\mathrm{AUC}_{\mathrm{PU}}$ and $\mathrm{AUC}_{\mathrm{PN}}$ represent the AUC score of PU learning and PN learning, respectively. Note that both $\frac{1}{1-\pi}$ and $\frac{\pi}{2(1-\pi)}$ are constants. In other words, $\mathrm{IPM}_{\mathrm{PU}}$ and $\mathrm{AUC}_{\mathrm{PU}}$ preserve the ordering of candidate models with respect to $\mathrm{IPM}_{\mathrm{PN}}$ and $\mathrm{AUC}_{\mathrm{PN}}$, respectively.

3.5 Variance reduction by control variate

We have demonstrated that $\mathrm{IPM}_{\mathrm{PU}}$ and $\mathrm{AUC}_{\mathrm{PU}}$ are equivalent performance estimators of $\mathrm{IPM}_{\mathrm{PN}}$ and $\mathrm{AUC}_{\mathrm{PN}}$, respectively. However, given an unknown parameter ς and its unbiased estimator z, i.e., $\mathbb{E}[z]=\varsigma$. It will not be accurate if the variance $\mathrm{Var}[z]$ is high. To reduce the variance, we can adopt the control variate method [41] and identify another related unbiased estimator z' such that $\mathbb{E}[z']=\zeta$, where ζ is the parameter that z' tries to estimate. We construct a new estimator parameterized by a constant η as follows:

$$z^{\star}=z+\eta(z'-\zeta). \tag{10}$$

It is straightforward to demonstrate that $z^{\star}$ remains unbiased owing to the linear property of the expectation operation:

$$\mathbb{E}[z^{\star}]=\mathbb{E}[z]+\eta\mathbb{E}[z'-\zeta]=\varsigma+\eta(\mathbb{E}[z']-\mathbb{E}[\zeta])=\varsigma. \tag{11}$$

The variance of $z^\star$ can be computed as

$$\mathrm{Var}[z^\star] = \mathrm{Var}[z + \eta(z' - \zeta)] = \eta^2 \,\mathrm{Var}[z'] + 2\eta \,\mathrm{Cov}(z, z') + \mathrm{Var}[z], \tag{12}$$

which is a quadratic form of η and achieves a global optimum $\min \mathrm{Var}[z^\star] = (1 - \rho^2_{z,z'}) \,\mathrm{Var}[z]$ when $\eta = -\frac{\mathrm{Cov}(z,z')}{\mathrm{Var}[z']}$, where $0 \leqslant |\rho_{z,z'}| \leqslant 1$ is the correlation coefficient of z and z'. Therefore, the variance is reduced, i.e., $\mathrm{Var}[z^\star] \leqslant \mathrm{Var}[z]$.

To reduce the variance $\mathrm{Var}\,[R_{\mathrm{PU}}]$ of interest, where R represents the IPM or AUC, we construct a non-traditional classifier (NTC) that discriminates samples between $\mathcal{P}$ and $\mathcal{U}$ as $g : \boldsymbol{x} \in \mathbb{R}^d \to \widetilde{y} \in [0, 1]$. The NTC treats samples in $\mathcal{U}$ as negative and $\mathcal{P}$ as positive. Let $q_p(\widetilde{y})$ and $q_u(\widetilde{y})$ be the probability density functions of the distribution of $g(\boldsymbol{x})$ with $\boldsymbol{x} \sim p(\boldsymbol{x} \,|\, y = 1)$ and $\boldsymbol{x} \sim p(\boldsymbol{x})$, respectively. Note that $p(\boldsymbol{x} \,|\, y = 1)$ and $p(\boldsymbol{x})$ are the distributions in which $\mathcal{P}$ and $\mathcal{U}$ are sampled. The expectation of the density ratio of $\widetilde{y} \sim q_p$ is

$$\mathbb{E}_{\widetilde{y} \sim q_p} \frac{q_u(\widetilde{y})}{q_p(\widetilde{y})} = \int \frac{q_u(\widetilde{y})}{q_p(\widetilde{y})} q_p(\widetilde{y}) \mathrm{d}\widetilde{y} = 1. \tag{13}$$

By substituting this into (10) as the control variate z' for the target model selection criteria estimation, we obtain

$$\begin{aligned} R^\star_{\mathrm{PU}} &= \frac{1}{|\mathcal{P}||\mathcal{U}|} \sum_{\boldsymbol{x} \in \mathcal{P}} \sum_{\boldsymbol{x}' \in \mathcal{U}} \ell(f(\boldsymbol{x}) - f(\boldsymbol{x}')) + \frac{\eta}{|\mathcal{P}|} \sum_{\boldsymbol{x} \in \mathcal{P}} \left(\frac{1 - g(\boldsymbol{x})}{g(\boldsymbol{x})} - \mathop{\mathbb{E}}_{\widetilde{y} \sim q_p} \frac{q_u(\widetilde{y})}{q_p(\widetilde{y})} \right) \\ &= \frac{1}{|\mathcal{P}||\mathcal{U}|} \sum_{\boldsymbol{x} \in \mathcal{P}} \sum_{\boldsymbol{x}' \in \mathcal{U}} \ell(f(\boldsymbol{x}) - f(\boldsymbol{x}')) + \frac{\eta}{|\mathcal{P}|} \sum_{\boldsymbol{x} \in \mathcal{P}} \left(\frac{1 - g(\boldsymbol{x})}{g(\boldsymbol{x})} - 1 \right). \end{aligned}$$

