Positive-Unlabeled Learning using Random Forests via Recursive Greedy Risk Minimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

The need to learn from positive and unlabeled data, or PU learning, arises in many applications and has attracted increasing interest. While random forests are known to perform well on many tasks with positive and negative data, recent PU algorithms are generally based on deep neural networks, and the potential of tree-based PU learning is under-explored. In this paper, we propose new random forest algorithms for PU-learning. Key to our approach is a new interpretation of decision tree algorithms for positive and negative data as *recursive greedy risk minimization algorithms*. We extend this perspective to the PU setting to develop new decision tree learning algorithms that directly minimizes PU-data based estimators for the expected risk. This allows us to develop an efficient PU random forest algorithm, PU extra trees. Our approach features three desirable properties: it is robust to the choice of the loss function in the sense that various loss functions lead to the same decision trees; it requires little hyperparameter tuning as compared to neural network based PU learning; it supports a feature importance that directly measures a feature's contribution to risk minimization. Our algorithms demonstrate strong performance on several datasets. Our code is available at `https://github.com/puetpaper/PUExtraTrees`.

1 Introduction

Positive and unlabeled learning (PU learning) has attracted increasing interest recently, due to its broad applications, including disease gene identification [23], landslide susceptibility modeling [26], inference of transcriptional regulatory network [22], cybersecurity [27] and many others [2].

Existing algorithms generally reduce PU learning to solving one or more supervised learning problems. A naive approach is to simply learn a classifier by treating all unlabeled examples as negative, but unlabeled examples can be either positive or negative. Various approaches have been developed to better handle the uncertainty on the labels of the unlabeled examples. A common approach follows an iterative two-step learning process: the first step identifies reliable negative examples, then the second step learns a model using the positive examples, the reliable negative examples and possibly unlabeled examples [21, 14]. Another common approach transforms the PU dataset into a weighted fully labeled dataset [19, 11, 10, 15]. The transformed dataset is often constructed so that the empirical risk on the transformed dataset provides a good estimate of the true risk. Depending on the assumptions on how PU data is generated, different transformations have been developed.

While random forests are known to perform well for many supervised learning problems and in principle tree-based learning methods can be applied in the above approaches, little work has been done to explore the potential of tree-based methods for PU learning. On the other hand, current state-of-the-art PU learning methods are often based on deep neural networks [15, 5]. This is partly due to the recent advances of deep learning, which allows training of powerful neural network models

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

with relatively little effort. In addition, while it is easy to train a neural network to minimize different loss functions, it is not clear how this can be done for tree-based methods (without overfitting the dataset), because existing tree-based methods have not been designed to directly minimize the loss. In particular, it is not clear how this can be done for nonstandard risk estimators such as the non-negative PU risk (nnPU), which has already been observed to work well with neural networks [15].

We fill this gap by providing a theoretically justified approach, which we call *recursive greedy risk minimization*, to learn random forests from PU data in this paper. Our contributions are as follows

- A new interpretation to existing decision tree learning algorithm that links the impurity reduction based learning algorithms to empirical risk minimization.
- Efficient new decision tree and random forest algorithms that directly minimize PU-data based estimators for the expected risk.
- Our approach has three desirable properties: it is robust to the choice of the loss function in the sense that various loss functions lead to the same decision trees; it requires little hyperparameter tuning as compared to neural network based PU learning; it supports a feature importance that directly measures a feature's contribution to risk minimization.
- Our algorithms demonstrate strong performance on several datasets.

We briefly review decision tree learning and PU learning in Section 2, introduce our recursive greedy risk minimization approach for decision tree learning in Section 3 and the PU version of extra trees in Section 4, present experiment results in Section 5, discuss related works in Section 6, and conclude in Section 7.

2 Background

Decision tree learning from positive and negative data A decision tree is generally constructed by recursively partitioning the training set so that we can quickly obtain subsets which are more or less in the same class.

Algorithm 1 shows a generic decision tree learning algorithm. We start with a single node associated with the entire set of training examples. When we encounter a node κ associated with a set S of training examples, we compute the prediction value at κ if the termination criterion is met. If not, we first compute an optimal split (f, t) for the dataset S, where a split (f, t) tests whether a feature f is larger than a value t, and the quality of a split is usually measured by its impurity reduction. Based on the test's outcome, we create two child nodes $\kappa_{f>t}$ and $\kappa_{f\leq t}$, split S into two subsets $S_{f>t}$ and $S_{f\leq t}$, and finally continue the learning process with $(\kappa_{f>t}, S_{f>t})$ and $(\kappa_{f\leq t}, S_{f\leq t})$.

Algorithm 1: $\texttt{LearnDT}(\kappa, S)$

Notations: κ - node; S - dataset; f - feature; t - threshold;
if *termination criterion is met* **then**
  Compute the prediction value at κ using S;
else
  Choose an optimal split (f, t);
  Create two child nodes $\kappa_{f>t}$ and $\kappa_{f\leq t}$ for κ;
  $\texttt{LearnDT}(\kappa_{f>t}, S_{f>t})$;
  $\texttt{LearnDT}(\kappa_{f\leq t}, S_{f\leq t})$;
end

Given an impurity measure $\text{Impurity}(S)$, a split (f, t)'s impurity reduction is

$$\text{IR}(f, t; S) \coloneqq \text{Impurity}(S) - \frac{|S_{f>t}|}{|S|}\text{Impurity}(S_{f>t}) - \frac{|S_{f\leq t}|}{|S|}\text{Impurity}(S_{f\leq t}). \tag{1}$$

Typically, the Gini impurity $G(S)$ and the entropy $H(S)$ are used as the impurity measure:

$$G(S) \coloneqq 1 - q_+^2 - q_-^2 = 2q_+(1 - q_+), \qquad H(S) \coloneqq -q_+ \ln q_+ - q_- \ln q_-, \tag{2}$$

where q_+ and q_- are the proportions of positive and negative examples in S respectively.

