Learning from biased positive-unlabeled data via threshold calibration

Paweł Teisseyre[1] **Timo Martens**[2] **Jessa Bekker**[2] **Jesse Davis**[2]
[1]Institute of Computer Science,
Polish Academy of Sciences,
01-248 Warsaw,
Poland
[2]KU Leuven, Dept. of Computer Science,
Leuven.AI,
B-3000 Leuven,
Belgium

Abstract

Learning from positive and unlabeled data (PU learning) aims to train a binary classification model when only positive and unlabeled examples are available. Typically, learners assume that there is a labeling mechanism that determines which positive labels are observed. A particularly challenging setting arises when the observed positive labels are a biased sample from the positive distribution. Current approaches either require estimating the propensity scores, which are the instance-specific probabilities that a positive example's label will be observed, or make overly restrictive assumptions about the labeling mechanism. We make a novel assumption about the labeling mechanism which we show is more general than several commonly used existing ones. Moreover, the combination of our novel assumption and theoretical results from robust statistics can simplify the process of learning from biased PU data. Empirically, our approach offers superior predictive and run time performance compared to the state-of-the-art methods.

1 INTRODUCTION

Positive-unlabeled (PU) learning (Bekker and Davis, 2020) considers the problem of learning a binary classifier from incompletely labeled training data. Specifically, the learner is assumed to only have access to a (possibly small) number of labeled positive examples, and unlabeled data that contains examples belonging to both classes. As an illustrative case where PU data may arise, consider analyzing side effects of a drug. Some patients who experience side effects will report them. Others will not (e.g., they were mild). Therefore, it is unsafe to assume that the absence of a report of side effects means that a patient did not experience them. A surprisingly large number of applications are characterized by the presence of PU data including text and image classification (Li and Liu, 2003; Fung et al., 2006; Chiaroni et al., 2018), healthcare (Akujuobi et al., 2020), surveys (Sechidis et al., 2017), fault detection (Jaskie et al., 2021) as well as numerous bioinformatics and computational biology tasks (Li et al., 2021) (e.g., drug interaction prediction (Hameed et al., 2017)). This has spurred significant interest in algorithms that learn from such data (Liu et al., 2003; Elkan and Noto, 2008; Mordelet and Vert, 2014; Kato et al., 2019; Bekker et al., 2019; Gerych et al., 2022).

A realistic, yet challenging PU learning scenario arises when the observed positive labels are a biased sample from the positive distribution, which is called the selected at random setting (SAR) (Bekker et al., 2019). In this setting, each positive example is associated with an instance-dependent probability called the propensity score, which specifies the probability of observing the instance's label. Unfortunately, learning from SAR data remains a challenging problem and existing approaches have several drawbacks. On the one hand, it is possible to restrict the types of considered propensity functions (Gerych et al., 2022; He et al., 2018), which simplifies the problem such that only a classifier needs to be learned. Yet, these functions are likely misaligned with the true function that governs which labels are observed. This can lead to reduced performance when these assumptions are violated. On the other hand, it is possible to make less restrictive assumptions about the propensity function. This introduces the complexity that the learned classifier ei-

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

ther requires the propensity scores (Bekker et al., 2019; Gong et al., 2021) or the class prior (Kato et al., 2019). In practice, these quantities must be estimated from data, which is computationally expensive. Hence, existing approaches do not offer both good predictive performance for reasonable assumptions and computational efficiency.

In this paper, we present a computationally efficient approach to learning from SAR data that does not make overly restrictive assumptions about the form of the propensity functions. Concretely, we make three contributions. First, we propose a novel assumption on the considered propensity function forms that contains existing restrictions as special cases. Second, we show that theoretical results from robust statistics in combination with our novel, more general assumption reduces the problem to (1) training a probabilistic classifier from the PU data and (2) setting a threshold on the learned classifier's output to make a hard prediction (i.e., an example is positive or negative). Third, we present an efficient, novel method to set this threshold from unlabeled data. Empirically, our approach outperforms its competitors both in predictive performance and run time. Moreover, we show that our approach, which we named `NTC-`τ`MI`, is robust to violations of its assumptions. Finally, our code is publicly available: `https://github.com/TimoM99/PUBiasCalibration`.

2 PRELIMINARIES

In PU Learning, an example $\{X{=}x, Y{=}y, S{=}s\}$ is characterized by its features $X{\in}\mathbb{R}^d$, class $Y{\in}\{0,1\}$ and whether its label is observed $S{\in}\{0,1\}$. $S{=}1$ denotes that the class Y is observed and thus the example belongs to the positive class, i.e., $P(Y{=}1|X{=}x, S{=}1){=}1$, which we call the PU property. For $S{=}0$, one can have either $Y{=}0$ or $Y{=}1$. Variables X, Y, S are represented in uppercase and their assignment x, y, s in lower case. Sets of assignments are denoted in bold, so $\{\mathbf{x}, \mathbf{s}\}$ is a PU dataset. We assume the single-training-set scenario, i.e.: $\{\mathbf{x}, \mathbf{y}\}$ is an i.i.d. sample of the real population.

Learning from PU data is nontrivial: an instance can be unlabeled either because it is a negative instance, or because it was not selected by the labeling mechanism. Which positive examples are labeled depends on the labeling mechanism. The probability of being selected to be labeled is characterized by the propensity score $e(x) = P(S{=}1|x, Y{=}1)$. Applying Bayes' rule to the propensity score and using the PU property results in:

$$P(Y = 1|x) = \frac{1}{e(x)} P(S = 1|x). \tag{1}$$

If the propensity scores are known, a model for $P(Y{=}1|x)$ can be obtained by scaling a model for $P(S{=}1|x)$ by $1/e(x)$. A model trained to predict $P(S{=}1|x)$ is called a *non-traditional classifier* (NTC) (Elkan and Noto, 2008). It is trained by treating all unlabeled examples as belonging to the negative class. Hence, it predicts the probability that a positive example's label is observed and not the posterior class probability. Any model class can be used that predicts a probability.

In practice, the propensity scores are unknown. To enable learning, assumptions are made on the labeling mechanism and/or the data distribution. PU learning approaches can be differentiated based on whether they make the *Selected Completely At Random* (Kiryo et al., 2017; Liu et al., 2003; Mordelet and Vert, 2014; Zhao et al., 2022; Li et al., 2022; Chen et al., 2020a) or the *Selected At Random* assumption (Zhu et al., 2023; Bekker et al., 2019; Gong et al., 2021; Gerych et al., 2022; Kato et al., 2019).

The strongest assumption about the labeling mechanism is that the observed instances are *Selected Completely At Random (SCAR)* from the positive instances. In that case $e(x) = c$, where c is the labeling frequency (i.e., the percentage of the positive instances that we expect to be observed) (Elkan and Noto, 2008). Under SCAR, c can be recovered under mild assumptions about the data distribution (Jain et al., 2016; Ramaswamy et al., 2016; Bekker and Davis, 2018), and hence, so can the posterior class probabilities, using Equation 1. This work makes the more realistic assumption, *Selected At Random (SAR)*, where $e(x)$ is dependent on the feature values x.

3 NON-TRADITIONAL CLASSIFIERS (NTC) FOR BAYES-OPTIMAL PU LEARNING

Learning from PU data under the SAR assumption is non-trivial if the propensity scores are unknown. One has to either simultaneously learn the propensity scores and the class probability or make assumptions on the labeling mechanism and/or the data to justify training a NTC. We take the second approach and propose the novel *Function of Posterior (FOP)* assumption on the labeling mechanism that generalizes several existing assumptions. Importantly, the FOP assumption covers realistic scenarios that previous assumptions do not. Moreover, we prove that in combination with some mild assumptions on the data stated in Theorem 3.2, the Bayes optimal hyperplane is parallel to the hyperplane found by the NTC. Consequently, this justifies training an NTC under weaker

assumptions (FOP) than considered in previous literature. We address the problem that the NTC's hyperplane is in the wrong place by setting an appropriate threshold to convert the NTC's numeric score to a hard prediction. We propose a novel and efficient method τ`MIcalibration` that uses mutual information to set this threshold.

At a high-level, our approach, called `NTC-`τ`MI`, works as follows. First, we train an NTC from a PU dataset. Second, we use τ`MIcalibration` (see Subsection 3.3) to estimate a threshold τ, which transforms the NTC's score into a hard prediction.

3.1 Function Of Posterior (FOP) Assumption

We propose the novel Function of Posterior (FOP) assumption on the propensity scores:

Assumption 3.1 (Function of Posterior (FOP))**.** The propensity score is a function of the posterior class probability: $e(x) = g(P(Y{=}1|x))$, with $g : \mathbb{R} \to [0, 1]$ being any activation function.

This assumption is very general as it does not restrict the form of g apart from requiring its values to be in the range $[0, 1]$. It follows from Equation 1 that under the FOP assumption, the label probability is also a function h of the posterior class probability: $P(S{=}1|x){=}h(P(Y{=}1|x))$, with $h(t) = g(t)t$.

Our FOP generalizes three commonly used assumptions that constrain the form of the propensity scores $g(t)$. We now describe these assumptions in increasing order of restrictiveness. (1) Kato et al. (2019) introduced the Invariance of Order (IOO) assumption, which requires that $P(S{=}1|x)$ and $P(Y{=}1|x)$ induce the same order on the feature space. Thus, $g(t)$ is selected such that $h(t){=}g(t){\cdot}t$ is monotonically increasing. (2) The Probabilistic Gap (PG) (He et al., 2018) assumption requires that $e(x)$ is a monotonically increasing function of $P(Y{=}1|x)$, thus $g(t)$ should be monotonically increasing. In that case, $h(t)$ is also monotonically increasing, hence, training an NTC recovers the ordering induced by $P(Y{=}1|x)$. (3) Gerych et al. (2022) further restrict the PG assumption by requiring $e(x)$ to be a linear function of $P(Y{=}1|x)$, i.e., $g(t){=}k{\cdot}t$, with k a constant. If there exists an area of the feature space where $P(Y{=}1|x){=}1$, then the propensity scores become identifiable and one can recover $P(Y{=}1|x)$ through Equation 1.

Moreover, FOP may hold when other assumptions do not. For example, Figure 1 (c)[1] depicts a setting where $P(S{=}1|x)$ decreases in range $[2, 4]$, while $P(Y{=}1|x)$ increases in that same range, meaning that the IOO assumption is clearly violated whereas FOP holds. Such a setting is, for instance, common in active learning, where one would label instances close to the decision boundary (i.e., $P(Y{=}1|X){\approx}0.5$).

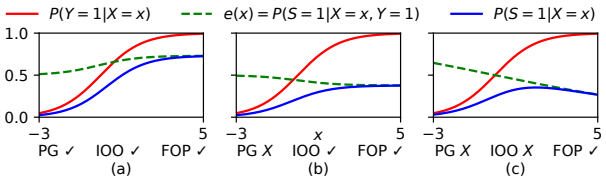


Figure 1: Example curves for posterior probability and propensity scores, where different assumptions hold.

3.2 Partial Optimal Hyperplane Recovery

The FOP assumption alone is not strong enough to enable learning. However, we show that combining this with assumptions on the data and a classifier with the right inductive bias enables learning from PU data that adheres to this assumption. To do so, we make a connection with the field of robust statistics which is concerned with modeling imperfect data (i.e., the assumptions of the model do not match the assumptions in the data). Learning an NTC does exactly this: we hope to gain insights about the true class distribution by learning a model from PU data.

An NTC predicts scores as $Z{=}\alpha_{PU}{+}X^T\beta_{PU}$, so that $\sigma(Z)$ predicts $P(S{=}1|x)$, with activation function σ. The NTC is learned by optimizing the following risk:

$$\alpha_{PU}, \beta_{PU} = \arg\min_{\alpha,\beta} \mathbb{E}_{(X,S)}[L(\alpha + X^T\beta, S)], \quad (2)$$

where L is a loss function. L and σ are defined by the model class used as the NTC. The following theorem shows that under FOP and some additional assumptions, β_{PU} is proportional to the parameter β^* of the optimal classifier. Consequently, learning the NTC allows recovering the direction of the Bayes optimal hyperplane (the proof is in Appendix A).

Theorem 3.2. *Given the following assumptions:*

(1) $P(Y{=}1|x){=}f(\alpha^*{+}x^T\beta^*)$*, with* $f : \mathbb{R}{\to}[0, 1]$ *an increasing activation function.*
(2) $e(x) = g(P(Y = 1|x))$ *with* $g : \mathbb{R}{\to}[0, 1]$ *any activation function. (FOP)*
(3) The NTC loss function $L(\cdot, S)$ *is convex w.r.t. its first argument.*
(4) The risk minimizer α_{PU}, β_{PU} *is uniquely defined.*
(5) The distribution of X *is elliptical.*[2,3]

[1] Appendix B explains how the curves are generated.

[2] I.e., its density function can be written as $p_X(x) = C \cdot r((x-\mu)^T\Sigma^{-1}(x-\mu))$, with C a normalizing constant, μ the mean (or median), Σ a semi-definite matrix and r a function of the probability density.

[3] Our experiments show that the approach is robust to

The property of scalability holds: $\beta_{PU}=\beta^*\cdot\gamma, \quad \gamma\in\mathbb{R}$.

Assumptions (1) and (2) limit Theorem 3.2 to linear classification settings, as both $P(Y=1|x)=f(t)$ and $P(S=1|x)=g(f(t))f(t)$ are assumed to be functions of $t=\alpha^*+x^T\beta^*$. However, our experiments show that our approach can be effectively extended to the case of non-linear classifiers such as deep neural networks. Assumption (3) is mild and is met for standard losses used in practice. Assumption (4) is usually met, e.g., for cross-entropy loss, the minimizer is unique, provided that the classes are not linearly separable, otherwise a regularization term is required. Assumption (5) applies to elliptical distributions, which are a broad generalization of Gaussian distributions. It will be violated for distributions whose density contours do not resemble ellipses. However, our experiments show that the approach is robust to violations of this assumption.