Here, we have $\eta = -\frac{\widehat{\mathrm{Cov}}(R_{\mathrm{PU}}, \frac{q_u(\widetilde{y})}{q_p(\widetilde{y})})}{\widehat{\mathrm{Var}}[\frac{q_u(\widetilde{y})}{q_p(\widetilde{y})}]}$, which can be computed from the validation set.

4 PU model selection with budget

After developing performance evaluators, we propose a robust model selection approach.

4.1 Formulation and proposed solution

Given a configuration set Ω, we aim to search for a configuration $\mathcal{C}$ that maximizes the testing performance under the searching budget. Let Performance$(\mathcal{C}, \mathcal{D})$ denote the generalization performance of configuration $\mathcal{C}$ trained on dataset $\mathcal{D}$. We can then formulate our objective as an optimization problem as follows:

$$\begin{aligned} &\max_{\mathcal{C} \in \Omega} \ \mathrm{Performance}(\mathcal{C}, \mathcal{D}) \\ &\text{s.t.} \ \ \mathrm{accumulated\_budget} \leqslant \mathrm{target\_budget}. \end{aligned} \tag{14}$$

Specifically, function Performance$(\cdot,\cdot)$ in problem (14) can be substituted by the IPM and AUC. We consider that the user-defined budget B is given as the total number of samples for training and B' as the total time allowed for model selection. B and B' can be provided either individually or together, and the budget can be manually assigned by practical resource limits, such as $B \leftarrow N$ or $B \leftarrow 2N$. Since it is infeasible to train all models from the start using the full data and then select the best models, we resort to pruning-during-training which simplifies the optimization objective and reduces the complexity. Given the resource constraints, we develop a new model selection algorithm that iteratively eliminates inferior configurations. By rewriting problem (14) and we obtain

$$\begin{aligned} &\max_{\mathcal{C} \in \Omega} \ \mathrm{Performance}(\mathcal{C}, \mathcal{D}) \\ &\text{s.t.} \ \ \Omega_1 = \Omega, \\ &\quad \Omega_{t+1} \subsetneqq \Omega_t, \ \ \forall 1 \leqslant t \leqslant T, \\ &\quad \sum_{t=1}^{T} |\Omega_t| \cdot |\mathcal{D}_t^{\mathrm{train}}| \leqslant B, \\ &\quad \sum_{k=1}^{t} \sum_{\mathcal{C}_k \in \Omega_k} \mathrm{Timing}(\mathcal{C}_k, \mathcal{D}_k^{\mathrm{train}}) \leqslant B'. \end{aligned} \tag{15}$$

Initially, Ω defines a set of candidate configurations. At the t-th iteration, Ω_t consists of a subset of configurations selected to continue to the subsequent iteration. We use Timing$(\mathcal{C}, \mathcal{D})$ to denote the time consumption of training $\mathcal{C}$ on $\mathcal{D}$. To solve problem (15), we develop a successive halving [42, 43] style algorithm that early-prunes candidate configurations by evaluating trained models using partial training data on the validation set. At each iteration, it uniformly allocates a budget to a set of configurations, evaluates the performance of all configurations, discards the worst half, and repeats until one configuration remains. Algorithm 1 summarizes the details of the proposed ODE.

Algorithm 1 The framework of ODE

Input: $\mathcal{D}$: positive and unlabeled data $\mathcal{D} = \{\boldsymbol{x}_i, s_i\}_{i=1}^N$;
Ω: configuration set;
η_t: pruning schedule, $0 < \eta_t < 1$.
Process:
Set number of iteration $T = \lceil -\log_\mu^{\eta_0 |\Omega|} \rceil + 1$;
for $t = 0, 1, \ldots, T-1$ **do**
  For each configuration $\mathcal{C}$ in Ω_t, feed $r_t = \lfloor \frac{B}{T|\Omega_t|} \rfloor$ training samples to $\mathcal{C}$;
  Train models for configurations in Ω_t and calculate the IPM or AUC R_{i,r_t} for the i-th configuration in Ω_t on validation set VAL;
  Let σ_t be a bijection on Ω_t such that $R_{\sigma_t(1),r_t} \leqslant R_{\sigma_k(2),r_t} \leqslant \cdots \leqslant R_{\sigma_t(|\Omega_t|),r_t}$;
  Set $\Omega_{t+1} = \{\mathcal{C}_i \in \Omega_t, 1 \leqslant i \leqslant |\Omega_t| : R_{\sigma_t(i),r_t} \geqslant R_{\sigma_t(\lfloor \eta_t |\Omega_t| \rfloor),r_t}\}$;
  Update η_t;
end for
return Ω_T.