We will use $\text{IR}_{\text{Gini}}(f, t; S)$ and $\text{IR}_{\text{entropy}}(f, t; S)$ to denote the impurity reduction when using the Gini impurity and the entropy impurity respectively.

PU learning We assume that the input $\boldsymbol{x} \in \mathbb{R}^d$ and label $y \in \{-1, +1\}$ follow an unknown joint distribution $p(\boldsymbol{x}, y) = p(\boldsymbol{x} \mid y)\, p(y)$. In the fully supervised case, the training data is generally

assumed to be independently sampled from $p(\boldsymbol{x}, y)$. In the PU setting, one common assumption on the data generation mechanism [10, 9, 15] is that the positive examples $P = \{\boldsymbol{x}_i^{(\mathsf{p})}\}_{i=1}^{n_\mathsf{p}}$ are sampled independently from the P marginal $p_\mathsf{p}(\boldsymbol{x}) \coloneqq p(\boldsymbol{x} \mid y = 1)$, and the unlabeled examples $U = \{\boldsymbol{x}_i^{(\mathsf{u})}\}_{i=1}^{n_\mathsf{u}}$ are sampled independently from the marginal $p(\boldsymbol{x}) = p(\boldsymbol{x}, +1) + p(\boldsymbol{x}, -1)$. Clearly, we have $p(\boldsymbol{x}) = \pi\, p_\mathsf{p}(\boldsymbol{x}) + (1 - \pi)\, p_\mathsf{n}(\boldsymbol{x})$, where $\pi \coloneqq p(y = +1)$ is the positive rate and $p_\mathsf{n}(\boldsymbol{x}) \coloneqq p(\boldsymbol{x} \mid y = -1)$ is the N marginal.

The objective is to learn a score function $g : \mathbb{R}^d \to \mathbb{R}$ (which corresponds to a classifier that outputs $+1$ iff $g(\boldsymbol{x}) > 0$) such that it minimizes its expected risk

$$R(g) \coloneqq \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\, \ell(g(\boldsymbol{x}), y), \tag{3}$$

where $\ell : \mathbb{R} \times \{-1, +1\} \to \mathbb{R}$ is a loss function that gives the loss $\ell(v, y)$ incurred by predicting a score v when the true label is y. We focus on the quadratic loss $\ell_{\text{quad}}(v, y) = (1 - vy)^2$ and the logistic loss $\ell_{\text{logistic}}(v, y) = \ln(1 + \exp(-vy))$ in this paper, but we will occasionally consider a few other losses: the zero-one loss $\ell_{0/1}(v, y) = (1 - \operatorname{sign}(vy))/2$, the ramp loss $\ell_{\text{ramp}}(v, y) = \max\{0, \min\{1, (1 - vy)/2\}\}$, the hinge loss $\ell_{\text{hinge}}(v, y) = \max\{0, 1 - vy\}$, the double hinge loss $\ell_{\text{double}}(v, y) = \max\{0, (1 - vy)/2, -vy\}$, and the sigmoid loss $\ell_{\text{sigmoid}}(v, y) = 1/(1 + \exp(vy))$.

The risk can be written in terms of positive and unlabeled data as [10]

$$R(g) = \pi\, \mathbb{E}_{\boldsymbol{x}\sim p_\mathsf{p}(\boldsymbol{x})}\, \ell(g(\boldsymbol{x}), +1) + \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\, \ell(g(\boldsymbol{x}), -1) - \pi\, \mathbb{E}_{\boldsymbol{x}\sim p_\mathsf{p}(\boldsymbol{x})}\, \ell(g(\boldsymbol{x}), -1). \tag{4}$$

This gives the following unbiased PU-data based risk estimator

$$\widehat{R}_{\mathrm{uPU}}(g) \coloneqq \sum_{\boldsymbol{x}\in P} w_\mathsf{p} \ell(g(\boldsymbol{x}), +1) - \sum_{\boldsymbol{x}\in P} w_\mathsf{p} \ell(g(\boldsymbol{x}), -1) + \sum_{\boldsymbol{x}\in U} w_\mathsf{u} \ell(g(\boldsymbol{x}), -1), \tag{5}$$

where $w_\mathsf{p} = \pi/n_\mathsf{p}$ and $w_\mathsf{u} = 1/n_\mathsf{u}$.