Ultimately, the goal is to use the NTC to make a prediction, which involves setting a threshold τ on the NTC's output:

$$\begin{cases} \alpha_{PU}+x^T\beta_{PU}\geq\tau & \implies \hat{Y}=1 \\ \alpha_{PU}+x^T\beta_{PU}<\tau & \implies \hat{Y}=0, \end{cases} \tag{3}$$

From Theorem 3.2 it is clear that, in a linear setting, the NTC-induced hyperplane shares the same direction as the optimal hyperplane. Consequently, one merely has to shift the NTC-induced hyperplane by a constant, which we accomplish by estimating the threshold τ.

3.3 Threshold Calibration using Mutual Information

A good threshold partitions the one-dimensional NTC score vector Z in a way that separates positive examples from negative ones. Intuitively, if the model is accurate, the Z values for positive examples should be close to each other but far away from the Z values for negative examples. That is, the positive and negative examples form two clusters that have low inter-class and high intra-class variability. The mutual information (MI) is a natural way to accomplish this. It has been used to this effect to select informative splits in decision trees (Shalev-Shwartz and Ben-David, 2013, chap. 18) and cluster analysis (Faivishevsky and Goldberger, 2010; Sugiyama et al., 2014).

Concretely, our goal is to find

$$\tau_* = \arg\max_{\tau} MI(\hat{Y}^\tau, Z), \tag{4}$$

where $\hat{Y}^\tau$ are predicted labels obtained by thresholding Z on τ. We need to calculate the MI between $\hat{Y}^\tau$, a discrete, and Z, a continuous distribution. We use the k-nearest neighbors approach proposed by Ross (2014) to estimate the MI. When computing the MI, we exclude the instances with an observed label to prevent selecting a threshold that simply separates labeled from unlabeled instances. See appendix D for more details.

We propose, `τMIcalibration`, an efficient way of finding the threshold that optimizes the MI, by reducing the computational complexity of Ross' method from $O(n^3)$ to worst case $O(n\log n+nmk)$ or more realistically $O(n\log n+mk^2)$. Here, n is the number of examples, and m and k are user-chosen hyperparameters for the number of candidate thresholds m and neighbors k, with $mk\ll n^2$. Ross' complexity stems from performing comparisons between all pairs of examples, for each possible threshold. Our insight is that (1) we can exploit the fact that Z is one-dimensional and hence, that the examples can be sorted, and (2) that examples that are not close to the threshold, only have same-label nearest neighbors. This removes the need for explicit comparisons between the majority of examples. Additionally, based on the same insights, we identify an upper and lower bound on the MI, solely based on the relative location of the threshold in the sorted Z vector. This allows us to identify a small range of values that must contain the optimal threshold and restricts the search to this range. The optimizations and full algorithm are detailed in Appendix C.

4 RELATED WORK

Research on SAR methods generally falls into one of two categories. The first category contains methods that estimate the propensity scores as part of the learning. For example, the SAR-EM (Bekker et al., 2019) and LBE (Gong et al., 2021) methods alternately optimize two models: one predicts class labels and the other propensity scores. Both make different assumptions on the form of the propensity scores. SAR-EM assumes that the propensity score only depends on a known subset of the features. LBE on the other hand assumes that both $P(Y=1|x)$ and $e(x)$ are logistic functions over a linear combination of the features. Because these approaches require simultaneously training two models via expectation maximization, they are computationally expensive and have a large number of parameters, potentially leading to a high variance in model performance. Moreover, the propensity scores are in general not identifiable (Gerych et al., 2022), so there are no theoretical guarantees on recovering the true classifier.

This paper belongs to the second category of SAR approaches, which impose assumptions on the labeling

violations of this assumption.

mechanism so that the NTC can be used. Gerych et al. (2022) assumes the strong and unrealistic linear Probabilistic Gap assumption in combination with the Positive Function assumption.[4] This makes the propensity scores identifiable and efficient to estimate. They find $P(Y{=}1|x)$ by scaling the NTC output with $e(x)$ as in Equation 1. PUe (Wang et al., 2023) also makes the linear Probabilistic Gap assumption and additionally requires the class prior. It first estimates propensity scores up to a multiplicative factor using a regularized NTC and next trains a classifier using propensity weighting (Bekker et al., 2019) and class balancing to match the class prior.

The closest method to our approach is PUSB (Kato et al., 2019).[5] This approach makes the Invariance of Order assumption and requires the class prior π. It works by training an NTC and sets the threshold τ on the NTC's output such that a fraction π of the training set is predicted as positive. Setting the threshold based on π has several significant drawbacks. First, this only leads to the Bayes optimal classifier when the dataset is balanced, i.e., when $\pi = 0.5$ (Elkan, 2001).[6] Second, because π is often not known, it must be estimated from data. Approaches for estimating π require training an additional model and either make the more restrictive SCAR assumption (Jain et al., 2016; Ramaswamy et al., 2016; Bekker and Davis, 2018) which negates their guarantees (i.e., their estimates can be off) or require estimating propensity scores (Jain et al., 2020) and suffer the aforementioned limitations associated with this task (e.g., computational cost). We make two major contributions over PUSB: 1) we propose the novel FOP assumption that greatly expands the applicability of NTC approaches for SAR data, and 2) our τ`MIcalibration` approach is more computationally efficient and more practical as it requires no information beyond the NTC's predictions.

In contrast to existing approaches for SAR data, our approach offers the advantages that it does not require (1) estimating the propensity scores, which is computationally and theoretically challenging, (2) making overly restrictive assumptions such as that the propensity score is a linear function or (3) having domain knowledge about the class prior. Moreover, it takes advantage of existing classification methods and is model-agnostic.

[4]Under the *Positive Function* assumption, there exists a region in the input space where $P(Y{=}1|x){=}1$.

[5]It was designed for the case-control setting. However, it can be converted to a single-training set scenario.

[6]E.g.: Consider a dataset consisting of two equal-sized regions A and B with $|X_{x\in A}|{=}|X_{x\in B}|$, $P(Y{=}1|X{\in}A){=}0.7$ and $P(Y{=}1|X{\in}B){=}0.1$. Here, the dataset is unbalanced with $\pi{=}0.4$. Then, the top fraction π examples doesn't cover all instances in A.

Finally, Platek and Mielniczuk (2023) made use of robust statistics theory (Li and Duan, 1989) in the more restrictive SCAR scenario using slightly different assumptions (e.g., they only consider logistic loss). Our insight is that our novel FOP assumption makes this theory applicable to the more realistic SAR setting.

5 EXPERIMENTS

Our experiments address the following questions:

Q1 How does `NTC-`τ`MI` compare to state-of-the-art SAR methods under different labeling mechanisms, in predictive performance and speed?

Q2 How does `NTC-`τ`MI` perform when using a deep neural network instead of a linear classifier as assumed by Theorem 3.2?

Q3 How robust is `NTC-`τ`MI` to violating the ellipticity assumption of the feature distribution?

Q4 Does τ`MIcalibration` or PUSB's approach to set the threshold get closer to the optimal threshold that maximizes the balanced accuracy?

5.1 Experimental setup

Methods. We compare `NTC-`τ`MI` against five methods discussed in the related work that make different assumptions on the labeling mechanism and/or data distribution: SAR-EM (Bekker et al., 2019), PUe (Wang et al., 2023), LBE (Gong et al., 2021), PGlin (Gerych et al., 2022) and PUSB (Kato et al., 2019). All methods, except PUe and LBE, were implemented using their original code. Due to code unavailability, we implemented PUe and LBE as described in their respective papers. For PUSB and PUe, we set the threshold by using the KM2 class prior estimator (Ramaswamy et al., 2016) which is recommended by the authors and used in their experiments (Kato et al., 2019). The other three methods estimate $P(Y{=}1|x)$ and we set the threshold at 0.5, i.e., if $P(Y{=}1|x) \geq 0.5$ then the model predicts positive and otherwise it predicts negative. The hyperparameters of `NTC-`τ`MI` are set to $k{=}3$ and $m{=}100$. The former is the optimal one from Ross (2014) and the latter is selected as a trade-off between training time and predictive performance. Additionally, we compare the methods to learning from fully labeled data, which is termed the oracle method. One can interpret this as the upper limit on performance.

Data. We compare the methods on 20 UCI datasets (Kelly et al., 2023) and four image datasets: see Table 1 for their characteristics. The UCI datasets were chosen to vary widely in size, number of features and

class separability. Multi-class UCI datasets were converted into a binary classification dataset by making the most common class the positive class and merging all other classes into the negative class. The image datasets are MNIST, USPS, FashionMNIST and CIFAR10. For MNIST, the even (odd) numbers form the positive (negative) class. In USPS, digits < 5 are positives and ≥ 5 are negatives. In FashionMNIST, the clothing items for the upper body are the positives and anything else is negative. In CIFAR10, all vehicles form the positive class and animals form the negative class.

Table 1: Summary statistics of the considered datasets. The first 20 datasets come from the UCI Machine Learning Repository. The last 4 datasets are image datasets. n and d denote the cardinality and dimensionality of the datasets, respectively.

Dataset	n	d	π
Abalone	4177	8	0.16
Banknote	1372	4	0.44
Breast-w	699	9	0.34
Diabetes	768	8	0.35
Haberman	306	3	0.26
Heart	270	13	0.44
Ionosphere	351	34	0.64
Isolet	7797	617	0.04
Jm1	10885	21	0.19
Kc1	2109	21	0.15
Madelon	2600	500	0.50
Musk	6598	166	0.15
Segment	2310	19	0.14
Semeion	1593	256	0.10
Sonar	208	60	0.53
Spambase	4601	57	0.39
Vehicle	846	18	0.26
Waveform	5000	40	0.34
Wdbc	569	30	0.37
Yeast	1484	8	0.31
USPS	9298	16×16	0.58
MNIST	70 000	28×28	0.49
FashionMNIST	70 000	28×28	0.50
CIFAR10	60 000	$32 \times 32 \times 3$	0.40

Constructing PU data. Since the considered datasets are fully labeled, we take the standard approach and convert them to PU data (Gong et al., 2021; Kato et al., 2019; Gerych et al., 2022). Specifically, we consider four different labeling mechanisms that range from strict to general:

S1. (*SCAR*) The propensity score is constant:

$e(x) = 0.1$.

S2. (*PG lin*) The propensity score is a linear function of the posterior:

$e(x) = 0.1 \cdot P(Y{=}1|x)$.

S3. (*IOO*) The propensity score is a decreasing function of the posterior:

$e(x) = f_{\text{sigmoid}}(-0.5 \cdot P(Y{=}1|x) - 1.5)$.

S4. (*FOP*) The propensity score is a decreasing function of the true signal β^*:

$e(x) = 0.5 \cdot f_{\text{sigmoid}}(-0.5 \cdot x^T \beta^*)$

In S2, S3, and S4, the posterior $P(Y{=}1|x)$ and true signal β^* are estimated using the oracle method.

Model. For the *UCI datasets*, all methods use logistic regression as the predictive model. For SAR-EM and LBE, we also use logistic regression to predict the propensity scores. We make this choice because (1) the LBE, PUSB, and SAR-EM papers all used this model on UCI datasets, and (2) it satisfies the assumptions of Theorem 3.2. We use the scikit-learn (Pedregosa et al., 2011) implementation with ℓ_2 regularization and its default hyperparameter values.

For the *image datasets* we employ deep neural networks and implement the architectures used in prior PU work (Chen et al., 2020b; Kiryo et al., 2017; Kato et al., 2019; Chen et al., 2020a); see Appendix E for the specific architectures. Optimization is done using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-5}. PUSB, SAR-EM and LBE use the loss functions mentioned in their respective papers and the NTC uses cross-entropy as its loss function. We train the models using a batch size of 512 for a max of 100 epochs using the loss on a validation set for early stopping.

Computing resources. For the *UCI* datasets, all experiments are executed in an isolated manner on the same machine with access to 128Gb RAM and 1 core of an Intel Xeon CPU E5-2640 v3.

For the *image* datasets, the experiments are executed on a machine with access to 256Gb RAM, one NVIDIA RTX A5000 GPU and an AMD EPYC 7502 32-Core CPU. Training the neural networks is done on the GPU, whereas `tMIcalibration` and class prior estimation is executed by the CPU.

Setup. Our evaluation employs the following methodology. For *UCI datasets*, the datasets are first randomly partitioned into a training (75%) and test set (25%). Second, we apply one of four labeling mechanisms to the training data. We then apply the different learning methods on this newly acquired PU data. We repeat this for 20 different train-test splits.

For the *image datasets*, we follow the predefined training-test partitions provided in the PyTorch library (Paszke et al., 2019). We randomly partition the training data into a training (80%) and validation (20%) set, repeated for 20 different splits. Then, we employ the same procedure as for the UCI datasets: apply a labeling mechanism to both the training and validation sets, fit the different models for 100 epochs (using the loss on the validation set for early stopping).

Balanced accuracy is used as evaluation criteria because it is sensitive to the chosen threshold, even for imbalanced classes;[7] the threshold selecting approach is a key difference among the considered methods. In addition, the supplement shows results for F1.

5.2 Results

Results for Q1. Table 2 shows the number of times (out of 20) that NTC-τMI's average balanced accuracy is higher (Win), within a margin of 0.01 (Draw) or lower (Loss) than that of the baselines for each of the four considered labeling mechanisms on the UCI datasets. Tables 6-9 in the supplement contain a detailed breakdown of all methods' performance. Regardless of the labeling mechanism, NTC-τMI offers significantly superior performance over SAR-EM, PUe (79 wins, 1 draws, 0 losses) and PGLin (74 wins, 5 draws, 1 loss). In the majority of cases, it outperforms or ties LBE (55 wins, 8 draws and 17 losses) and PUSB (59 wins, 10 draws, 11 losses). Across all datasets and labeling mechanisms, NTC-τMI results in an average relative increase in balanced accuracy of 7.4%, 17.6%, 19.5%, 36.2% and 50.5% over LBE, PUSB, PGLin, SAR-EM and PUe respectively. Tables 5 and 10 - 13 in the Supplement show similar results when comparing the F1 score instead.