After pruning less promising configurations, the algorithm allocates exponentially more resources to the remaining configurations. However, directly discarding the worst half of the configurations may be too aggressive, especially in the early stages. Therefore, we take advantage of parameter η to control the pace of pruning. Physically, $1/\eta$ corresponds to the "age" of the trained models: when $1/\eta$ is small, only several models with the largest losses will be discarded; as η increases, a large proportion of candidates will gradually be discarded. To fully investigate the influence of pruning schedules, we consider the following five different halving methods:

- Successive halving (SH). The plain successive-halving method that discards a half of the worst configurations, i.e., $\eta_t = \frac{1}{2}$.
- log-schedule. The pruning parameter is set to $\eta_t = (1 - \exp(-\frac{t}{T} \times 5)) \times (1 - \frac{1}{K}) + \frac{1}{K}$. This parameter increases most rapidly at the beginning of the model selection.
- exp-schedule. The pruning parameter is set to $\eta_t = \exp((\frac{t}{T} - 1) \times 5) \times (1 - \frac{1}{K}) + \frac{1}{K}$. The parameter increases most rapidly at the end of the model selection.
- linear-schedule. The pruning parameter is set to $\eta_t = \frac{t}{T} \times (1 - \frac{1}{K}) + \frac{1}{K}$. The parameter increases linearly as the pruning progresses.
- selfpaced-schedule. The pruning parameter is set to a constant between 0 and 1.

Remark 1. To apply configuration pruning, we assume a partial order relationship between the training set pair $\mathcal{D}_i$ and $\mathcal{D}_j$, $\mathcal{D}_i \subset \mathcal{D}_j$. Then, for any configuration $\mathcal{C}$, we have Performance$(\mathcal{C}, \mathcal{D}_i) \leqslant$ Performance$(\mathcal{C}, \mathcal{D}_j)$. As $|\mathcal{D}_j|$ increases, it approximates the true generalization performance more accurately.

4.2 Analysis

In the following, we first demonstrate that the algorithm never takes a total number of samples that exceeds the budget B by setting $K = |\Omega|$ and $T = \lceil -\log_\mu^{\eta_0 K} \rceil + 1$:

$$\sum_{t=0}^{T-1} |\Omega_t| \cdot \left\lfloor \frac{B}{T|\Omega_t|} \right\rfloor \leqslant \sum_{t=0}^{T-1} \frac{B}{T} \leqslant B. \tag{16}$$

The rationale for setting $T = \lceil -\log_\mu^{\eta_0 K} \rceil + 1$, where μ is the discounting factor of the initial pruning parameter η_0 in selfpaced-schedule, is because that $\lceil -\log_\mu^{\eta_0 K} \rceil$ is the minimum integer solution of $\eta_0 \mu^T K \leqslant 1$. In other words, exactly one candidate configuration is retained after the last iteration. For other halving schedules, we return the best configuration in the last iteration. Next, we consider the performance of the algorithm in terms of identifying the best configuration. An intuitive result is provided in [42, 43], as stated in Theorem 2.

Table 1 Dataset statistics

Dataset	#Training	#Testing	#Feature	Class ratio	Temporal	Domain	Scale
twonorm	5180	2220	20	0.496	×	Computer	Small
spambase	3220	1381	57	0.399	×	Email	Small
krvskp	2237	959	36	0.486	×	Game	Small
sat	2218	913	36	0.495	×	Score	Small
ethn	1840	790	30	0.503	×	Image	Small
titanic	1540	661	3	0.324	×	Disaster	Small
texture	1418	608	19	0.492	×	Sediment	Small
zhihu	758339	533785	62	0.015	✓	Text	Large
myhug	470910	323192	69	0.046	✓	Video	Large

Lemma 2 (Performance guarantee [42]). Let $\nu_i = \lim_{\tau\to\infty} \ell_{i,\tau}, \overline{\gamma}(t) = \max_{i=1,\dots,K} \gamma_i(t)$ and

$$\begin{aligned} z &= 2\left[\log_2(n)\right] \max_{i=2,\dots,n} i\left(1+\overline{\gamma}^{-1}\left(\frac{\nu_i-\nu_1}{2}\right)\right) \\ &\leqslant 2\left[\log_2(n)\right]\left(n+\sum_{i=2,\dots,n}\overline{\gamma}^{-1}\left(\frac{\nu_i-\nu_1}{2}\right)\right). \end{aligned}$$

If budget $B > z$, then the best arm is returned from the algorithm.