While the uPU risk estimator is unbiased, it can lead to negative estimates and cause overfitting. The non-negative risk estimator alleviates this problem by enforcing the risk estimate for the negative examples to be non-negative [15]

$$\widehat{R}_{\mathrm{nnPU}}(g) \coloneqq \sum_{\boldsymbol{x}\in P} w_\mathsf{p} \ell(g(\boldsymbol{x}), +1) + \max\left\{0, \sum_{\boldsymbol{x}\in U} w_\mathsf{u} \ell(g(\boldsymbol{x}), -1) - \sum_{\boldsymbol{x}\in P} w_\mathsf{p} \ell(g(\boldsymbol{x}), -1)\right\}. \tag{6}$$

3 Recursive Greedy Risk Minimization

3.1 Decision trees for positive and negative data

We first introduce our recursive greedy risk minimization approach in the fully labeled case. Consider the empirical risk of $g : \mathbb{R}^d \to \mathbb{R}$ on a labeled training set $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$: $\widehat{R}(g) = \sum_{(\boldsymbol{x},y)\in D} w\ell(g(\boldsymbol{x}), y)$, where $w = 1/|D|$. If g predicts a constant v on a subset $S = \{(\boldsymbol{x}, y)\}$ of the training examples, then the contribution to the total empirical risk is the *partial empirical risk* $\widehat{R}(v; S) \coloneqq \sum_{(\boldsymbol{x},y)\in S} w\ell(v, y)$. The optimal constant value prediction is $v_S^* = \operatorname{argmin}_{v\in\mathbb{R}} \widehat{R}^*(v; S)$ with a minimum partial empirical risk $\widehat{R}^*(S) \coloneqq \widehat{R}^*(v_S^*; S)$.

If we switch from a constant value prediction rule to a decision stump that uses the split (f, t), then the minimum partial empirical risk for such a decision stump is $\widehat{R}^*(S_{f>t}) + \widehat{R}^*(S_{f\le t})$, thus the *risk reduction* for the split (f, t) is

$$\mathrm{RR}(f, t; S) \coloneqq \widehat{R}^*(S) - \widehat{R}^*(S_{f>t}) - \widehat{R}^*(S_{f\le t}). \tag{7}$$

In the following, when we need to make the loss function explicit, we will use subscript to indicate that. For example, $\widehat{R}^*_{\mathrm{quad}}(S)$ and $\mathrm{RR}_{\mathrm{quad}}(f, t; S)$, are the minimum partial empirical risk for a constant-valued prediction on S and the risk reduction for the split (f, t) on S when using the quadratic loss.

When we choose the (f, t) with maximum risk reduction in Algorithm 1, we obtain a recursive greedy risk minimization algorithm that recursively zooms in to an input region with a constant prediction,

then improves the prediction rule in a greedy manner by replacing it with the decision stump with minimal empirical risk in that region.

Our first main result shows that the Gini impurity reduction and entropy reduction of a split are just scaled versions of the risk reductions when using the quadratic loss and the logistic loss respectively. This implies that the standard impurity reduction based decision learning algorithm is in fact performing recursive greedy risk minimization. All proofs are in the appendix.

Theorem 1. *(a) For any $S \subseteq D$, we have $\widehat{R}^*_{\mathrm{quad}}(S) = 2|S|wG(S)$. As a consequence, for any $S \subseteq D$ and any split (f, t),*

$$\mathrm{RR}_{\mathrm{quad}}(f, t; S) = 2|S|w\mathrm{IR}_{\mathrm{gini}}(f, t; S). \tag{8}$$

(b) For any $S \subseteq D$, we have $\widehat{R}^*_{\mathrm{logistic}}(S) = 2|S|wH(S)$. As a consequence, for any $S \subseteq D$ and any split (f, t),

$$\mathrm{RR}_{\mathrm{logistic}}(f, t; S) = |S|w\mathrm{IR}_{\mathrm{entropy}}(f, t; S). \tag{9}$$

3.2 Decision trees for positive and unlabeled data

We are now ready to introduce a recursive greedy risk minimization approach to PU learning. We do this by making two changes to Algorithm 1: the fully labeled dataset S is replaced by a set $P' \subseteq P$ of positive examples and a set $U' \subseteq U$ of unlabeled examples, and the split (f, t) is chosen to optimize a PU version of the risk reduction.

uPU **risk reduction** We first consider the simpler case of the uPU estimator. Similarly to the fully labeled case, if g predicts a constant v on P' and U', then the contribution to the total empirical uPU risk is

$$\widehat{R}_{\mathrm{uPU}}(v; P', U') \coloneqq \sum_{\boldsymbol{x} \in P'} w_{\mathsf{p}} \ell(v, +1) - \sum_{\boldsymbol{x} \in P'} w_{\mathsf{p}} \ell(v, -1) + \sum_{\boldsymbol{x} \in U'} w_{\mathsf{u}} \ell(v, -1).$$

The optimal constant value prediction is $v^*_{P',U'} = \mathrm{argmin}_{v \in \mathbb{R}} \widehat{R}^*_{\mathrm{uPU}}(v; P', U')$ with a minimum empirical risk $\widehat{R}^*_{\mathrm{uPU}}(P', U') \coloneqq \widehat{R}^*_{\mathrm{uPU}}(v^*_{P',U'}; P', U')$. The uPU *risk reduction* for the split (f, t) on (P', U') is

$$\mathrm{RR}_{\mathrm{uPU}}(f, t; P', U') \coloneqq \widehat{R}^*_{\mathrm{uPU}}(P', U') - \widehat{R}^*_{\mathrm{uPU}}(P'_{f>t}, U'_{f>t}) - \widehat{R}^*_{\mathrm{uPU}}(P'_{f\leq t}, U'_{f\leq t}). \tag{10}$$

The following result shows that the optimal risk $\widehat{R}^*_{\mathrm{uPU}}(P', U')$ has a closed-form formula and is thus efficiently computable. As a consequence, the uPU risk reduction for a split (f, t) is efficiently computable.