Figure 2 visualizes the performance of NTC-τMI compared to its two closest competitors LBE and PUSB for all labeling mechanisms. Each point represents the balanced accuracy of a competitor on the x-axis and NTC-τMI on the y-axis for a specific dataset. Being farther from the diagonal indicates a larger difference in performance. Generally, when NTC-τMI is better it offers larger wins whereas in the few cases where it loses it performs similarly to its competitor.

Table 3 shows the median training time (in seconds) required to train the different methods on the UCI datasets and the median speed-up in training time of NTC-τMI over each baseline.[8] Our method takes on median less than a second to train, which is similar to PGLin. Contrarily, the training times of SAR-EM, LBE, PUe and PUSB are longer and seem to be highly affected by the dimensionality of the dataset. For instance, SAR-EM, PUe and LBE take long to train on Isolet, which has 617 dimensions. PUSB and PUe are also affected by the cardinality of the dataset. For example, on Jm1, which counts 10885 instances, they have on median their longest training times.

Consequently, NTC-τMI is on median 5947 times faster than PUSB and 291 times faster than LBE, which are its nearest competitors in terms of predictive performance. Moreover, it is on median 249 and 4509 times faster than respectively SAR-EM and PUe. The speedup versus PUSB and PUe arises because τMIcalibration is far more computationally efficient than estimating the class prior from data. The added computational cost for LBE and SAR-EM arises from the expectation maximization process to learn two models simultaneously. Our training time is similar to that of PGLin. However, PGLin offers substantially worse predictive performance than our approach because it makes much stronger assumptions.

In summary, our proposed method NTC-τMI offers better or similar predictive performance while being orders of magnitude faster than its nearest competitors.

Results for Q2 Empirically, we evaluate how robust NTC-τMI is to violating Theorem 3.2's assumption of a linear classifier. Consequently, all methods employ a deep neural network as their base model. Table 2 shows the number of wins, losses and draws of our NTC-τMI versus each baseline on the four image datasets; see Table 16 in the appendix for detailed results. Generally, NTC-τMI outperforms most competitors. An exception is PUSB, which offers similar performance over the four labeling mechanisms. However, NTC-τMI always outperforms PUSB on the CIFAR10 dataset, arguably the hardest of the four datasets.

Results for Q3 We use synthetic data to evaluate how robust our method is to violations of the ellipticity assumption. We generate three 10 dimensional artificial datasets with 2000 instances (50% for training and 50% for testing). To generate the feature vectors, each dataset considers a different distribution: a multivariate normal distribution, a uniform distribution and a lognormal distribution, which is a right-skewed distribution. Among the three, only the first is elliptic, and thus the only one for which assumption (5) of Theorem 3.2 is met. For each dataset, the true class variable Y was generated with posterior probability $P(Y{=}1|x){=}f_{\text{sigmoid}}(x^T\beta^*)$, where $\beta^*{=}(1,\ldots,1)$ which ensures that the FOP assumption holds.

Figure 3 shows the distributions of the balanced accu-

[7] The unweighted accuracy's value is mainly determined by whether examples belonging to the majority class are correctly classified.

[8] Table 14 and 15 in the Appendix show how these training times and speedups are distributed.

Table 2: Number of wins (W), losses (L), and draws (D) (absolute value difference in average balanced accuracy ≤ 0.01) in terms of balanced accuracy for NTC-τMI versus each competitor on 20 *UCI* and 4 *image* datasets. Results are shown for each of the four labeling mechanisms.

	UCI															*image*													
	SAR-EM			PUe			LBE			PGlin			PUSB			SAR-EM			PUe			LBE			PGlin			PUSB		
	W	L	D	W	L	D	W	L	D	W	L	D	W	L	D	W	L	D	W	L	D	W	L	D	W	L	D	W	L	D
S1	**20**	0	0	**20**	0	0	**14**	5	1	**20**	0	0	**15**	3	2	**4**	0	1	**4**	0	0	**4**	0	0	**4**	0	0	1	**2**	1
S2	**19**	0	1	**20**	0	0	**14**	3	3	**18**	0	2	**16**	2	2	**4**	0	0	**4**	0	0	**4**	0	0	**4**	0	0	**2**	1	1
S3	**20**	0	0	**20**	0	0	**14**	5	1	**19**	0	1	**13**	4	3	**4**	0	0	**4**	0	0	**4**	0	0	**4**	0	0	1	**2**	1
S4	**20**	0	0	**19**	0	1	**13**	4	3	**17**	1	2	**15**	2	3	**2**	**2**	0	**4**	0	0	**4**	0	0	**4**	0	0	1	**3**	0

Table 3: The median training time (in seconds) of the different methods and the median speedup ($\times$) of NTC-τMI compared to every method per dataset. The results are calculated over the four labeling mechanisms.

	Training time (seconds)						Speedup ($\times$)				
L.S.	SAR-EM	PUe	LBE	PGlin	PUSB	NTC-τMI	SAR-EM	PUe	LBE	PGlin	PUSB
Abalone	11.97	1375.64	62.39	0.02	1910.46	0.21	56.92	5631.86	223.36	0.10	9147.91
Banknote	2.61	68.04	46.63	0.01	333.08	0.21	11.47	324.65	216.08	0.05	1832.51
Breast-w	3.52	13.01	43.53	0.02	218.73	0.23	14.07	51.06	177.38	0.06	829.05
Diabetes	5.13	17.45	43.92	0.02	263.80	0.24	21.36	73.01	183.20	0.07	1095.27
Haberman	1.66	1.02	38.82	0.01	51.16	0.18	9.07	5.28	201.37	0.05	292.32
Heart	2.45	0.94	40.22	0.01	54.71	0.20	11.51	4.05	170.60	0.07	266.62
Ionosphere	6.17	1.27	40.64	0.02	82.31	0.28	21.53	5.64	145.12	0.08	372.94
Isolet	392.35	6319.62	2821.20	4.42	7189.68	5.17	68.58	1042.91	490.50	0.91	1206.07
Jm1	519.39	32177.25	162.93	0.24	33476.98	0.44	1204.54	64874.62	349.24	0.50	69219.96
Kc1	24.68	275.31	56.28	0.04	1088.48	0.30	84.75	912.67	183.29	0.15	3496.29
Madelon	641.01	178.73	701.83	1.28	637.23	1.52	374.29	100.46	397.39	0.82	403.71
Musk	526.90	4779.14	671.10	0.71	5607.58	0.92	779.80	3711.28	614.40	0.80	4867.86
Segment	24.68	246.03	57.01	0.03	656.61	0.21	93.85	1034.64	218.83	0.12	2959.14
Semeion	6.69	62.92	221.31	0.08	361.97	0.28	15.60	187.12	631.33	0.30	1035.85
Sonar	4.14	0.50	38.52	0.01	27.75	0.18	22.39	2.78	205.40	0.08	145.19
Spambase	561.52	2111.36	201.16	0.16	3428.01	0.42	1400.46	4901.32	448.25	0.37	7858.95
Vehicle	5.70	14.26	45.42	0.02	200.36	0.20	29.19	68.08	172.13	0.11	947.67
Waveform	166.84	2167.54	116.68	0.07	2787.59	0.26	720.10	6943.93	416.06	0.22	10272.45
Wdbc	4.05	6.38	45.18	0.03	227.84	0.29	17.63	24.00	156.99	0.12	782.11
Yeast	6.51	63.96	47.59	0.02	364.50	0.22	29.11	288.31	215.39	0.08	1923.50
Average	145.90	2494.02	275.12	0.36	2948.44	0.60	249.31	4509.38	290.82	0.25	5947.77

racy when generating 20 different datasets under labeling mechanism S1.[9] NTC-τMI performs similar to the oracle method (i.e., the upper limit) and better than the baselines for the normal and uniform distributions. When the features follow the skewed lognormal distribution, the performance of all methods declines, including the oracle method. This indicates that this deviation makes the problem more difficult, regardless of the method. That is, the violation of the assumption is not the sole reason for the decrease in NTC-τMI's performance. However, our method still achieves significantly higher accuracy than the competitors.

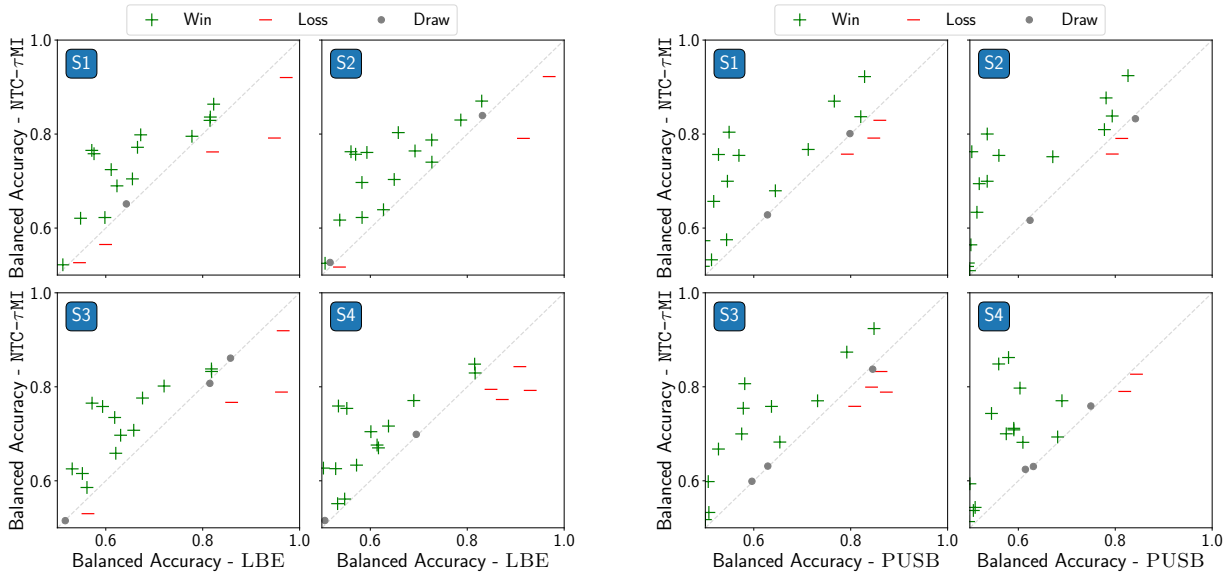
Results for Q4. We compare using our τMIcalibration to PUSB's approach for selecting the threshold using the class prior as estimated by KM2 (Ramaswamy et al., 2016). We compute each approach's relative error to the optimal threshold: $(\text{Bacc}_{\text{optimal}} - \text{Bacc}_{\text{method}})/\text{Bacc}_{\text{optimal}}$, where $\text{Bacc}_{\text{optimal}}$ is the threshold found by maximizing the balanced accuracy on a fully labeled training set (i.e., the idealized case). Table 4 reports the 25% quartile, median and 75% quartile of the relative error on the UCI datasets for each labeling mechanism.[10] τMIcalibration achieves a lower relative error than the baseline in all settings.

6 CONCLUSION

We proposed a new assumption on the labeling mechanism called Function Of Posterior. We show that this assumption, in conjunction with some model and data assumptions, enables a theoretically justified approach

[9] See Figure 6 in Supplement for results under S2-S4.

[10] Tables 17-20 in Supplement contain detailed results.



(a) NTC-τMI vs LBE (b) NTC-τMI vs PUSB

Figure 2: Each point shows the balanced accuracy on one UCI dataset of (a) LBE and (b) PUSB vs. NTC-τMI. Points above the diagonal mean that NTC-τMI outperforms the competitor on that dataset.

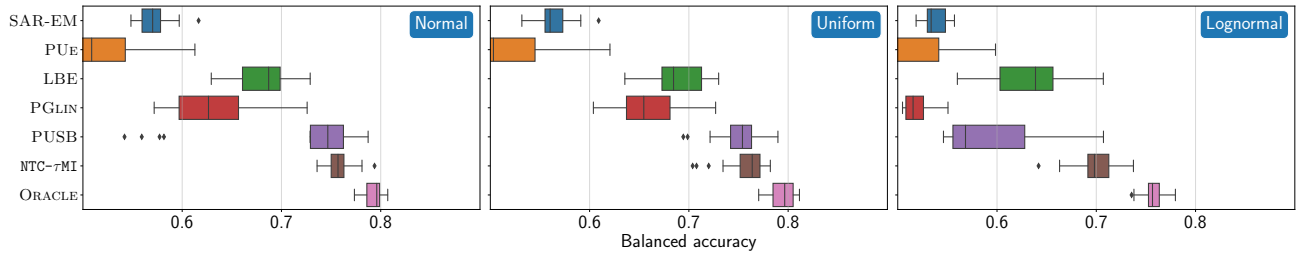


Figure 3: Balanced accuracy for artificial datasets using various feature distributions under S1.

Table 4: The 25%, median and 75% quartile of the relative error for the balanced accuracy when using τMIcalibration or PUSB's π-based approach to threshold the NTC's output. The correct threshold is obtained using fully labeled data. The values are computed over all UCI datasets and labeling mechanisms.

	τMIcalibration			Estimated π		
L.S.	25%	Median	75%	25%	Median	75%
S1	0.011	0.034	0.072	0.031	0.091	0.137
S2	0.013	0.040	0.071	0.059	0.104	0.162
S3	0.014	0.027	0.088	0.023	0.075	0.098
S4	0.013	0.027	0.078	0.027	0.071	0.131

to learning in a SAR PU setting. A key step in our approach is setting the threshold on a model's numeric output and we propose a novel approach based on mutual information to estimate the threshold from PU data. Our experiments showed that our method offers superior predictive performance while being orders of magnitude faster than state-of-the-art PU methods. Moreover, empirically, our approach is robust to violations of assumptions of the theoretical results (e.g., requiring a linear model). Intuitively, a reason that the approach may still work for non-linear classifiers such as (deep) neural networks is because, essentially, every layer in a neural network is a linear combination of the previous layer's output. Consequently, Theorem 3.2 could be applied locally to every layer. An important direction for future work is to formally prove this.