Here, the limit $\nu_i = \lim_{\tau\to\infty} \ell_{i,\tau}$ is assumed to exist and is equal to $-\frac{1}{|\mathrm{VAL}|}\sum_{i=1}^{|\mathrm{VAL}|} R(f_{\theta,t}(\boldsymbol{x}_i), y_i)$, where VAL denotes the validation set. Without loss of generality, we assume $\nu_1 < \nu_2 \leqslant \cdots \leqslant \nu_K$. Let $\gamma_i(\tau)$ be the bound of the approximation error of $\ell_{i,\tau}$ with respect to v_i as a function of τ. γ_i is defined as the point-wise smallest, non-increasing function of τ such that

$$|\ell_{i,\tau} - \nu_i| \leqslant \gamma_i(\tau), \quad \forall\tau, \tag{17}$$

and $\gamma_i^{-1}(\alpha) = \min\{\tau \in \mathbb{N} : \gamma_i(\tau) \leqslant \alpha\}$, $\forall i \in [K]$.

5 Experiments

In this section, we examine the performance of the proposed ODE method on benchmark datasets and real-world tasks. Specifically, we evaluate our algorithm in the following four aspects:

(i) **Shortcomings of existing methods.** we demonstrate that the supervised model selection method is prone to overfitting and existing PU model evaluators are sensitive to the class prior.

(ii) **Comparisons on benchmark datasets.** we compare ODE with existing model selection methods on benchmark datasets to demonstrate the robustness of ODE.

(iii) **Real-world task.** we evaluate the effectiveness of the proposed approach on recommender systems, which is a natural application of PU learning.

(iv) **Ablation studies.** we examine how pruning schedules, the variance reduction method, and different types of budgets affect the performance of ODE.

(v) **Efficiency.** we compare the efficiency of ODE to that of existing model selection methods.

The comprehensive statistics of the datasets are listed in Table 1. As demonstrated, two larger datasets in real-world problems contain temporal features and are extremely class-imbalanced in comparison with the benchmark datasets. To simulate PU learning problems in practice, we constructed PU data with varying class-frequency c. More specifically, we ran all methods by setting $c \in \{0.01, 0.05, 0.1, 0.2\}$ through random downsampling positive samples. For each c, we repeated the experiment 10 times and reported the average performance. It is observed in Table 2 that using the IPM as the model evaluator typically led to more robust performance than the AUC, therefore we report results of the IPM in following experiments. In the implementation, we applied our model selection algorithm to well established PU learning algorithms, i.e., uPU and nnPU. In particular, the hyperparameters search space is detailed in Table 3.

Table 2 Comparison between the IPM and AUC as model selection criteria. The shown numbers are AUC (%). We set the class frequency $c = 0.01$[a)]

Dataset	ethn	titanic	krvskp	sat	texture	spambase	twonorm
ODE (IPM)	70.29	**66.17**	**70.89**	99.73	**99.99**	**82.75**	**99.65**
ODE (AUC)	**70.53**	65.64	69.88	**99.86**	89.49	80.89	96.87

a) Better results are in bold.

Table 3 A summary of hyperparameter search space

Algorithm	Parameter	Feasible space	Cardinality
uPU	λ	$[-5, 5]$	50
	basis	{"lm", "gauss"}	2
nnPU	γ	$[0, 1]$	50
	β	$[0, 1]$	10

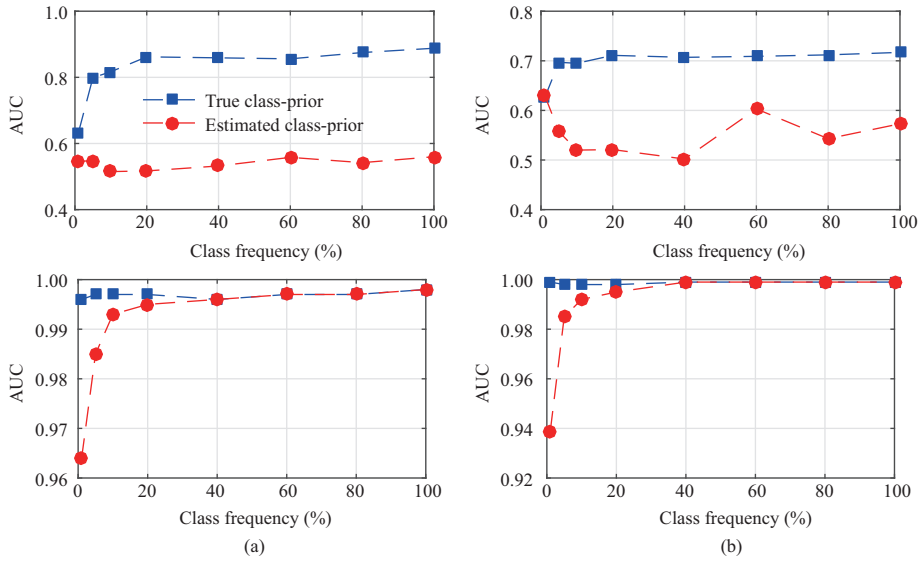

Figure 2 (Color online) Performance of uPU (a) and nnPU (b) with the true and estimated class prior on four representative datasets.