Proposition 1. *Consider arbitrary $P' \subseteq P$ and $U' \subseteq U'$. Let $W_{\mathsf{p}} = |P'|w_{\mathsf{p}}$, $W_{\mathsf{n}} = |U'|w_{\mathsf{u}} - |P'|w_{\mathsf{p}}$, and $v^* \coloneqq \frac{W_{\mathsf{p}}}{W_{\mathsf{p}}+W_{\mathsf{n}}}$.*

(a) When using the quadratic loss, we have $v^*_{P',U'} = v^*$. In addition,

$$\widehat{R}^*_{\mathrm{uPU}}(P', U') = \begin{cases} -\infty, & v^* = +\infty \\ 4(W_{\mathsf{p}} + W_{\mathsf{n}})v^*\left(1 - v^*\right), & \text{otherwise.} \end{cases} \tag{11}$$

(b) When using the logistic loss, we have

$$\widehat{R}^*_{\mathrm{uPU}}(P', U') = \begin{cases} (W_{\mathsf{p}} + W_{\mathsf{n}})\left(-v^* \ln(v^*) - (1 - v^*)\ln(1 - v^*)\right), & 0 < v^* < 1 \\ 0, & v^* \in \{0, 1\}, \\ -\infty, & v^* > 1 \end{cases} \tag{12}$$

Intuitively, if $P' \subseteq P$ and $U' \subseteq U$ are the examples in a node κ in the decision tree, then the weights W_{p} and W_{n} serve as unbiased estimates of the probabilities $p(\boldsymbol{x} \in \kappa, y = 1)$ and $p(\boldsymbol{x} \in \kappa, y = -1)$ respectively. Thus $v^* = \frac{W_{\mathsf{p}}}{W_{\mathsf{p}}+W_{\mathsf{n}}}$ serves as an estimate of the probability of positive examples. Note that v^* is non-negative, but it can be larger than 1 though. In particular, $W_{\mathsf{p}} + W_{\mathsf{n}} = |U'|w_{\mathsf{u}} = 0$, then $v^* = +\infty$.

nnPU **risk reduction** For the case of the nnPU estimator, define

$$\widehat{R}_{\mathrm{nnPU}}(v; P', U') \coloneqq \sum_{\boldsymbol{x}\in P'} w_{\mathsf{p}}\ell(v,+1) + \max\left\{0, \sum_{\boldsymbol{x}\in U'} w_{\mathsf{u}}\ell(v,-1) - \sum_{\boldsymbol{x}\in P'} w_{\mathsf{p}}\ell(v,-1)\right\}.$$

We can then similarly define the risk reduction for the nnPU estimator.

The following result shows that in the nnPU case, the optimal risk $\widehat{R}^*_{\mathrm{uPU}}(P', U')$ also has a closed-form formula and is thus efficiently computable.

Proposition 2. *Using the same notations as in Proposition 1, we have the following results.*

(a) When using the quadratic loss, we have

$$\widehat{R}^*_{\mathrm{nnPU}}(P', U') = \begin{cases} 0, & v^* > 1 \\ 4(W_{\mathsf{p}} + W_{\mathsf{n}})v^*\left(1 - v^*\right), & \text{otherwise.} \end{cases} \tag{13}$$

(b) When using the logistic loss, we have

$$\widehat{R}^*_{\mathrm{nnPU}}(P', U') = \begin{cases} (W_{\mathsf{p}} + W_{\mathsf{n}})\left(-v^*\ln(v^*) - (1 - v^*)\ln(1 - v^*)\right), & 0 < v^* < 1 \\ 0, & \textit{otherwise}. \end{cases} \tag{14}$$

It is important to note that while in the uPU case, if P' is partitioned into P'_1 and P'_2, and U' into U'_1 and U'_2, then we have $\widehat{R}_{\mathrm{uPU}}(u; P', U') = \widehat{R}_{\mathrm{uPU}}(u, P'_1, U'_1) + \widehat{R}_{\mathrm{uPU}}(u, P'_2, U'_2)$, we only have $\widehat{R}_{\mathrm{nnPU}}(u; P', U') \leq \widehat{R}_{\mathrm{nnPU}}(u, P'_1, U'_1) + \widehat{R}_{\mathrm{nnPU}}(u, P'_2, U'_2)$. Thus recursive greedy risk minimization minimizes an upper bound of the partial empirical risk rather than the partial empirical risk. This can be considered as a regularization mechanism. In fact, we observe nnPU risk to work better than the uPU risk in our experiments, which is consistent with the findings in [15].