Acknowledgements

This research is supported by TAILOR Connectivity Fund, the European Union's Horizon 2020 research and innovation programme under GA No 952215 (PT), FWO-Vlaanderen G0D8819N (JD & TM), the Flemish government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme (JD & JB) and iBOF/21/075 (JD & TM).

References

U. Akujuobi, J. Chen, M. Elhoseiny, M. Spranger, and X. Zhang. Temporal positive-unlabeled learning for biomedical hypothesis generation via risk estimation. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 4597–4609, 2020.

J. Bekker and J. Davis. Estimating the class prior in positive and unlabeled data through decision tree induction. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 2712–2719, 2018.

J. Bekker and J. Davis. Learning from positive and unlabeled data: a survey. *Machine Learning*, 109: 719–760, 2020.

J. Bekker, P. Robberechts, and J. Davis. Beyond the selected completely at random assumption for learning from positive and unlabeled data. In *Proceedings of the 2019 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 71–85, 2019.

S. Cambanis, S. Huang, and G. Simons. On the theory of elliptically contoured distributions. *Journal of Multivariate Analysis*, 11(3):368–385, 1981.

H. Chen, F. Liu, Y. Wang, L. Zhao, and H. Wu. A variational approach for learning from positive and unlabeled data. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 14844–14854, 2020a.

X. Chen, W. Chen, T. Chen, Y. Yuan, C. Gong, K. Chen, and Z. Wang. Self-PU: Self boosted and calibrated positive-unlabeled training. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1510–1519, 2020b.

F. Chiaroni, M-C. Rahal, N. Hueber, and F. Dufaux. Learning with a generative adversarial network from a positive unlabeled dataset for image classification. In *Proceedings of the 25th IEEE International Conference on Image Processing*, pages 1368–1372, 2018.

C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 213–220, 2008.

L. Faivishevsky and J. Goldberger. A nonparametric information theoretic clustering algorithm. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, page 351–358, 2010.

G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu. Text classification without negative examples revisit. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):6–20, 2006.

W. Gerych, T. Hartvigsen, L. Buquicchio, E. Agu, and E. Rundensteiner. Recovering the propensity score from biased positive unlabeled data. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pages 6694–6702, 2022.

C. Gong, Q. Wang, T. Liu, B. Han, J. You, J. Yang, and D. Tao. Instance-dependent positive and unlabeled learning with labeling bias estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):1–16, 2021.

P. N. Hameed, K. Verspoor, S. Kusljic, and S. K. Halgamuge. Positive-unlabeled learning for inferring drug interactions based on heterogeneous attributes. *BMC Bioinformatics*, 18(1):1–15, 2017.

F. He, T. Liu, J. Webb, and D. Tao. Instance-dependent PU learning by Bayesian optimal relabeling. *arXiv preprint*, 2018. URL `https://arxiv.org/abs/1808.02180`.

S. Jain, M. White, and P. Radivojac. Estimating the class prior and posterior from noisy positives and unlabeled data. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, page 2693–2701, 2016.

S. Jain, J. Delano, H. Sharma, and P. Radivojac. Class prior estimation with biased positives and unlabeled examples. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 4255–4263, 2020.

K. Jaskie, J. Martin, and A. Spanias. Pv fault detection using positive unlabeled learning. *Applied Sciences*, 11(12), 2021.

M. Kato, T. Teshima, and J. Honda. Learning from positive and unlabeled data with a selection bias. In *Proceedings of the 7th International Conference on Learning Representations*, pages 1–12, 2019.

M. Kelly, R. Longjohn, and K. Nottingham. UCI Machine Learning Repository, 2023. URL `http://archive.ics.uci.edu`.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015. URL `https://arxiv.org/abs/1412.6980`.

R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1674–1684, 2017.

C. Li, X. Li, L. Feng, and J. Ouyang. Who is your right mixup partner in positive and unlabeled learning. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

F. Li, S. Dong, A. Leier, M. Han, X. Guo, J. Xu, X. Wang, S. Pan, C. Jia, Y. Zhang, G. Webb, L. J. M. Coin, C. Li, and J. Song. Positive-unlabeled learning in bioinformatics and computational biology: a brief review. *Briefings in Bioinformatics*, 23 (1), 2021.

K. Li and N. Duan. Regression analysis under link violation. *The Annals of Statistics*, 17(3):1009–1052, 1989.

X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, page 587–592, 2003.

B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 179–186, 2003.

F. Mordelet and J-P. Vert. A bagging SVM to learn from positive and unlabeled examples. *Pattern Recognition Letters*, 37:201–209, 2014.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

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.

M. Platek and J. Mielniczuk. Enhancing naive classifier for positive unlabeled data based on logistic regression approach. In *Proceedings of the 18th Conference on Computer Science and Intelligence Systems*, pages 225–233, 2023.

H. Ramaswamy, C. Scott, and A. Tewari. Mixture proportion estimation via kernel embeddings of distributions. In *Proceedings of the 33rd International Conference on Machine Learning*, page 2052–2060, 2016.

B. C. Ross. Mutual information between discrete and continuous data sets. *PLOS One*, 9(2):1–9, 2014.

K. Sechidis and G. Brown. Simple strategies for semi-supervised feature selection. *Machine Learning*, 107 (2):357–395, 2018.

K. Sechidis, B. Calvo, and G. Brown. Statistical hypothesis testing in positive unlabelled data. In *Proceedings of the 2014 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 66–81, 2014.

K. Sechidis, M. Sperrin, E. S. Petherick, M. Luján, and G. Brown. Dealing with under-reported variables: An information theoretic solution. *International Journal of Approximate Reasoning*, 85:159 – 177, 2017.

S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2013.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

M. Sugiyama, G. Niu, M. Yamada, M. Kimura, and H. Hachiya. Information-maximization clustering based on squared-loss mutual information. *Neural Compututation*, 26(1):84–131, 2014.

X. Wang, H. Chen, T. Guo, and Y. Wang. PUe: Biased positive-unlabeled learning enhancement by causal inference. In *Proceedings of the 37th Conference on Neural Information Processing Systems*, pages 1–12, 2023.

Y. Zhao, Q. Xu, Y. Jiang, P. Wen, and Q. Huang. Dist-PU: Positive-unlabeled learning from a label distribution perspective. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14461–14470, 2022.

Z. Zhu, L. Wang, P. Zhao, C. Du, W. Zhang, H. Dong, B. Qiao, Q. Lin, S. Rajmohan, and D. Zhang. Robust positive-unlabeled learning via noise negative sample self-correction. In *Proceedings of the 29th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 3663–3673, 2023.

Checklist

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] Our paper clearly describes the assumptions made for our algorithm to work.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] Section 3.3 overviews the complexity of our algorithm.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] Source code for our algorithm and to reproduce the experiments will be provided as supplementary material.
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] Theorem 3.2 clearly states the full set of assumptions.
   (b) Complete proofs of all theoretical results. [Yes] A proof for Theorem 3.2 is available in the supplement.
   (c) Clear explanations of any assumptions. [Yes] We believe that we have been complete in explaining the assumptions.
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] The code and data will be made available as supplementary material.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] Our experimental setup overviews the training details.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] We provide a clear explanation for the metrics and statistics used.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] We give an overview of the computing resources required in our experimental setup.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] Our code refers to the source of the applied algorithms.
   (b) The license information of the assets, if applicable. [Yes] We include the license of the existing assets in our supplementary material.
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes] The supplementary material will contain all new assets (code).
   (d) Information about consent from data providers/curators. [Yes] The datasets that we use are generally open source.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable] No private information is being processed for our work.
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Learning from biased positive-unlabeled data via threshold calibration: Supplementary Materials

A Proof of Theorem 3.2

Proof. First, note $P(S{=}1|x) = h(\alpha^* + x^T\beta^*)$, where $h(t) = g(f(t))\cdot f(t)$, which follows from assumptions (1) and (2) of Theorem 3.2 and equation 1. Observe that variable S has the same conditional distribution as

$$r(\alpha^* + X^T\beta^*, U) := I(h(\alpha^* + X^T\beta^*) > U),$$

where $I(\cdot)$ is the indicator function and U is a random variable generated from uniform distribution on $[0, 1]$ and is independent from X. This follows from

$$\begin{aligned} P[r(\alpha^* + x^T\beta^*, U) = 1|X = x] &= P[U < h(x^T\beta^* + \alpha^*)] \\ &\overset{\text{(i)}}{=} h(x^T\beta^* + \alpha^*) = P(S = 1|X = x). \end{aligned}$$

Equality (i) uses the property of uniform distribution $P(U < t) = t$. Next, for any α, β, the risk $R(\alpha, \beta)$ can be bounded from below as

$$\begin{aligned} &R(\alpha, \beta) = \mathbb{E}L(\alpha + X^T\beta, S) \\ &\overset{\text{(i)}}{=} \mathbb{E}[L(\alpha + X^T\beta, r(\alpha^* + X^T\beta^*, U))] \\ &\overset{\text{(ii)}}{=} \mathbb{E}\left\{\mathbb{E}[L(\alpha + X^T\beta, r(\alpha^* + X^T\beta^*, U))|X^T\beta^*, U]\right\} \\ &\overset{\text{(iii)}}{\geq} \mathbb{E}\left\{L(\mathbb{E}[\alpha + X^T\beta|X^T\beta^*, U], r(\alpha^* + X^T\beta^*, U))\right\} \\ &\overset{\text{(iv)}}{=} \mathbb{E}\left\{L(\alpha + \mathbb{E}[X^T\beta|X^T\beta^*], S)\right\} \\ &\overset{\text{(v)}}{=} \mathbb{E}\left\{L(\alpha + a + \gamma X^T\beta^*, S)\right\} = R(\alpha + a, \beta^*\gamma). \end{aligned}$$

Equality (i) uses the fact that S has the same conditional distribution as $r(\alpha^* + X^T\beta^*, U)$. Equality (ii) uses the Law of Total Expectation $\mathbb{E}(A) = \mathbb{E}[\mathbb{E}(A|B)]$ to condition on U and $X^T\beta^*$, so that $r(\alpha^* + X^T\beta^*, U)$ becomes a constant. Inequality (iii) uses the Jensen inequality, $r(\alpha^*+X^T\beta^*, U)$ being constant, and the fact that L is convex with respect to its first argument (i.e., assumption (3)). Equality (iv) substitutes $r(\alpha^* + X^T\beta^*, U)$ for S similar to (i) and because $X^T\beta$ and U are independent, we can drop U from the conditional inside the inner expected value. Finally, equality (v) follows from the fact that for elliptical distributions, $\mathbb{E}[X^T\beta|X^T\beta^*] = a + \gamma X^T\beta^*$, for some scalars a and γ (Cambanis et al., 1981). □

B Generating Figure 1

The plots showing $P(Y=1|X=x)$, $e(x)=P(S=1|Y=1,X=x)$ and $P(S=1|X=x)$ were generated as follows. Variable x was generated from a uniform distribution on $[-3,5]$. Probability $P(Y=1|X=x)=f_{\text{sigmoid}}(x)$ and the propensity score function was:

1. Figure 1 (a): $e(x)=f_{\text{sigmoid}}(P(Y=1|X=x))$
2. Figure 1 (b): $e(x)=f_{\text{sigmoid}}(-0.5\cdot P(Y=1|X=x))$
3. Figure 1 (c): $e(x)=f_{\text{sigmoid}}(-0.2x)=f_{\text{sigmoid}}(-0.2f^{-1}_{\text{sigmoid}}(P(Y=1|X=x)))$

The FOP assumption holds in all three cases. The probability for S is found using the formula:

$$P(S=1|X=x)=e(x)P(Y=1|X=x).$$

C Efficient MI-based threshold calibration

This section details the τ`MIcalibration` method for finding the threshold on Z that optimizes the mutual information (MI). Here, Z is the 1-dimensional score vector computed by the NTC: $Z{=}\alpha_{PU}{+}X^T\beta_{PU}$. Specifically, we introduce four optimizations to compute the threshold more efficiently.

The goal is to find threshold τ_* such that

$$\tau_*=\arg\max_{\tau} MI(\hat{Y}^\tau,Z) \tag{5}$$

where $\hat{Y}^\tau$ are predicted labels obtained by thresholding Z on τ, i.e., $\hat{Y}^\tau{=}1$ if $Z{\geq}\tau$ else 0. We need to calculate the MI between $\hat{Y}^\tau$, a discrete, and Z, a continuous distribution. To do this, we use Ross' (2014) k-nearest neighbours method, that estimates the MI for a threshold τ from vectors $\mathbf{z}$ and $\hat{\mathbf{y}}^\tau=\mathbf{z}\geq\tau$ of size n as:

$$\begin{aligned}\widehat{MI}(\hat{\mathbf{y}}^\tau,\mathbf{z}) =&\psi(n)+\psi(k) && -\langle\psi\left(|\hat{\mathbf{y}}^\tau{=}y_i^\tau|\right)\rangle && -\langle\psi\left(r_i\right)\rangle \\ =&\psi(n)+\psi(k) && -\frac{1}{n}\sum_{i=0}^{n}\psi\left(\sum_{j=0}^{n}\mathbb{1}[y_i^\tau=y_j^\tau]\right) && -\frac{1}{n}\sum_{i=0}^{n}\psi\left(r_i\right),\end{aligned} \tag{6}$$

where ψ is the digamma function, $\langle\cdot\rangle$ is the average over n examples, $|\hat{\mathbf{y}}^\tau{=}y_i^\tau|$ is the number of instances in $\hat{\mathbf{y}}^\tau$ with the same label as the i-th instance, and r_i is the number of examples in $\mathbf{z}$ whose Z value is at least as close to z_i as z_i's k^{th} nearest same-label neighbor. The first two terms are independent of τ and can be ignored when comparing thresholds.