5.1 Competing methods

The following methods are compared:

• Supervised. This method trains a supervised classifier with the lightGBM [44]. It takes unlabeled data as negative and transforms the problem to standard supervised learning using hyperopt [45] for hyperparameter optimization.

• WSVM. This method proposed in [13]. This method treats each unlabeled instance as a combination of positive and negative samples. The LibSVM [46] package is used to train the model.

• uPU. This method proposed in [15] uses unbiased PU learning loss.

• nnPU. This method proposed in [16] uses non-negative PU learning loss. It is an improved version of uPU which usually overfits because the value of uPU loss can become negative.

5.2 Sensitivity to class prior of uPU and nnPU

We showcase the performance of uPU and nnPU with varying class-frequency in Figure 2 to emphasize the importance of developing class prior free model evaluators. As stated in Observation 1, the non-negative risk estimation used in nnPU is extremely sensitive to inaccurate estimation of class prior. nnPU degenerates its performance severely when the class-frequency is small. Similarly, the unbiased risk estimation developed in uPU degenerates its performance as much as 50% in the worst cases. The results demonstrate the value of our developed model evaluators.

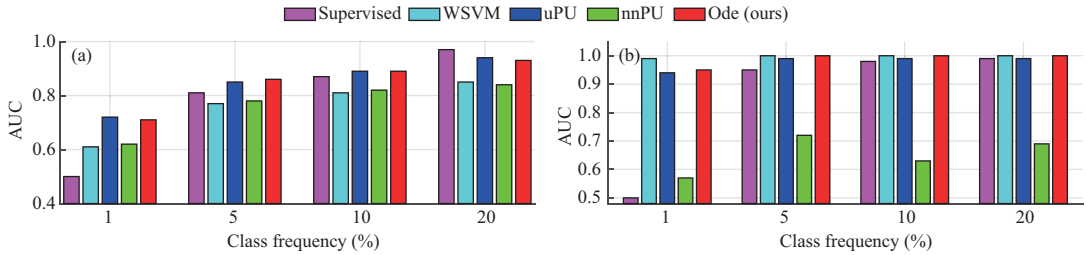

Figure 3 (Color online) Performance comparison on krvskp (a) and texture (b) datasets with varying class-frequencies.

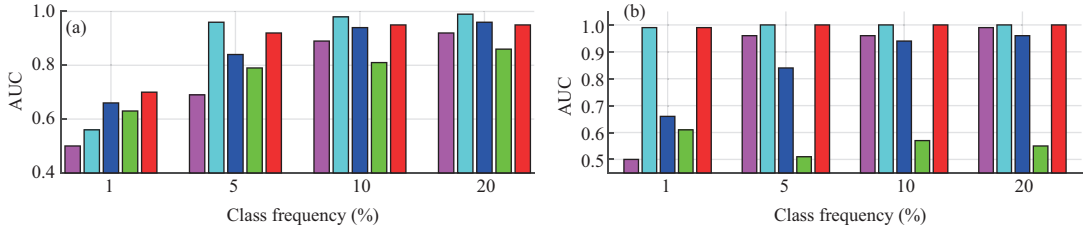

Figure 4 (Color online) Performance comparisons on ethn (a) and sat (b) datasets with varying class-frequency.

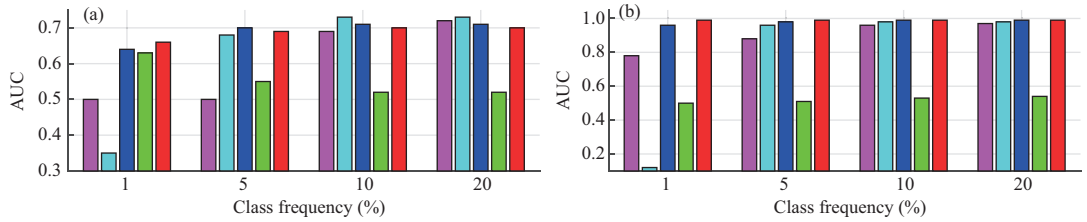

Figure 5 (Color online) Performance comparison on titanic (a) and twonorm (b) datasets with varying class-frequencies.