Optimizing the split With the above closed-form formula, we can then efficiently compute the risk reduction for a given split (f, t) on (P', U'). While there are infinitely many possible t values to choose from, note that they often split the dataset in the same way, and if f's values split $\mathbb{R}$ into multiple intervals, we only need to consider $t_1 < t_2 < \ldots < t_m$, each chosen from an interval so that we cover all possible cases. While computing the risk reduction for each (f, t_i) takes $O(|P'| + |U'|)$ time, we can efficiently get the risk reductions for all of $(f, t_1), \ldots, (f, t_m)$ in $O((|P'| + |U'|)\ln(|P'| + |U'|) + m)$ time, by sorting the examples according to their f values, and going through the thresholds in a sorted order. See the Appendix for details.

Invariance properties An interesting property of the greedy recursive learning approach is that it is robust to the choice of the loss function in the sense that different loss functions can lead to identical decision trees, as shown in the result below.

Proposition 3. *The $\widehat{R}^*_{\mathrm{uPU}}(P', U')$ value is the same for the logistic loss and the sigmoid loss.*

Another interesting property is that the optimal prediction among -1 and +1 are often the same when we use different estimators and different loss functions.

Proposition 4. *If a node contains the examples $P' \subseteq P$ and $U' \subseteq U$, v^* is as defined in Proposition 1, and the prediction is chosen from $\{-1, +1\}$ to minimize the empirical risk estimate on P' and U', then the optimal prediction is $2\mathbb{1}(v^* > 0.5) - 1$ for both the* uPU *and* nnPU *risk estimate and all the loss functions in Section 2. Intuitively, we predict +1 iff the estimated proportion of positive data at the current node is larger than 0.5.*

4 PU Extra Trees

With our PU decision tree algorithms, we can obtain a PU random forest algorithm by adapting the random forest algorithm [3] for positive and negative data: repeatedly obtain bootstrap samples for both the positive data and the unlabeled data, then train a PU decision tree for each sample with optimal split chosen over a randomly sampled subset of features each time. While choosing the optimal split from a subset of features makes training each decision tree faster, it is still the most expensive part in the tree construction process and can potentially be very slow on a large dataset. We describe a more efficient version that we used in our experiments below. In addition, we introduce a new feature importance score that directly measures a feature's contribution to risk minimization.

PU ET (Extra Trees) We develop a more efficient random forest algorithm by using the randomization trick in the Extra Trees algorithm [12] to further reduce the computational cost for finding an optimal split below. Besides sampling only a subset of features, only one random threshold for each sampled feature is considered. We implemented a more general version which allows sampling multiple random thresholds for sampled feature. A complete pseudocode for PU ET is given in the Appendix.

Termination criteria Decision trees can overfit the training data if allowed to grow until each leaf has just one example. Learning generally terminated early to alleviate overfitting. In our implementation, we terminate when all feature values are constant, when a maximum tree depth $d_{\max}$ is reached, when the node size falls below a threshold $s_{\min}$, or when the node is *pure*. A node is said to be pure if the impurity measure takes value $-\infty$ in the uPU setting, or value 0 in the nnPU setting.

Risk reduction importance Our new feature importance, called *risk reduction importance*, is the total risk reduction for a feature across all nodes in all the trees. This is similar to the Gini importance, which is the total reduction in Gini impurity for a feature across all nodes in all the trees. However, the risk reduction importance directly measures a feature's contribution to risk minimization, while the Gini importance does not. In addition, the Gini impurity does not take the dataset size into account, and thus a feature often used in splitting small datasets may appear very important as compared to features used in splitting large datasets, even though it does not contribute as much to reducing the risk and improving classification performance. As a generalization of Gini importance, we can define the normalized risk reduction as the risk reduction divided by the total weight of the examples, then we can similarly define the normalized risk reduction importance.

5 Experiments

In this section we compare PU Extra Trees with several other PU learning methods, including a baseline method uPU [10, 9] and two state of the art methods nnPU [15] and Self-PU [5]. We ran experiments on heterogeneous devices. In particular, random forests were trained using 32GB RAM and one of Intel i7-10700, Intel i7-11700 or AMD Epyc 7702p CPU. Neural networks were trained on one of NVIDIA RTX A4000 or NVIDIA RTX A6000 GPU due to the lack of identical devices. While the running times are not always comparable and thus not provided in the current version of the paper, some small scale experiments suggest that training a single PU ET generally takes less time than training a single neural net used in the experiments.

Datasets We consider a selection of common datasets for classification from LIBSVM [4], as well as MNIST digits [18], the intrusion detection dataset UNSW-NB15 [24] and CIFAR-10 [16] to demonstrate the versatility of our method. Table 1 is a summary of the benchmark datasets.

Table 1: Benchmark datasets. *: random 80%-20% train-test split was used as no train-test splits were provided.

Name	# Train	# Test	# Feature	π
mushrooms*	6499	1625	112	0.52
20News	11 314	7 532	300	0.56
covtype.binary*	464 809	116 203	54	0.51
epsilon	400 000	100 000	2 000	0.5
MNIST	60 000	10 000	784	0.5
CIFAR-10	50 000	10 000	3 072	0.4
UNSW-NB15	175 340	82 331	39	0.68

The 20News, epsilon, MNIST and CIFAR-10 datasets were processed in the same way as in [15] for consistency. For UNSW-NB15, only numerical features were retained and all attack types make up the P class and the benign data makes up the N class; for mushroom and covtype, the most prominent class makes up the P class and epsilon dataset is provided with P and N classes. Each feature in the covtype and UNSW-NB15 datasets was scaled between 0 and 1 when using uPU and nnPU as there is a large mismatch in scales between the features which seemed to significantly reduce performance of the methods. No such scaling of the data is required for PU ET as performance does not seem to be affected in such cases.