C.1 Optimizing the threshold calibration

Naively searching for τ_* entails computing Equation 6 for every possible threshold, i.e., all possible $n-1$ splits. This results in a complexity of $O(n^3)$, due to all pairwise comparisons for all possible thresholds. However, because $\mathbf{z}$ is one-dimensional, we can sort the array to make the computations more efficient. We also observe that for the computation, the exact threshold value τ is not required, only the fraction of examples $\gamma_\tau=|\mathbf{z}<\tau|/n$ that are labeled as negative for the considered threshold.

Optimization 1: $\langle\psi\left(|\hat{\mathbf{y}}^\tau{=}y_i^\tau|\right)\rangle$

The number of zeros and ones in $\hat{\mathbf{y}}^\tau$ is determined by γ_τ and n: $\gamma_\tau n$ zeros and $(1-\gamma_\tau)n$ ones. Therefore, the average number of instances in $\hat{\mathbf{y}}^\tau$ with the same label, can be computed without iterating over $\hat{\mathbf{y}}^\tau$ as:

$$\langle\psi\left(|\hat{\mathbf{y}}^\tau{=}y_i^\tau|\right)\rangle=\gamma_\tau\psi(\gamma_\tau n)+(1-\gamma_\tau)\psi((1-\gamma_\tau)n). \tag{7}$$

This optimization improves the complexity of calculating $\langle\psi\left(|\hat{\mathbf{y}}^\tau{=}y_i^\tau|\right)\rangle$ from $O(n^2)$ to $O(1)$.

Optimization 2: $\langle \psi(r_i) \rangle$

This optimization stems from the insight that the majority of examples are far away from examples with different labels, the only exceptions are situated around the threshold. Remember that r_i is the number of examples in $\mathbf{z}$ whose Z value is at least as close to z_i as z_i's k-nearest same-label neighbor. So for the majority of examples $r_i = k$. In our sorted setting, only examples that are at most k examples removed from the threshold in $\mathbf{z}$ can have a different value that needs to be computed. This leaves at most $2k$ computations, which is a significant computational speed up as $2k \ll n$. Computing r_i on the sorted array, requires r_i+1 comparisons, because the examples can be compared in order of distance. In the worst case $r_i=\gamma_\tau n+k$ if $\hat{y}_i^\tau=1$ or $r_i=(1-\gamma_\tau)n+k$ if $\hat{y}_i^\tau=0$, but it is usually much smaller, e.g., for uniformly distributed $\mathbf{z}$, it is $r_i \leq 2k$.

This optimization improves the complexity of calculating $\langle \psi(r_i) \rangle$ from $O(n^2)$ to worst case $O(nk)$, but realistically $O(k^2)$.

Optimization 3: Search only the viable threshold range

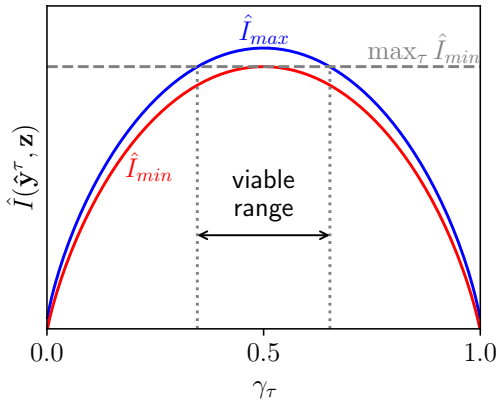


Figure 4: Upper bound $\widehat{MI}_{max}$ and lower bound $\widehat{MI}_{min}$ on $\hat{I}(\hat{\mathbf{y}}^\tau, \mathbf{z})$, for n=1000 and k=5. Thresholds with an upper bound that is lower than the maximum lower bound $\max_\tau \widehat{MI}_{min}$ can never be the optimal threshold and should hence not be considered. The x-axis depicts γ_τ, the ratio of examples in $\mathbf{z}$ that are set as negative by the threshold τ

A key challenge in calculating the optimal split is that there are a large number of thresholds to try ($n-1$ different splits are possible). This optimisation restricts the range where the possible optimal threshold can be situated, hence candidate thresholds outside of this range can be disregarded. Our insight is that for each possible threshold, a lower and upper bound on the MI estimate $\widehat{MI}(\hat{\mathbf{y}}^\tau, \mathbf{z})$ can be calculated directly that only depends on γ_τ, n and k. Figure 4 illustrates our approach. From the lower bound $\widehat{MI}_{min}(\gamma_\tau, n, k)$ (in red) and upper bound $\widehat{MI}_{max}(\gamma_\tau, n, k)$ (in blue), a viable range of thresholds can be calculated. The maximum lower bound $\max_\tau \widehat{MI}_{min}(\gamma_\tau, n, k)$ (in gray) provides a lower bound on the optimal mutual information. Therefore, there is no need to consider thresholds whose upper bound $\widehat{MI}_{max}(\gamma_\tau, n, k)$ is lower than the maximum lower bound. They will never result in the maximum mutual information.

Now, we show how the bounds are computed. The first three terms of Equation 6 can be written using solely γ_τ, n and k (see optimization 1 for the derivation of the third term) and hence can contribute exactly to the bounds. Therefore, the bounds will differ by the bounds on the last term:

$$\widehat{MI}_{min}(\gamma_\tau, n, k) = \psi(n) + \psi(k) - \gamma_\tau \psi(\gamma_\tau n) - (1-\gamma_\tau)\psi((1-\gamma_\tau)n) - \text{upperbound}(\langle \psi(r_i) \rangle)$$
$$\widehat{MI}_{max}(\gamma_\tau, n, k) = \psi(n) + \psi(k) - \gamma_\tau \psi(\gamma_\tau n) - (1-\gamma_\tau)\psi((1-\gamma_\tau)n) - \text{lowerbound}(\langle \psi(r_i) \rangle)$$

The last term of Equation 6 depends on the values in $\mathbf{z}$. Let Δz_{knn} be the difference between z_i and its k-nearest neighbor in $\mathbf{z}$ that shares its predicted label, then $r_i = |\{z_r \in \mathbf{z} \ : \ |z_r - z_i| \leq \Delta z_{knn}\}|$. We derive the lower and upper bounds on the last term from the knowledge that the digamma function ψ strictly increases for positive numbers and the following three facts:

1. As seen for optimization 2, $r_i{=}k$ for all z_i that who's k-nearest same-class neighbor is closer than the threshold τ, i.e., z_i that satisfy $\Delta z_{knn}{<}|z_i{-}\tau|$. This is the case for at least $n{-}2k$ instances, namely the instances that are removed from the threshold by at least k other examples.

2. For the other examples, r_i is greater or equal to k. Together with fact 1 and strictly increasing ψ, this leads to
$$\text{lowerbound}\left(\langle\psi(r_i)\rangle\right) = \psi(k).$$

3. An upperbound on r_i is defined, using the insight that in the worst case, all examples on the other side of the threshold are closer than the closest k examples on the same side. Therefore, $r_i \leq \gamma_\tau n + k$ if $\hat{y}_i^\tau{=}1$ and $r_i \leq (1{-}\gamma_\tau)n + k$ if $\hat{y}_i^\tau{=}0$, where k accounts for the k-nearest neighbor on the same side of the threshold. This upperbound is used for the k examples closest to the threshold on both sides of the threshold:
$$\text{upperbound}\left(\langle\psi\left(r_i\right)\rangle\right) = \frac{n-2k}{n}\psi(k) + \frac{k}{n}\psi(\gamma_\tau n + k) + \frac{k}{n}\psi((1-\gamma_\tau)n + k)$$

This leads to the following upper and lower bound for $\hat{I}(\hat{\mathbf{y}}^\tau, \mathbf{z})$, visualized in Figure 4:

$$\begin{aligned}
\widehat{MI}_{min}(\gamma_\tau, n, k) =& \psi(n) + \frac{2k}{n}\psi(k) - \gamma_\tau\psi(\gamma_\tau n) - (1{-}\gamma_\tau)\psi((1{-}\gamma_\tau)n) \\
&- \frac{k}{n}\psi(\gamma_\tau n + k) - \frac{k}{n}\psi((1-\gamma_\tau)n + k) \\
\widehat{MI}_{max}(\gamma_\tau, n, k) =& \psi(n) + -\gamma_\tau\psi(\gamma_\tau n) - (1{-}\gamma_\tau)\psi((1{-}\gamma_\tau)n)
\end{aligned}$$

We are only interested in the maximum lower bound, because only thresholds with an upperbound above this number have a chance of being the best threshold. Fortunately, the maximum lower bound does not need to be calculated for every candidate threshold, but can be computed directly. Because $\widehat{MI}_{min}$ is a symmetric concave function in γ_τ around its maximum $\gamma_\tau = 0.5$, the maximum value is

$$\begin{aligned}
\max_\tau \widehat{MI}_{min}(\tau, n, k) &= \widehat{MI}_{min}(0.5, n, k) \\
&= -\psi\left(\frac{n}{2}\right) - \frac{n{-}2k}{n}\psi(k) - \frac{2k}{n}\psi\left(\frac{n}{2}{+}k\right)
\end{aligned}$$

Furthermore, because the upper bound is concave and symmetric around its maximum in 0.5, it suffices to find z_i for which $\widehat{MI}_{max}(\gamma_{z_i}, n, k) = \widehat{MI}_{min}(0.5, n, k)$ for $\gamma_{z_i}{\in}[0, 0.5]$ to find the start of the threshold range. Since $\mathbf{z}$ is ordered, the full viable range is then defined as $[z_i, z_{n-i}]$.

Optimization 4: Approximation using m candidate thresholds

We improve the computational efficiency further, by approximating the optimization and computing the MI for only m thresholds. These thresholds are equally spaced, spanning the viable range.

The overall complexity is thus reduced to the complexity of sorting the array ($O(n \log n)$ and the complexity of calculating $\langle\psi(r_i)\rangle$ for the m candidate thresholds. This leads to an overall worst case complexity $O(n \log n + mnk)$, but more realistically $O(n \log n + mk^2)$ for uniform $\mathbf{z}$, which is thus primarily limited by the $O(n \log n)$ complexity of sorting $\mathbf{z}$.[11]

C.2 Algorithm

Algorithm 1 shows our proposed method τ`MIcalibration` for finding the threshold, which implements the proposed optimizations. First, we sort the NTC scores (line 1). Then, we calculate the range for viable thresholds (optimization 3, line 2), in which we select m evenly spaced thresholds (optimization 4, line 3). The evaluation process of every threshold is fast as $r_i{=}k$ for any instance that is at least k instances away from the threshold in the sorted $\mathbf{z}$ (optimization 2, line 7). That leaves at most $2k$ examples for which we have to calculate r_i (line 8).

[11] Sorting a model's predictions is a subroutine in approaches that evaluate the quality of multiple thresholds or set the threshold such that a fraction of the predictions are positive (such as PUSB does).

Finding their k same-class nearest neighbours requires at most $2k$ comparisons on the sorted $\mathbf{z}$ (lines 9-10). Then, $\langle \psi(|\hat{\mathbf{y}}^\tau{=}y_i^\tau|)\rangle$ is directly calculated without any comparisons (optimization 1, line 11). With all of its terms prepared, the MI can now be computed using Equation 6 (line 12). Finally, the best threshold τ_*, is the one with the highest corresponding MI (line 13).

Algorithm 1 τMIcalibration

Given: $\mathbf{z}$ the output of the NTC for unlabeled examples, before the final activation,
$k \geq 1$ the number of neighbors, m the number of thresholds to try.
Compute: $\tau_* = \arg\max_\tau \hat{I}(\hat{\mathbf{y}}^\tau, \mathbf{z})$

1: $\mathbf{z} \leftarrow \text{sort}(\mathbf{z})$
2: Set i such that $\widehat{MI}_{max}(\gamma_{z_i}, n, k){=}\widehat{MI}_{min}(0.5, n, k)$ ▷ *Optimization 3*
3: $\tau \leftarrow \text{linspace}(z_i, z_{n-i}, m)$ ▷ *Optimization 4*
4: $I_{best} \leftarrow -\infty, \quad \tau_* \leftarrow \infty$
5: ***for*** $\tau_j \in \tau$ ***do*** ▷ *try all candidate thresholds*
6: $\quad \hat{\mathbf{y}}^\tau \leftarrow \mathbf{z} > \tau_j$ ▷ *predicted label by thresholding* $\mathbf{z}$ *on* τ_j
7: $\quad \mathbf{r} \leftarrow$ *array*(n), *all values initialized to* k ▷ *Optimization 2*
8: $\quad$ ***for*** $l \in \text{range}(|\hat{\mathbf{y}}^\tau{=}0| - k, |\hat{\mathbf{y}}^\tau{=}0| + k)$ ***do*** ▷ *calculate* r *for examples close to threshold*
9: $\quad\quad \Delta z_{knn} \leftarrow$ *knn distance of* z_l *in* $\{z_i : y_i = y_l\}$ ▷ *dist. to the* k^{th} *nearest same-label neighbor*
10: $\quad\quad r_l \leftarrow |\{z_i{:}|z_i{-}z_l|{\leq}\Delta z_{knn}\}|$ ▷ *number of examples within distance* Δz_{knn}
11: $\quad \langle \psi(|\hat{\mathbf{y}}^\tau{=}y_i^\tau|)\rangle \leftarrow \left(\frac{|\hat{\mathbf{y}}^\tau=0|}{n}\psi(|\hat{\mathbf{y}}^\tau{=}0|) + \frac{|\hat{\mathbf{y}}^\tau=1|}{n}\psi(|\hat{\mathbf{y}}^\tau{=}1|)\right)$ ▷ *Optimization 1*
12: $\quad MI \leftarrow \psi(n) + \psi(k) - \langle \psi(|\hat{\mathbf{y}}^\tau{=}y_i^\tau|)\rangle - \langle \psi(r_i)\rangle$ ▷ *Equation 6*
13: $\quad$ ***if*** $MI > MI_*$ ***then*** $MI_* \leftarrow MI, \quad \tau_* \leftarrow \tau_j$ ▷ *keep threshold with best MI*
14: ***return*** τ_*

D Optimizing MI over unlabeled instances.