5.3 Comparison results

To verify the effectiveness of ODE, we compare it with the above-mentioned baselines and leading PU learning algorithms. The comparison results are reported in Figures 3–5, where the means of testing performance based on 10 random samplings are shown. The results demonstrate that the supervised model selection baseline performed poorly when only a modest set of positive samples was provided. It achieved competitive results when additional positive samples were given because sampled unlabeled samples are more likely to be negative; however, its performance was extremely unstable. This validates that directly applying a supervised hyperparameter search baseline is prone to overfitting when positive data is scarce. We implemented WSVM using a Gaussian kernel and fit the data well in most cases except when the number of positive samples was limited, facing similar problems to those of the supervised baseline. Since a deep model is employed for nnPU, it is unsurprising that nnPU generally had worse performance on small datasets (e.g., ethn, krvskp) than other algorithms. uPU generally achieves the second-best performance across different settings; however, it performed poorly on some datasets. In comparison, the proposed ODE was more robust across many datasets and class-frequencies.

5.4 Real-world tasks

Here, we examine the performance of ODE on real-world problems in recommender systems and report the comparison results in Figure 6. The main purpose of these tasks is to predict whether users are interested in documents, videos, or goods. As the WSVM algorithm has considerable training complexity, we employed a linear kernel, replacing the Gaussian kernel. In this part of experiment, we compared uPU, nnPU, and ODE using estimated class prior $\hat{\pi}$ by [36]. Particularly, on the zhihu dataset, uPU was highly sensitive to an inaccurate estimation value of π, and its performance decreased by up to 20% in terms of AUC. In contrary, our ODE outperformed all competing methods with varying class-frequency. On the myhug dataset, ODE and uPU both achieved relatively robust performance; however, that of ODE is slightly higher in the cases $c = 0.1$ and $c = 0.2$. These results verify the robustness of the proposed ODE,

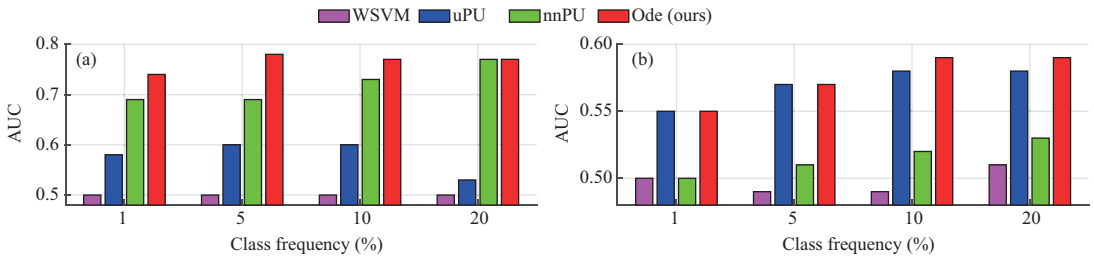

Figure 6 (Color online) Performance comparison on real-world tasks, zhihu (a) and myhug (b), with varying class-frequencies.

Table 4 Ablation study for the configuration pruning strategy. The displayed numbers are AUC (%)[a)]

Pruning schedule	ethn	titanic	krvskp	sat
SH	71.52	63.99	70.52	91.85
log-schedule	71.90	66.09	70.11	90.24
exp-schedule	72.20	**66.30**	69.74	**99.74**
linear-schedule	**72.55**	65.65	70.09	96.74
selfpaced-schedule	70.64	66.17	**71.01**	99.73

a) Best results are in bold.

Table 5 Ablation study for control variate. The displayed numbers are AUC (%)[a)]

Method	ethn	titanic	krvskp	sat
ODE (w/o control variate)	69.33	**67.69**	68.15	96.54
ODE (w/ control variate)	**72.20**	66.30	**69.74**	**99.74**

a) Best results are in bold.

which not only achieves promising results on benchmark datasets but also succeeded in real-world tasks.

5.5 Ablations on pruning schedule

Here, we study the effect of the successive-halving method on four datasets with different pruning strategies. As illustrated in Table 4, selfpace-schedule and exp-schedule achieved the best performance when compared to the baseline SH. More specifically, selfpaced-schedule improved the AUC score from 91.85% to 99.73% on sat. This indicates that selfpaced-schedule and exp-schedule postponed releasing candidate configurations to the end of the model selection phase. Finally, selfpaced-schedule was adopted in the proposed ODE for its stable performance.

5.6 Ablations on control variate

Here, we conducted experiments on four datasets to evaluate the effectiveness of the control variate. The results are reported in Table 5. It can be observed that ODE (w/ control variate) achieved significantly higher performance than ODE (w/o control variate) in three out of four datasets, which empirically confirms the efficacy of the control variate method in improving estimation robustness.

5.7 Ablations on different types of budgets

Here, we investigate the performance of ODE with different types of resources. We consider two types of resources, i.e., the number of training samples used and the running time. As illustrated in Figure 7, all pruning strategies typically exhibit improved performance as the number of resources increases. It is thus unsurprising to observe that the improvement in performance generally accelerated in the beginning and slowed at the end. This is because, with abundant resources, most strategies can identify promising configurations.