To convert the PN data to PU data we follow the experimental setup in [15] whereby 1000 positive examples are randomly sampled to form the P set, and the entire dataset is used for the U set. In practice it may be the case that π is not known a-priori, in which case there are methods for estimating π using only PU data, for example, in [11], or [6].

Model hyperparameters Following common practice [25, 12], the default hyperparameters for PU ET forest are: 100 trees, no explicit restriction on the maximum tree depth, sample $\lceil \sqrt{d} \rceil$ features out of a total of d features and sample 1 threshold value when computing an optimal split,

The architectures for the neural networks used in uPU, nnPU and Self-PU were copied from [15] for the 20News, epsilon, MNIST and CIFAR-10 datasets. For the remaining datasets we used a 6-layer MLP with ReLU [17, 13] (more specifically, $d - 300 - 300 - 300 - 300 - 1$). For each dataset the neural networks were trained for 200 epochs. The batch size, learning rate, use of batch-norm (for datasets not included in [15]), weight decay and choice of optimiser were tuned for each dataset.

It is noteworthy that we spend little effort to tune PU ET hyperparameters and simply use the same hyperparameters on all datasets. In addition, PU ET seems to be robust to hyperparameter choice in some ad hoc experiments. On the other hand, neural nets are sensitive to hyperparameter choice, and significant tuning is often needed for each dataset.

5.1 Classification performance

To take into account of the randomness in the learning algorithms, we train each model five times on the same dataset, then report the mean and standard deviation of the accuracy and the F-score. Results for uPU, nnPU and Self-PU were reproduced to the best of our ability, particularly on the datasets that were included in the original results section of the respective papers.

Effect of loss function and risk estimator on PU ET We first evaluate the effect of different loss functions and risk estimators for PU ET. The results are shown in Table 2 and Table 3. PU ET with nonnegative risk estimator seems to perform about as well using either quadratic or logistic loss Unbiased risk estimator leads to overfitting, but less so with logistic loss. Logistic loss leads to better performance than quadratic loss when using unbiased risk estimator

Table 2: Accuracy mean% (sd) on the test set for PU ET using various PU data based risk estimators. †: using 20 trees and $d_{\max} = 20$ due to computational restrictions.

Dataset	Unbiased risk estimator		Nonnegative risk estimator	
	Quadratic Loss	Logistic Loss	Quadratic Loss	Logistic Loss
Epsilon	50.65 (0.31)†	53.35 (1.99)†	57.39 (0.76)	57.83 (0.70)
20News	43.64 (0.05)	73.44 (0.87)	83.34 (0.22)	83.34 (0.31)
Covtype	53.54 (1.48)†	71.38 (1.06)†	76.51 (0.52)	75.63 (0.52)
Mushroom	0.607 (0.48)	99.02 (0.32)	99.70 (0.24)	99.36 (0.48)
MNIST	50.74 (0.0)	89.05 (0.61)	93.60 (0.39)	94.01 (0.19)
CIFAR-10	60.0.(0.01)	75.30 (0.66)	79.74 (0.37)	79.86 (0.39)
UNSW-NB15	47.18 (0.15)	83.89 (1.03)	82.24 (0.86)	81.51 (1.58)

Table 3: F mean% (sd) on the test set for PU ET using various PU data based risk estimators. †: 20 trees and $d_{\max} = 20$ due to computational restrictions.

Dataset	Unbiased risk estimator		Nonnegative risk estimator	
	Quadratic Loss	Logistic Loss	Quadratic Loss	Logistic Loss
Epsilon	4.18 (2.32)†	21.43 (14.33)†	39.52 (1.19)	40.25 (1.38)
20News	0.43 (0.17)	72.65 (1.11)	85.33 (0.22)	85.26 (0.28)
Covtype	20.15 (6.08)†	65.78 (2.35)†	75.19 (0.37)	74.20 (0.58)
Mushroom	39.02 (1.19)	99.04 (0.32)	99.71 (0.24)	99.38 (0.47)
MNIST	0.0 (0.0)	88.06 (0.74)	93.49 (0.42)	93.90 (0.19)
CIFAR-10	0.02 (0.04)	61.33 (1.44)	71.31 (0.43)	71.67 (0.53)
UNSW-NB15	7.97 (0.47)	86.33 (0.87)	85.65 (0.59)	85.17 (1.15)

Comparison of PU ET with other methods We compare PU ET with nnPU risk estimator and quadratic loss against uPU neural nets, nnPU neural nets and Self-PU. The results are shown in

Table 4 and Table 5. PU ET shows strong performance in terms of both accuracy and F score on a wide variety of datasets compared to neural network based models.

Table 4: Accuracy% mean (sd) on the test set for PU ET with nonnegative risk estimator and quadratic loss and three neural network based PU learning methods. *: Self-PU was run without self-calibration due to limitations with available hardware.