Under SAR, in certain settings, it is possible to find three main clusters for the NTC output: labeled, unlabeled positive and unlabeled negative instances. It may be that, by optimizing the MI over the NTC output of all instances, the found threshold splits the NTC output between the labeled and unlabeled positive instances. This threshold does not reflect the optimal threshold that should distinguish between positive and negative examples. By optimizing the MI over only unlabeled instances, we can avoid such misalignment. Figure 5 shows this effect: it is generated by fitting a Logistic Regression model on a one-dimensional feature space consisting of a labeled, unlabeled positive and unlabeled negative cluster. These contain respectively 2000, 1000 and 1000 instances sampled from respectively these normal distributions: $N(-5, 0.3)$, $N(3, 0.5)$ and $N(5, 0.5)$. One can clearly see that by leaving out the labeled instances, we obtain the optimal threshold.

E Neural network architectures for image datasets

We make a distinction between the CIFAR10 dataset and the other three. Since the CIFAR10 dataset is slightly more complicated than the others, we use an all-convolutional-net (Springenberg et al., 2015) with 9 hidden convolutional layers: $[C(3\times3, 96, 1)]*2$ - $C(3\times3, 96, 2)$ - $[C(3\times3, 192, 1)]*2$ - $C(3\times3, 192, 2)$ - $C(3\times3, 192, 1)$ - $C(1\times1, 192, 1)$ - $C(1\times1, 10, 1)$, where $[\cdot]*2$ denotes that we use a layer twice. $C(k\times k, c, s)$ denotes a convolutional layer with kernel size k, c channels and a stride of s. On top of these convolutional layers, we add two hidden fully connected layers of 1000 neurons. For the other three datasets, we use a Multilayer Perceptron with four hidden layers. In this architecture, everything is fully connected and each hidden layer has 300 units. Both architectures apply batch normalization before the hidden layers and a ReLU activation layer after every hidden layer.

F On the use of Mutual Information for PU Learning

This paper proposes to use mutual information (MI) to set a threshold on the output of a NTC. MI has previously been used in PU learning, but in the context of feature selection where $I(X, S){=}0$ if and only if $I(X, Y){=}0$ (Sechidis et al., 2014; Sechidis and Brown, 2018) in a SCAR setting. We mention it for completeness as this is a different use case of MI under the stricter SCAR assumption.

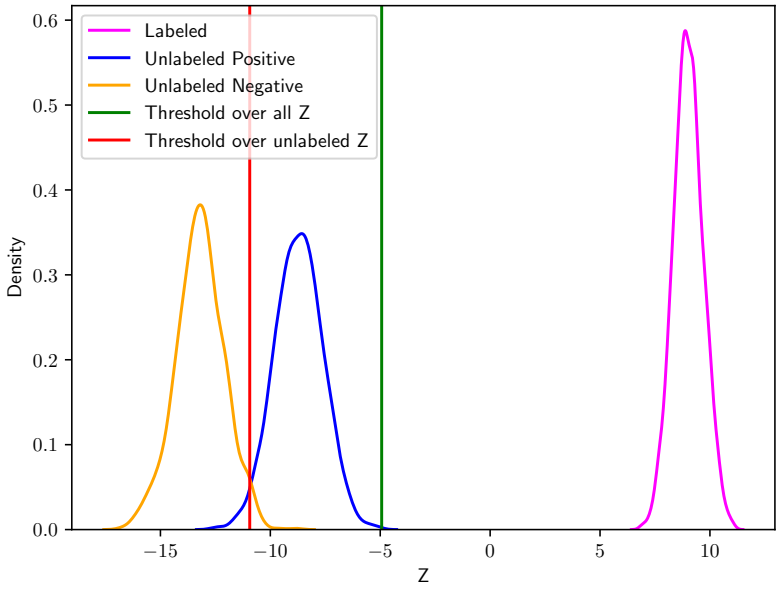


Figure 5: The threshold that optimizes the mutual information over all instances might not be optimal, whereas optimizing only over unlabeled instances does return the optimal threshold in this case. The horizontal axis shows the NTC scores Z and the vertical axis represents the data density.

G Results of experiments

In this section, we present the detailed results of our experiments, which are summarized in the main text. We have compared the following methods:

1. ORACLE (i.e., a model trained on fully labeled data)
2. SAR-EM (Bekker et al., 2019)
3. PUe (Wang et al., 2023)
4. LBE (Gong et al., 2021)
5. PGLIN (Gerych et al., 2022)
6. PUSB (Kato et al., 2019)
7. Our proposed approach `NTC-`τ`MI`

We compare these methods on the datasets in Table 1.

Results for Q1 We compare our approach to the different baselines in terms of balanced accuracy on a held-out test set. First, Tables 6-9 report the averaged balanced accuracy and its standard deviation on each UCI data set under the different labeling mechanisms. Table 6 shows that our `NTC-`τ`MI` method offers the best performance on 12 out of 20 datasets under S1, the SCAR labeling mechanism. Similarly, Table 7, 8 and 9 show that under respectively S2, S3 and S4, we manage to achieve the best performance on 13, 11 and 12 out of 20 datasets. LBE and PUSB are our nearest competitors, but under no labeling mechanism do they achieve more wins (i.e., higher performance on a dataset) than `NTC-`τ`MI`. Unsurprisingly, we manage to outperform PGLIN that makes much stronger assumptions on the labeling mechanism. Table 5 shows that using the F1 score as the evaluation metric gives similar results. Tables 10-13 provide the results for the F1 score on a per dataset basis.

Second, Table 14 and 15 respectively detail the distributions of training time (in seconds) required to train the different methods on the UCI datasets and the speedup of NTC-τMI compared to the baselines. Whereas NTC-τMI and PGLIN have almost constant training times, the other baselines are heavily affected by the dimensionality and/or cardinality of the datasets. Consequently, we achieve massive speed-ups compared to the baselines (excluding PGLIN). An overview is provided in the main text.

Results for Q2 We compare our approach to the baselines, replacing the linear NTC by a deep neural network, which violates the linear classifier assumption of Theorem 3.2. The main text overviews the results in Table 2, whereas Table 16 shows the balanced accuracy of all methods for each considered labeling mechanism. Generally, NTC-τMI outperforms most competitors. An exception is PUSB, to which it offers similar performance over the four labeling mechanisms. However, NTC-τMI always outperforms PUSB on the CIFAR10 dataset, arguably the hardest of the four datasets. We do not report training times as the training time of the methods is dominated by the time required to fit a deep neural network, for which all methods incur a similar cost.

Results for Q3 We evaluate our method on synthetic datasets that are sampled from different data distributions. Figure 6a, 6b and 6c show boxplots for the methods' balanced accuracy on three artificial datasets under respectively S2, S3 and S4. Figure 3 in the main text shows the same under S1. These artificial datasets differ in the distribution of their input features, where only the normally distributed input features meet the ellipticity assumption required by Theorem 3.2. We notice a similar trend in the results as mentioned in the main text about the results under S1. For all three labeling mechanisms, performance of all methods changes little when going from normally distributed input features to uniformly distributed input features. However, for lognormally distributed input features, all methods (including the oracle method) seem to drop in performance. This indicates that the deviation from a normal distribution makes the problem more difficult, regardless of the method. Therefore, the violation of the ellipticity assumption is not the sole reason for the decreased performance of our method. Besides, we still achieve significantly higher balanced accuracy compared to most of the existing SAR-based methods.

Results for Q4 We compare using our τMIcalibration approach to picking the threshold on the NTC's output to PUSB's approach of using the class prior (ground-truth or an estimation) to pick the threshold. The main text discusses the relative error of both approaches compared to the optimal approach, which were detailed in Table 4. Tables 17-20 show the balanced accuracy when using τMIcalibration, the KM2 (Ramaswamy et al., 2016) estimated class prior $\hat{\pi}$ or the ground-truth class prior π to set the threshold on the NTC's output. Additionally, they show the result when using the optimal threshold found by maximizing the balanced accuracy on a fully labeled training set, which can be seen as an upper limit on the balanced accuracy. We make a direct comparison between the τMIcalibration approach and using $\hat{\pi}$ as it is unrealistic to know the class prior. We achieve a better balanced accuracy (i.e., a difference larger than 0.01) in 13, 14, 9 and 11 out of 20 datasets under respectively S1-4, whereas we lose only on 3, 2, 4 and 3 datasets. Moreover, estimating the class prior using the KM2 algorithm is slow as we can see from PUSB's training times in Table 14, whereas τMIcalibration is quite fast.

Table 5: Number of wins (W), losses (L), and draws (D) (absolute value difference in average F1 ≤ 0.01) in terms of F1 for our NTC-τMI versus each competitor on the 20 *UCI* datasets. Results are shown for each of the four labeling mechanisms.

	SAR-EM			PUE			LBE			PGLIN			PUSB		
	W	L	D	W	L	D	W	L	D	W	L	D	W	L	D
S1	**20**	0	0	**20**	0	0	**12**	7	1	**17**	3	0	**15**	4	1
S2	**20**	0	0	**20**	0	0	**15**	4	1	**16**	2	2	**15**	4	1
S3	**20**	0	0	**20**	0	0	**13**	6	1	**16**	3	1	**10**	7	3
S4	**20**	0	0	**20**	0	0	**14**	6	0	**15**	3	2	**13**	4	3

Table 6: Average (± standard deviation) balanced accuracy for all methods on the UCI datasets under labeling mechanism **S1**.

Dataset	Oracle	SAR-EM	PUe	LBE	PGlin	PUSB	NTC-τMI
Abalone	0.500 ± 0.000	0.503 ± 0.005	0.499 ± 0.020	$\mathbf{0.602 \pm 0.020}$	0.502 ± 0.008	0.544 ± 0.060	0.575 ± 0.025
Banknote	0.991 ± 0.005	0.596 ± 0.034	0.548 ± 0.129	$\mathbf{0.967 \pm 0.012}$	0.878 ± 0.058	0.829 ± 0.050	0.922 ± 0.019
Breast-w	0.955 ± 0.019	0.549 ± 0.044	0.500 ± 0.152	0.836 ± 0.063	0.763 ± 0.078	0.766 ± 0.145	$\mathbf{0.870 \pm 0.028}$
Diabetes	0.711 ± 0.033	0.501 ± 0.003	0.506 ± 0.035	0.636 ± 0.051	0.604 ± 0.085	0.546 ± 0.042	$\mathbf{0.700 \pm 0.039}$
Haberman	0.507 ± 0.019	0.500 ± 0.000	0.480 ± 0.045	$\mathbf{0.557 \pm 0.077}$	0.504 ± 0.024	0.513 ± 0.023	0.532 ± 0.066
Heart	0.822 ± 0.040	0.503 ± 0.007	0.496 ± 0.083	0.659 ± 0.069	0.721 ± 0.066	0.527 ± 0.039	$\mathbf{0.757 \pm 0.053}$
Ionosphere	0.847 ± 0.030	0.493 ± 0.013	0.470 ± 0.067	0.666 ± 0.063	0.705 ± 0.096	0.440 ± 0.085	$\mathbf{0.755 \pm 0.059}$
Isolet	0.966 ± 0.014	0.511 ± 0.010	0.500 ± 0.000	0.570 ± 0.022	0.546 ± 0.025	$\mathbf{0.793 \pm 0.072}$	0.757 ± 0.006
Jm1	0.545 ± 0.007	0.518 ± 0.008	0.502 ± 0.028	0.592 ± 0.026	0.504 ± 0.004	$\mathbf{0.628 \pm 0.016}$	0.628 ± 0.013
Kc1	0.598 ± 0.018	0.513 ± 0.014	0.495 ± 0.131	0.615 ± 0.033	0.546 ± 0.035	0.645 ± 0.075	$\mathbf{0.679 \pm 0.030}$
Madelon	0.562 ± 0.011	0.499 ± 0.006	0.500 ± 0.000	0.512 ± 0.017	0.504 ± 0.008	0.496 ± 0.013	$\mathbf{0.519 \pm 0.023}$
Musk	0.828 ± 0.012	0.593 ± 0.031	0.500 ± 0.000	$\mathbf{0.814 \pm 0.029}$	0.625 ± 0.027	0.712 ± 0.036	0.767 ± 0.012
Segment	0.989 ± 0.007	0.526 ± 0.026	0.507 ± 0.161	$\mathbf{0.928 \pm 0.055}$	0.763 ± 0.103	0.848 ± 0.074	0.791 ± 0.014
Semeion	0.932 ± 0.021	0.517 ± 0.018	0.500 ± 0.000	0.572 ± 0.031	0.544 ± 0.033	0.569 ± 0.055	$\mathbf{0.755 \pm 0.022}$
Sonar	0.778 ± 0.068	0.499 ± 0.005	0.495 ± 0.021	0.536 ± 0.060	0.547 ± 0.059	0.497 ± 0.014	$\mathbf{0.573 \pm 0.057}$
Spambase	0.917 ± 0.010	0.564 ± 0.023	0.450 ± 0.068	0.827 ± 0.025	0.532 ± 0.019	0.821 ± 0.031	$\mathbf{0.837 \pm 0.014}$
Vehicle	0.935 ± 0.023	0.547 ± 0.029	0.512 ± 0.090	0.788 ± 0.060	0.680 ± 0.073	0.549 ± 0.067	$\mathbf{0.804 \pm 0.042}$
Waveform	0.842 ± 0.012	0.571 ± 0.031	0.469 ± 0.085	0.813 ± 0.013	0.583 ± 0.026	$\mathbf{0.860 \pm 0.012}$	0.829 ± 0.012
Wdbc	0.973 ± 0.010	0.562 ± 0.033	0.495 ± 0.130	0.678 ± 0.047	0.588 ± 0.065	0.798 ± 0.155	$\mathbf{0.801 \pm 0.043}$
Yeast	0.517 ± 0.010	0.500 ± 0.001	0.499 ± 0.023	0.640 ± 0.040	0.508 ± 0.017	0.517 ± 0.051	$\mathbf{0.657 \pm 0.024}$

Table 7: Average (± standard deviation) balanced accuracy for all methods on the UCI datasets under labeling mechanism **S2**.