5.8 Efficiency of proposed ODE

In addition to evaluating the performance, we also examine the efficiency of the proposed ODE in comparison with competing methods, i.e., WSVM, uPU, and nnPU. The comparison results are reported in Table 6. For fair comparison, we set the sample budget $B = N$ and did not impose the time budget B'. We fixed the class-frequency $c = 0.6$ for all datasets. Table 6 reveals that ODE consistently outperformed

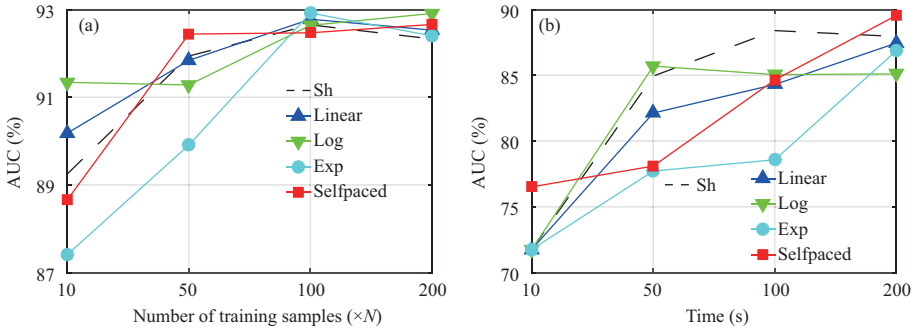

Figure 7 (Color online) Comparisons of different pruning strategies in terms of sample size (a) and running time (b).

Table 6 A summary of average training time in seconds (ratio) for PU learning algorithms based on 10 replications[a)]

Dataset	WSVM	uPU	nnPU	ODE
ethn	100.3 (6.4)	15.6 (1.0)	29.2 (1.8)	**15.4 (1.0)**
sat	101.3 (15.7)	13.5 (2.1)	7.4 (1.1)	**6.4 (1.0)**
spambase	683.8 (45.0)	36.7 (2.4)	16.9 (1.1)	**15.1 (1.0)**
texture	29.7 (26.7)	7.4 (4.0)	7.1 (6.4)	**1.1 (1.0)**
titianic	22.6 (64.8)	1.4 (17.3)	7.7 (22.0)	**0.3 (1.0)**
twonorm	1182.4 (273.7)	15.2 (3.5)	20.6 (4.7)	**4.3 (1.0)**

a) Best results are in bold.

WSVM, uPU, and nnPU in all cases. In particular, ODE was 72× faster than WSVM on average, over 5× faster than uPU, and 6.2× faster than nnPU. This demonstrates that our proposed algorithm not only achieves robust performance, but can also effectively identify promising configurations.

6 Conclusion

In this work, we formulate model selection in PU learning under resource budgets as a constrained optimization problem. To evaluate model performance based on PU data, two unbiased model evaluators are proposed that are free of the class prior. To facilitate efficient model selection, we propose a fast halving-style algorithm with five different pruning schedules. Finally, we present the results of extensive numerical studies on both benchmark datasets and real-world problems in recommender systems, which demonstrate the robustness and efficiency of the proposed algorithm.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61772262). The authors want to thank the associate editor and reviewers for their helpful comments and suggestions.

References

1 Hsieh C, Natarajan N, Dhillon I S. PU learning for matrix completion. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning, 2015. 2445–2453
2 McAuley J, Pandey R, Leskovec J. Inferring networks of substitutable and complementary products. In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015. 785–794
3 Partalas I, Kosmopoulos A, Baskiotis N, et al. LSHTC: a benchmark for large-scale text classification. 2015. ArXiv:1503.08581
4 Wei H H, Li M. Positive and unlabeled learning for detecting software functional clones with adversarial training. In: Proceedings of International Joint Conference on Artificial Intelligence, 2018. 2840–2846
5 Nguyen M N, Li X L, Ng S K. Positive unlabeled leaning for time series classification. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, 2011. 1421–1426
6 Yang P, Li X L, Chua H N, et al. Ensemble positive unlabeled learning for disease gene identification. PLoS ONE, 2014, 9: 97079
7 Liu B, Lee W S, Yu P S, et al. Partially supervised classification of text documents. In: Proceedings of the 19th International Conference on Machine Learning, 2002. 387–394
8 Liu B, Dai Y, Li X L, et al. Building text classifiers using positive and unlabeled examples. In: Proceedings of the 3rd IEEE International Conference on Data Mining ICDM, 2003. 179–188
9 Li X L, Liu B. Learning to classify texts using positive and unlabeled data. In: Proceedings of International Joint Conference on Artificial Intelligence, 2003. 587–592
10 Wei T, Shi F, Wang H, et al. MixPUL: consistency-based augmentation for positive and unlabeled learning. 2020. ArXiv:2004.09388
11 Lee W S, Liu B. Learning with positive and unlabeled examples using weighted logistic regression. In: Proceedings of the 20th International Conference on Machine Learning, 2003. 448–455
12 Shi H, Pan S J, Yang J, et al. Positive and unlabeled learning via loss decomposition and centroid estimation. In: Proceedings of International Joint Conference on Artificial Intelligence, 2018. 2689–2695