Dataset	PU ET	Neural Network		
		uPU	nnPU	Self-PU
Epsilon	57.39 (0.76)	61.79 (2.57)	73.80 (0.49)	50.05 (0.0)
20News	83.34 (0.22)	44.95 (1.09)	74.11 (3.65)	62.87 (2.89)
Covtype	76.51 (0.52)	72.38 (1.23)	71.64 (0.66)	54.67 (0.40)
Mushroom	99.70 (0.24)	98.09 (1.03)	99.04 (0.54)	99.25 (0.61)
MNIST	93.60 (0.39)	55.95 (1.17)	93.08 (0.46)	93.13 (0.41)
CIFAR-10	79.74 (0.37)	62.68 (2.95)	81.87 (04.20)	88.22 (0.51)*
UNSW-NB15	82.24 (0.86)	76.62 (0.01)	76.63 (0.02)	76.73 (0.18)

Table 5: F mean% (sd) on the test set for PU ET with nonnegative risk estimator and quadratic loss and three neural network based PU learning methods. *: Self-PU was run without self-calibration due to limitations with available hardware.

Dataset	PU ET	Neural Network		
		uPU	nnPU	Self-PU
Epsilon	39.52 (1.19)	43.44 (8.14)	71.53 (0.46)	0.00 (0.00)
20News	85.33 (0.22)	12.95 (3.82)	78.05 (3.81)	71.78 (6.85)
Covtype	75.19 (0.37)	69.56 (1.77)	68.29 (0.93)	0.501 (2.55)
Mushroom	99.71 (0.24)	98.13 (1.02)	99.06 (0.53)	99.27 (0.60)
MNIST	93.49 (0.42)	19.13 (3.82)	92.77 (0.60)	92.96 (0.47)
CIFAR-10	71.31 (0.43)	32.43 (25.03)	76.01 (7.62)	85.51 (0.60)*
UNSW-NB15	85.65 (0.59)	82.47 (0.01)	82.48 (0.02)	80.18 (0.28)

5.2 Feature Importance

Interpretability is a desirable property of machine learning algorithm, but neural networks are often hard to interpret. In contrast, tree based methods are generally easier to interpret. We illustrate our risk reduction importances of both PU ET and PN (positive-negative) ET on the UNSW-NB15 and the MNIST datasets (lighter color indicates a higher score) in Figure 1 and Figure 2 respectively .

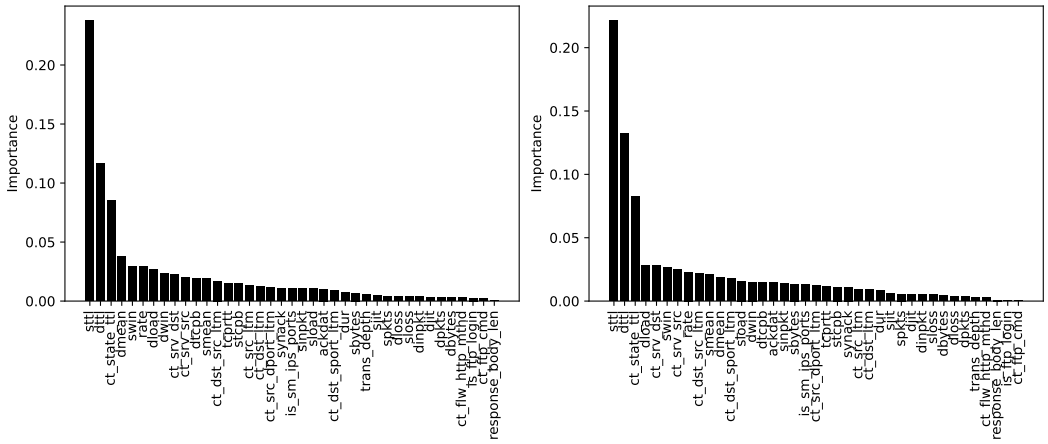

Figure 1: Risk reduction importances on UNSW-NB15. Left: PU ET. Right: PN ET.

For MNIST, the plot of the importance scores for PU ET for each digit often suggests the shape of the digit. On the other hand, the Appendix provides some additional figures that show that the

normalized risk reduction importance makes many more pixels more important. This observation is consistent with our discussion in Section 4. Interestingly, the importances for PU ET appear to be very similar to those for PN ET (visually similar importance maps for MNIST, and qualitatively similar importance values which give similar feature rankings for UNSW-NB15). This suggests that the PU model is likely quite similar to the PN model.

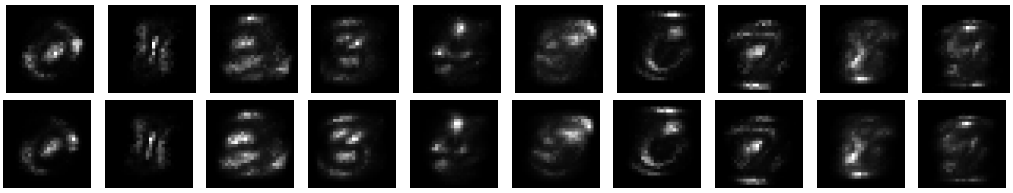

Figure 2: Risk reduction importances learnt from ET classifier on MNIST digits. Top: PU ET with nonnegative risk estimator and quadratic loss. Bottom: PN ET with Gini impurity.

6 Related Work

PU learning was initially studied in [8, 7, 20], and many approaches have been developed based on different assumptions on the data generation mechanism.