Dataset	Oracle	SAR-EM	PUe	LBE	PGlin	PUSB	NTC-τMI
Abalone	0.500 ± 0.000	0.504 ± 0.013	0.499 ± 0.020	0.514 ± 0.020	0.504 ± 0.017	0.495 ± 0.017	$\mathbf{0.518 \pm 0.039}$
Banknote	0.991 ± 0.005	0.595 ± 0.034	0.548 ± 0.129	$\mathbf{0.970 \pm 0.013}$	0.874 ± 0.060	0.826 ± 0.046	0.924 ± 0.020
Breast-w	0.955 ± 0.019	0.532 ± 0.038	0.500 ± 0.152	0.819 ± 0.061	0.736 ± 0.069	0.781 ± 0.113	$\mathbf{0.877 \pm 0.027}$
Diabetes	0.711 ± 0.033	0.501 ± 0.003	0.506 ± 0.035	0.647 ± 0.056	0.614 ± 0.069	0.536 ± 0.043	$\mathbf{0.700 \pm 0.027}$
Haberman	0.507 ± 0.019	0.500 ± 0.000	0.480 ± 0.045	$\mathbf{0.526 \pm 0.069}$	0.509 ± 0.026	0.499 ± 0.006	0.509 ± 0.052
Heart	0.822 ± 0.040	0.501 ± 0.004	0.496 ± 0.083	0.635 ± 0.058	0.724 ± 0.066	0.504 ± 0.012	$\mathbf{0.762 \pm 0.055}$
Ionosphere	0.847 ± 0.030	0.492 ± 0.015	0.470 ± 0.067	0.674 ± 0.065	0.691 ± 0.095	0.401 ± 0.061	$\mathbf{0.738 \pm 0.051}$
Isolet	0.966 ± 0.014	0.510 ± 0.010	0.500 ± 0.000	0.569 ± 0.023	0.539 ± 0.020	$\mathbf{0.794 \pm 0.070}$	0.758 ± 0.007
Jm1	0.545 ± 0.007	0.516 ± 0.007	0.502 ± 0.028	0.588 ± 0.022	0.503 ± 0.001	$\mathbf{0.624 \pm 0.014}$	0.617 ± 0.015
Kc1	0.598 ± 0.018	0.512 ± 0.013	0.497 ± 0.131	0.566 ± 0.045	0.520 ± 0.014	0.519 ± 0.052	$\mathbf{0.694 \pm 0.032}$
Madelon	0.562 ± 0.011	0.500 ± 0.003	0.500 ± 0.000	0.508 ± 0.014	0.501 ± 0.006	0.497 ± 0.008	$\mathbf{0.526 \pm 0.020}$
Musk	0.828 ± 0.012	0.593 ± 0.032	0.500 ± 0.000	0.739 ± 0.034	0.587 ± 0.031	0.671 ± 0.040	$\mathbf{0.752 \pm 0.014}$
Segment	0.989 ± 0.007	0.526 ± 0.026	0.508 ± 0.161	$\mathbf{0.918 \pm 0.052}$	0.760 ± 0.096	0.813 ± 0.106	0.791 ± 0.013
Semeion	0.932 ± 0.021	0.514 ± 0.018	0.500 ± 0.000	0.569 ± 0.030	0.539 ± 0.031	0.560 ± 0.054	$\mathbf{0.755 \pm 0.024}$
Sonar	0.778 ± 0.068	0.500 ± 0.000	0.495 ± 0.021	0.535 ± 0.058	0.562 ± 0.064	0.502 ± 0.009	$\mathbf{0.564 \pm 0.077}$
Spambase	0.917 ± 0.010	0.560 ± 0.021	0.451 ± 0.068	0.834 ± 0.022	0.526 ± 0.013	0.794 ± 0.030	$\mathbf{0.838 \pm 0.014}$
Vehicle	0.935 ± 0.023	0.541 ± 0.034	0.511 ± 0.089	0.743 ± 0.076	0.647 ± 0.075	0.536 ± 0.054	$\mathbf{0.800 \pm 0.044}$
Waveform	0.842 ± 0.012	0.539 ± 0.017	0.469 ± 0.084	0.791 ± 0.026	0.546 ± 0.019	$\mathbf{0.842 \pm 0.013}$	0.833 ± 0.013
Wdbc	0.973 ± 0.010	0.563 ± 0.034	0.495 ± 0.130	0.679 ± 0.044	0.588 ± 0.066	0.778 ± 0.166	$\mathbf{0.810 \pm 0.045}$
Yeast	0.517 ± 0.010	0.501 ± 0.002	0.499 ± 0.023	$\mathbf{0.636 \pm 0.053}$	0.506 ± 0.019	0.515 ± 0.031	0.634 ± 0.039

Table 8: Average (± standard deviation) balanced accuracy for all methods on the UCI datasets under labeling mechanism **S3**.

Dataset	Oracle	SAR-EM	PUe	LBE	PGLin	PUSB	NTC-τMI
Abalone	0.500 ± 0.000	0.504 ± 0.008	0.499 ± 0.020	0.561 ± 0.016	0.500 ± 0.001	0.596 ± 0.023	$\mathbf{0.599 \pm 0.042}$
Banknote	0.991 ± 0.005	0.621 ± 0.033	0.548 ± 0.129	$\mathbf{0.968 \pm 0.012}$	0.886 ± 0.055	0.848 ± 0.040	0.924 ± 0.019
Breast-w	0.955 ± 0.019	0.566 ± 0.059	0.500 ± 0.152	0.848 ± 0.053	0.780 ± 0.067	0.792 ± 0.157	$\mathbf{0.874 \pm 0.029}$
Diabetes	0.711 ± 0.033	0.502 ± 0.003	0.506 ± 0.035	0.629 ± 0.043	0.634 ± 0.054	0.575 ± 0.050	$\mathbf{0.700 \pm 0.031}$
Haberman	0.507 ± 0.019	0.500 ± 0.000	0.480 ± 0.045	$\mathbf{0.555 \pm 0.055}$	0.502 ± 0.008	0.507 ± 0.031	0.533 ± 0.059
Heart	0.822 ± 0.040	0.509 ± 0.014	0.496 ± 0.083	0.663 ± 0.066	0.720 ± 0.070	0.578 ± 0.066	$\mathbf{0.754 \pm 0.058}$
Ionosphere	0.847 ± 0.030	0.491 ± 0.019	0.470 ± 0.067	0.686 ± 0.053	0.722 ± 0.073	0.428 ± 0.104	$\mathbf{0.780 \pm 0.036}$
Isolet	0.966 ± 0.014	0.517 ± 0.011	0.500 ± 0.000	0.582 ± 0.028	0.555 ± 0.023	$\mathbf{0.808 \pm 0.068}$	0.758 ± 0.007
Jm1	0.545 ± 0.007	0.525 ± 0.007	0.502 ± 0.027	0.536 ± 0.026	0.505 ± 0.004	0.629 ± 0.016	$\mathbf{0.631 \pm 0.012}$
Kc1	0.598 ± 0.018	0.526 ± 0.027	0.493 ± 0.130	0.621 ± 0.045	0.551 ± 0.028	0.654 ± 0.068	$\mathbf{0.682 \pm 0.030}$
Madelon	0.562 ± 0.011	0.501 ± 0.005	0.500 ± 0.000	0.511 ± 0.018	0.506 ± 0.008	0.500 ± 0.013	$\mathbf{0.517 \pm 0.024}$
Musk	0.828 ± 0.012	0.558 ± 0.014	0.500 ± 0.000	$\mathbf{0.852 \pm 0.020}$	0.647 ± 0.022	0.732 ± 0.031	0.770 ± 0.011
Segment	0.989 ± 0.007	0.537 ± 0.042	0.505 ± 0.161	$\mathbf{0.945 \pm 0.044}$	0.790 ± 0.101	0.874 ± 0.066	0.789 ± 0.014
Semeion	0.932 ± 0.021	0.523 ± 0.024	0.500 ± 0.000	0.587 ± 0.039	0.556 ± 0.034	0.637 ± 0.104	$\mathbf{0.758 \pm 0.018}$
Sonar	0.778 ± 0.068	0.497 ± 0.007	0.495 ± 0.021	0.557 ± 0.062	0.572 ± 0.065	0.506 ± 0.024	$\mathbf{0.598 \pm 0.061}$
Spambase	0.917 ± 0.010	0.568 ± 0.020	0.446 ± 0.070	0.827 ± 0.020	0.548 ± 0.034	$\mathbf{0.845 \pm 0.023}$	0.837 ± 0.013
Vehicle	0.935 ± 0.023	0.552 ± 0.028	0.511 ± 0.087	$\mathbf{0.831 \pm 0.060}$	0.720 ± 0.086	0.581 ± 0.073	0.807 ± 0.029
Waveform	0.842 ± 0.012	0.599 ± 0.031	0.470 ± 0.084	0.821 ± 0.011	0.623 ± 0.025	$\mathbf{0.863 \pm 0.009}$	0.833 ± 0.009
Wdbc	0.973 ± 0.010	0.581 ± 0.045	0.487 ± 0.131	0.701 ± 0.051	0.603 ± 0.056	$\mathbf{0.843 \pm 0.120}$	0.799 ± 0.039
Yeast	0.517 ± 0.010	0.500 ± 0.001	0.499 ± 0.023	0.616 ± 0.032	0.518 ± 0.033	0.527 ± 0.055	$\mathbf{0.668 \pm 0.019}$

Table 9: Average (± standard deviation) balanced accuracy for all methods on the UCI datasets under labeling mechanism **S4**.

Dataset	Oracle	SAR-EM	PUe	LBE	PGLin	PUSB	NTC-τMI
Abalone	0.500 ± 0.000	0.511 ± 0.012	0.499 ± 0.020	0.528 ± 0.008	0.503 ± 0.004	0.615 ± 0.026	$\mathbf{0.624 \pm 0.030}$
Banknote	0.991 ± 0.005	0.512 ± 0.016	0.552 ± 0.130	$\mathbf{0.883 \pm 0.057}$	0.757 ± 0.100	0.580 ± 0.062	0.862 ± 0.042
Breast-w	0.955 ± 0.019	0.587 ± 0.048	0.500 ± 0.152	0.810 ± 0.060	0.816 ± 0.076	0.560 ± 0.028	$\mathbf{0.849 \pm 0.026}$
Diabetes	0.711 ± 0.033	0.503 ± 0.004	0.506 ± 0.035	0.635 ± 0.033	0.624 ± 0.080	0.591 ± 0.056	$\mathbf{0.708 \pm 0.041}$
Haberman	0.507 ± 0.019	0.500 ± 0.006	0.480 ± 0.045	0.541 ± 0.055	0.502 ± 0.006	0.511 ± 0.051	$\mathbf{0.543 \pm 0.051}$
Heart	0.822 ± 0.040	0.509 ± 0.016	0.496 ± 0.083	0.673 ± 0.076	0.687 ± 0.073	0.591 ± 0.063	$\mathbf{0.712 \pm 0.063}$
Ionosphere	0.847 ± 0.030	0.491 ± 0.018	0.470 ± 0.067	0.667 ± 0.071	0.707 ± 0.074	0.411 ± 0.128	$\mathbf{0.759 \pm 0.058}$
Isolet	0.966 ± 0.014	0.504 ± 0.007	0.500 ± 0.000	0.546 ± 0.023	0.521 ± 0.017	0.750 ± 0.134	$\mathbf{0.759 \pm 0.006}$
Jm1	0.545 ± 0.006	0.530 ± 0.009	0.501 ± 0.027	0.508 ± 0.007	0.522 ± 0.021	$\mathbf{0.631 \pm 0.012}$	0.630 ± 0.010
Kc1	0.598 ± 0.018	0.539 ± 0.034	0.491 ± 0.132	0.604 ± 0.045	0.591 ± 0.053	0.681 ± 0.059	$\mathbf{0.693 \pm 0.029}$
Madelon	0.562 ± 0.011	0.500 ± 0.010	0.500 ± 0.000	$\mathbf{0.514 \pm 0.022}$	0.504 ± 0.015	0.497 ± 0.018	0.513 ± 0.021
Musk	0.828 ± 0.012	0.554 ± 0.022	0.500 ± 0.000	$\mathbf{0.869 \pm 0.013}$	0.670 ± 0.052	0.690 ± 0.038	0.770 ± 0.009
Segment	0.989 ± 0.007	0.536 ± 0.042	0.508 ± 0.162	$\mathbf{0.941 \pm 0.050}$	0.838 ± 0.087	0.820 ± 0.073	0.790 ± 0.013
Semeion	0.932 ± 0.021	0.514 ± 0.022	0.500 ± 0.000	0.555 ± 0.029	0.538 ± 0.028	0.545 ± 0.051	$\mathbf{0.743 \pm 0.031}$
Sonar	0.778 ± 0.068	0.500 ± 0.016	0.495 ± 0.021	0.565 ± 0.071	0.542 ± 0.072	0.500 ± 0.031	$\mathbf{0.594 \pm 0.067}$
Spambase	0.917 ± 0.010	0.511 ± 0.026	0.470 ± 0.075	0.691 ± 0.027	0.534 ± 0.029	0.575 ± 0.060	$\mathbf{0.700 \pm 0.020}$
Vehicle	0.935 ± 0.023	0.557 ± 0.030	0.513 ± 0.088	$\mathbf{0.854 \pm 0.069}$	0.752 ± 0.100	0.603 ± 0.072	0.797 ± 0.044
Waveform	0.842 ± 0.012	0.638 ± 0.040	0.470 ± 0.084	0.813 ± 0.011	0.685 ± 0.041	$\mathbf{0.844 \pm 0.016}$	0.827 ± 0.010
Wdbc	0.973 ± 0.010	0.504 ± 0.006	$\mathbf{0.537 \pm 0.113}$	0.525 ± 0.024	0.530 ± 0.033	0.507 ± 0.015	0.537 ± 0.060
Yeast	0.517 ± 0.010	0.501 ± 0.002	0.499 ± 0.023	0.605 ± 0.019	0.524 ± 0.022	0.609 ± 0.038	$\mathbf{0.682 \pm 0.028}$

Table 10: Average (± standard deviation) F1 for all methods on the UCI datasets under labeling mechanism **S1**.