13 Elkan C, Noto K. Learning classifiers from only positive and unlabeled data. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008. 213–220
14 du Plessis M C, Niu G, Sugiyama M. Analysis of learning from positive and unlabeled data. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2014. 703–711
15 du Plessis M C, Niu G, Sugiyama M. Convex formulation for learning from positive and unlabeled data. In: Proceedings of the 32nd International Conference on Machine Learning, 2015. 1386–1394
16 Kiryo R, Niu G, du Plessis M C, et al. Positive-unlabeled learning with non-negative risk estimator. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2017. 1674–1684
17 Thornton C, Hutter F, Hoos H H, et al. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013. 847–855
18 Feurer M, Klein A, Eggensperger K, et al. Efficient and robust automated machine learning. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, 2015. 2962–2970
19 Yao Q M, Wang M S, Hugo J E, et al. Taking human out of learning applications: a survey on automated machine learning. 2018. ArXiv:1810.13306
20 Li Y F, Wang H, Wei T, et al. Towards automated semi-supervised learning. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 2019
21 Wei T, Guo L Z, Li Y F, et al. Learning safe multi-label prediction for weakly labeled data. Mach Learn, 2018, 107: 703–725
22 Dietterich T G. Robust artificial intelligence and robust human organizations. Front Comput Sci, 2019, 13: 1–3
23 Li Y F, Liang D M. Safe semi-supervised learning: a brief introduction. Front Comput Sci, 2019, 13: 669–676
24 Zhou Z H. Abductive learning: towards bridging machine learning and logical reasoning. Sci China Inf Sci, 2019, 62: 076101
25 Tolstikhin I O, Bousquet O, Gelly S, et al. Wasserstein auto-encoders. In: Proceedings of the 6th International Conference on Learning Representations, 2018
26 Xie Z, Li M. Semi-supervised AUC optimization without guessing labels of unlabeled data. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018. 4310–4317
27 Niu G, du Plessis M C, Sakai T, et al. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016. 1199–1207
28 du Plessis M C, Niu G, Sugiyama M. Analysis of learning from positive and unlabeled data. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, 2014. 703–711
29 Claesen M, Davis J, de Smet F, et al. Assessing binary classifiers using only positive and unlabeled data. 2015. ArXiv:1504.06837
30 Jain S, White M, Radivojac P. Recovering true classifier performance in positive-unlabeled learning. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence, 2017. 2066–2072
31 Conn A R, Scheinberg K, Vicente L N. Introduction to Derivative-free Optimization. Philadelphia: SIAM, 2009
32 Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res, 2012, 13: 281–305
33 Bergstra J S, Bardenet R, Bengio Y, et al. Algorithms for hyper-parameter optimization. In: Proceedings of the 24th International Conference on Neural Information Processing Systems, 2011. 2546–2554
34 Snoek J, Larochelle H, Adams R P. Practical bayesian optimization of machine learning algorithms. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, 2012. 2951–2959
35 Klein A, Falkner S, Bartels S, et al. Fast bayesian optimization of machine learning hyperparameters onlarge datasets. 2016. ArXiv:1605.07079
36 Bekker J, Davis J. Estimating the class prior in positive and unlabeled data through decision tree induction. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018. 2712–2719
37 Menon A K, van Rooyen B, Ong C S, et al. Learning from corrupted binary labels via class-probability estimation. In: Proceedings of the 32nd International Conference on Machine Learning, 2015. 125–134
38 Ramaswamy H G, Scott C, Tewari A. Mixture proportion estimation via kernel embeddings of distributions. In: Proceedings of the 33nd International Conference on Machine Learning, 2016. 2052–2060
39 du Plessis M C, Niu G, Sugiyama M. Class-prior estimation for learning from positive and unlabeled data. Mach Learn, 2017, 106: 463–492
40 Sriperumbudur B K, Fukumizu K, Gretton A, et al. On the empirical estimation of integral probability metrics. Electron J Statist, 2012, 6: 1550–1599
41 You K C, Wang X M, Long M S, et al. Towards accurate model selection in deep unsupervised domain adaptation. In: Proceedings of International Conference on Machine Learning, 2019. 7124–7133
42 Jamieson K G, Talwalkar A. Non-stochastic best arm identification and hyperparameter optimization. In: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, 2016. 240–248
43 Li L S, Jamieson K G, DeSalvo G, et al. Hyperband: a novel bandit-based approach to hyperparameter optimization. J Mach Learn Res, 2017, 18: 1–52
44 Ke G L, Meng Q, Finley T, et al. LightGBM: a highly efficient gradient boosting decision tree. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017. 3149–3157
45 Komer B, Bergstra J, Eliasmith C. Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In: Proceedings of ICML workshop on AutoML, 2014
46 Chang C C, Lin C J. LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol, 2011, 2: 1–27