The two-step approach discussed in the introduction have been adopted in many papers with numerous instantiations. We refer the readers to [2] for a comprehensive list.

Another important line of work, which is most related to our work and has achieved the state-of-the-art performance recently, minimizes a risk estimate based on a weighted dataset of positive and negative examples obtained by transforming the PU dataset [19, 11, 10, 15]. Our work follows the risk minimization approach, but we develop general risk minimization algorithms for learning tree models, that allows exploiting the under-explored potential of random forests.

Numerous works have been done on tree-based methods, and random forest algorithms are among the most successful ones [3, 12, 1]. In the PU learning literature, PU decision trees were explored in early works [7, 20], and PU bagging [22] has been explored for SVMs [22] and decision trees [26]. All of these approaches do not aim to directly minimize the risk.

Our recursive greedy risk minimization framework gives a new interpretation to existing tree learning algorithms for positive and negative data, and allows us to develop new algorithms for positive and unlabeled data. In addition, our approach allows us to formulate a new feature importance score that directly measures a feature's contribution to risk minimization.

7 Conclusions

Random forests perform very well on fully supervised learning methods, but its potential in PU learning has been under-explored due to the strong emphasis on neural networks and the lack of a principled method for learning decision trees from PU data. We fill this gap in this paper. We first consider learning from positive and negative data, and develop a recursive greedy risk minimization approach for learning decision trees, and show that with suitable choice of loss functions, its instantiations are equivalent to standard impurity-based decision tree learning algorithms. We extend this approach to the PU setting, and develop a very efficient random forest algorithm which requires little hyperparameter and yet strong performance against the state-of-the-art PU learning algorithms. Our approach supports a new feature importance score that directly measures a feature's contribution to risk mnimization.

Our work opens up some new opportunities to build tree models. First, decision trees can easily overfit the training set, and various heuristics have been used in practice to alleviate overfitting. Our recursive greedy risk minimization framework provides an alternative way to control the size of the learned decision by always expanding the node with the largest risk reduction until the risk reduction is smaller than a threshold or when the tree size is larger than a threshold. Second, we focused on quadratic loss and logistic loss in our experiments, but we can also study alternative losses such as the hinge loss, or double hinge loss.

References

[1] Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *arXiv preprint: arXiv.1610.01271*, 2016.

[2] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[5] Xuxi Chen, Wuyang Chen, Tianlong Chen, Ye Yuan, Chen Gong, Kewei Chen, and Zhangyang Wang. Self-pu: Self boosted and calibrated positive-unlabeled training. In *International Conference on Machine Learning*, pages 1510–1519. PMLR, 2020.

[6] Marthinus Christoffel, Gang Niu, and Masashi Sugiyama. Class-prior estimation for learning from positive and unlabeled data. In *Asian Conference on Machine Learning*, pages 221–236. PMLR, 2016.

[7] Francesco De Comité, François Denis, Rémi Gilleron, and Fabien Letouzey. Positive and unlabeled examples help learning. In *International conference on algorithmic learning theory*, pages 219–230. Springer, 1999.

[8] François Denis. PAC learning from positive statistical queries. In *International conference on algorithmic learning theory*, pages 112–126. Springer, 1998.

[9] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *International conference on machine learning*, pages 1386–1394. PMLR, 2015.

[10] Marthinus C Du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. *Advances in neural information processing systems*, 27, 2014.

[11] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, 2008.

[12] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[13] Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

[14] Fengxiang He, Tongliang Liu, Geoffrey I Webb, and Dacheng Tao. Instance-dependent pu learning by bayesian optimal relabeling. *arXiv preprint arXiv:1808.02180*, 2018.

[15] Ryuichi Kiryo, Gang Niu, Marthinus C du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. *arXiv preprint arXiv:1703.00593*, 2017.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pages 448–455, 2003.

[20] Fabien Letouzey, François Denis, and Rémi Gilleron. Learning from positive and unlabeled examples. In *International Conference on Algorithmic Learning Theory*, pages 71–85. Springer, 2000.

[21] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Building text classifiers using positive and unlabeled examples. In *Third IEEE international conference on data mining*, pages 179–186. IEEE, 2003.

[22] Fantine Mordelet and J-P Vert. A bagging svm to learn from positive and unlabeled examples. *Pattern Recognition Letters*, 37:201–209, 2014.

[23] Fantine Mordelet and Jean-Philippe Vert. Prodige: Prioritization of disease genes with multitask machine learning from positive and unlabeled examples. *BMC bioinformatics*, 12(1):1–15, 2011.

[24] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] Bangyu Wu, Weirong Qiu, Junxiong Jia, and Naihao Liu. Landslide susceptibility modeling using bagging-based positive-unlabeled learning. *IEEE Geoscience and Remote Sensing Letters*, 18(5):766–770, 2021.

[27] Zhiang Wu, Jie Cao, Yaqiong Wang, Youquan Wang, Lu Zhang, and Junjie Wu. hpsd: a hybrid pu-learning-based spammer detection model for product reviews. *IEEE transactions on cybernetics*, 50(4):1595–1606, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See e.g., Proposition 4.
   (b) Did you include complete proofs of all theoretical results? [No] To be included in supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See URL in Abstract.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Reported standard deviation after running experiments multiple times, see e.g., Table 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5. Systematic comparison of the running time of the algorithms will be performed.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]