Dataset	Oracle	SAR-EM	PUe	LBE	PGLin	PUSB	NTC-τMI
Abalone	0.000 ± 0.000	0.030 ± 0.021	0.117 ± 0.125	$\mathbf{0.328 \pm 0.021}$	0.279 ± 0.013	0.253 ± 0.087	0.306 ± 0.026
Banknote	0.990 ± 0.006	0.317 ± 0.094	0.496 ± 0.145	$\mathbf{0.961 \pm 0.014}$	0.856 ± 0.078	0.789 ± 0.074	0.912 ± 0.019
Breast-w	0.944 ± 0.025	0.166 ± 0.138	0.263 ± 0.251	0.789 ± 0.088	0.673 ± 0.148	0.639 ± 0.290	$\mathbf{0.808 \pm 0.031}$
Diabetes	0.607 ± 0.056	0.003 ± 0.009	0.257 ± 0.221	0.569 ± 0.059	0.561 ± 0.059	0.217 ± 0.140	$\mathbf{0.625 \pm 0.046}$
Haberman	0.044 ± 0.057	0.000 ± 0.000	0.229 ± 0.196	0.407 ± 0.085	$\mathbf{0.408 \pm 0.054}$	0.085 ± 0.075	0.374 ± 0.089
Heart	0.791 ± 0.051	0.011 ± 0.027	0.305 ± 0.262	0.543 ± 0.126	0.659 ± 0.108	0.095 ± 0.135	$\mathbf{0.732 \pm 0.061}$
Ionosphere	0.917 ± 0.023	0.011 ± 0.029	0.118 ± 0.209	0.740 ± 0.079	0.711 ± 0.224	0.083 ± 0.096	$\mathbf{0.788 \pm 0.055}$
Isolet	0.951 ± 0.014	0.042 ± 0.037	0.000 ± 0.000	0.232 ± 0.063	0.163 ± 0.083	$\mathbf{0.575 \pm 0.134}$	0.143 ± 0.012
Jm1	0.184 ± 0.019	0.083 ± 0.031	0.135 ± 0.132	0.370 ± 0.021	0.020 ± 0.016	$\mathbf{0.397 \pm 0.018}$	0.395 ± 0.012
Kc1	0.326 ± 0.045	0.063 ± 0.054	0.229 ± 0.105	0.334 ± 0.044	0.167 ± 0.107	0.357 ± 0.147	$\mathbf{0.381 \pm 0.026}$
Madelon	0.555 ± 0.018	0.044 ± 0.015	0.000 ± 0.000	0.337 ± 0.049	0.109 ± 0.022	0.137 ± 0.024	$\mathbf{0.531 \pm 0.030}$
Musk	0.761 ± 0.018	0.307 ± 0.085	0.000 ± 0.000	$\mathbf{0.650 \pm 0.038}$	0.394 ± 0.068	0.569 ± 0.070	0.447 ± 0.018
Segment	0.987 ± 0.009	0.096 ± 0.091	0.214 ± 0.142	$\mathbf{0.896 \pm 0.072}$	0.664 ± 0.184	0.775 ± 0.117	0.447 ± 0.021
Semeion	0.871 ± 0.024	0.065 ± 0.067	0.000 ± 0.000	0.245 ± 0.088	0.157 ± 0.110	0.226 ± 0.174	$\mathbf{0.326 \pm 0.046}$
Sonar	0.794 ± 0.066	0.000 ± 0.000	0.038 ± 0.117	0.324 ± 0.137	0.521 ± 0.162	0.040 ± 0.065	$\mathbf{0.594 \pm 0.080}$
Spambase	0.902 ± 0.011	0.234 ± 0.072	0.382 ± 0.080	0.790 ± 0.032	0.124 ± 0.068	0.781 ± 0.043	$\mathbf{0.801 \pm 0.016}$
Vehicle	0.909 ± 0.029	0.170 ± 0.095	0.325 ± 0.119	$\mathbf{0.700 \pm 0.093}$	0.508 ± 0.157	0.192 ± 0.182	0.645 ± 0.057
Waveform	0.787 ± 0.013	0.259 ± 0.097	0.243 ± 0.165	0.741 ± 0.017	0.297 ± 0.077	$\mathbf{0.795 \pm 0.016}$	0.754 ± 0.017
Wdbc	0.967 ± 0.014	0.216 ± 0.104	0.397 ± 0.164	0.541 ± 0.092	0.283 ± 0.181	0.678 ± 0.344	$\mathbf{0.750 \pm 0.049}$
Yeast	0.139 ± 0.032	0.002 ± 0.005	0.175 ± 0.219	$\mathbf{0.554 \pm 0.040}$	0.479 ± 0.020	0.191 ± 0.138	0.550 ± 0.036

Table 11: Average (± standard deviation) F1 for all methods on the UCI datasets under labeling mechanism **S2**.

Dataset	Oracle	SAR-EM	PUe	LBE	PGLin	PUSB	NTC-τMI
Abalone	0.000 ± 0.000	0.039 ± 0.049	0.117 ± 0.125	0.079 ± 0.084	$\mathbf{0.265 \pm 0.031}$	0.045 ± 0.047	0.259 ± 0.035
Banknote	0.990 ± 0.006	0.314 ± 0.098	0.496 ± 0.145	$\mathbf{0.964 \pm 0.015}$	0.850 ± 0.081	0.786 ± 0.070	0.914 ± 0.022
Breast-w	0.944 ± 0.025	0.112 ± 0.124	0.263 ± 0.251	0.768 ± 0.089	0.627 ± 0.141	0.685 ± 0.235	$\mathbf{0.816 \pm 0.032}$
Diabetes	0.607 ± 0.056	0.006 ± 0.012	0.257 ± 0.221	0.524 ± 0.090	0.561 ± 0.059	0.152 ± 0.165	$\mathbf{0.625 \pm 0.033}$
Haberman	0.044 ± 0.057	0.000 ± 0.000	0.229 ± 0.196	0.250 ± 0.145	$\mathbf{0.409 \pm 0.056}$	0.005 ± 0.022	0.351 ± 0.073
Heart	0.791 ± 0.051	0.004 ± 0.017	0.305 ± 0.262	0.469 ± 0.130	0.645 ± 0.113	0.017 ± 0.050	$\mathbf{0.735 \pm 0.066}$
Ionosphere	0.917 ± 0.023	0.008 ± 0.021	0.118 ± 0.209	0.730 ± 0.074	0.654 ± 0.238	0.029 ± 0.055	$\mathbf{0.775 \pm 0.047}$
Isolet	0.951 ± 0.014	0.038 ± 0.037	0.000 ± 0.000	0.231 ± 0.067	0.143 ± 0.070	$\mathbf{0.578 \pm 0.130}$	0.143 ± 0.012
Jm1	0.184 ± 0.019	0.071 ± 0.027	0.135 ± 0.132	0.319 ± 0.050	0.012 ± 0.006	$\mathbf{0.393 \pm 0.021}$	0.385 ± 0.015
Kc1	0.326 ± 0.045	0.052 ± 0.048	0.232 ± 0.105	0.235 ± 0.115	0.080 ± 0.052	0.053 ± 0.134	$\mathbf{0.393 \pm 0.028}$
Madelon	0.555 ± 0.017	0.012 ± 0.008	0.000 ± 0.000	0.202 ± 0.043	0.041 ± 0.020	0.075 ± 0.019	$\mathbf{0.537 \pm 0.024}$
Musk	0.761 ± 0.018	0.308 ± 0.093	0.000 ± 0.000	$\mathbf{0.612 \pm 0.056}$	0.293 ± 0.088	0.493 ± 0.089	0.436 ± 0.019
Segment	0.987 ± 0.009	0.096 ± 0.090	0.215 ± 0.142	$\mathbf{0.893 \pm 0.066}$	0.662 ± 0.171	0.720 ± 0.208	0.446 ± 0.020
Semeion	0.871 ± 0.024	0.053 ± 0.067	0.000 ± 0.000	0.237 ± 0.087	0.141 ± 0.102	0.196 ± 0.172	$\mathbf{0.326 \pm 0.045}$
Sonar	0.794 ± 0.066	0.000 ± 0.000	0.038 ± 0.117	0.278 ± 0.133	0.494 ± 0.168	0.010 ± 0.045	$\mathbf{0.597 \pm 0.087}$
Spambase	0.902 ± 0.011	0.223 ± 0.064	0.383 ± 0.080	0.799 ± 0.029	0.102 ± 0.047	0.741 ± 0.045	$\mathbf{0.802 \pm 0.014}$
Vehicle	0.909 ± 0.029	0.146 ± 0.111	0.324 ± 0.118	0.629 ± 0.133	0.435 ± 0.167	0.136 ± 0.165	$\mathbf{0.641 \pm 0.056}$
Waveform	0.787 ± 0.013	0.153 ± 0.062	0.243 ± 0.165	0.721 ± 0.031	0.178 ± 0.065	$\mathbf{0.780 \pm 0.013}$	0.758 ± 0.019
Wdbc	0.967 ± 0.014	0.219 ± 0.107	0.397 ± 0.165	0.541 ± 0.086	0.280 ± 0.183	0.633 ± 0.370	$\mathbf{0.759 \pm 0.050}$
Yeast	0.139 ± 0.032	0.003 ± 0.006	0.175 ± 0.219	0.499 ± 0.082	0.479 ± 0.019	0.120 ± 0.099	$\mathbf{0.526 \pm 0.048}$

Table 12: Average (± standard deviation) F1 for all methods on the UCI datasets under labeling mechanism **S3**.

Dataset	Oracle	SAR-EM	PUe	LBE	PGLin	PUSB	NTC-τMI
Abalone	0.000 ± 0.000	0.049 ± 0.033	0.117 ± 0.125	0.305 ± 0.017	0.278 ± 0.013	0.322 ± 0.023	$\mathbf{0.325 \pm 0.036}$
Banknote	0.990 ± 0.006	0.386 ± 0.087	0.496 ± 0.145	$\mathbf{0.962 \pm 0.014}$	0.866 ± 0.075	0.819 ± 0.055	0.914 ± 0.018
Breast-w	0.944 ± 0.025	0.217 ± 0.167	0.263 ± 0.251	0.804 ± 0.071	0.707 ± 0.113	0.678 ± 0.287	$\mathbf{0.813 \pm 0.034}$
Diabetes	0.607 ± 0.056	0.008 ± 0.013	0.257 ± 0.221	0.572 ± 0.055	0.573 ± 0.058	0.310 ± 0.130	$\mathbf{0.625 \pm 0.041}$
Haberman	0.044 ± 0.057	0.000 ± 0.000	0.229 ± 0.196	$\mathbf{0.418 \pm 0.067}$	0.404 ± 0.046	0.095 ± 0.086	0.375 ± 0.075
Heart	0.791 ± 0.051	0.034 ± 0.051	0.305 ± 0.262	0.576 ± 0.095	0.654 ± 0.109	0.255 ± 0.195	$\mathbf{0.730 \pm 0.063}$
Ionosphere	0.917 ± 0.023	0.013 ± 0.030	0.118 ± 0.209	0.777 ± 0.068	0.732 ± 0.202	0.105 ± 0.127	$\mathbf{0.816 \pm 0.037}$
Isolet	0.951 ± 0.014	0.064 ± 0.042	0.000 ± 0.000	0.260 ± 0.073	0.194 ± 0.076	$\mathbf{0.599 \pm 0.126}$	0.143 ± 0.012
Jm1	0.184 ± 0.019	0.110 ± 0.028	0.124 ± 0.134	0.340 ± 0.015	0.023 ± 0.015	0.397 ± 0.017	$\mathbf{0.398 \pm 0.012}$
Kc1	0.326 ± 0.045	0.106 ± 0.089	0.227 ± 0.106	0.337 ± 0.045	0.189 ± 0.090	0.380 ± 0.100	$\mathbf{0.383 \pm 0.029}$
Madelon	0.555 ± 0.018	0.077 ± 0.013	0.000 ± 0.000	0.404 ± 0.045	0.158 ± 0.028	0.189 ± 0.027	$\mathbf{0.530 \pm 0.024}$
Musk	0.761 ± 0.018	0.208 ± 0.042	0.000 ± 0.000	$\mathbf{0.662 \pm 0.023}$	0.447 ± 0.050	0.600 ± 0.055	0.449 ± 0.018
Segment	0.987 ± 0.009	0.128 ± 0.138	0.213 ± 0.142	$\mathbf{0.907 \pm 0.053}$	0.709 ± 0.188	0.812 ± 0.088	0.444 ± 0.022
Semeion	0.871 ± 0.024	0.085 ± 0.084	0.000 ± 0.000	0.285 ± 0.100	0.195 ± 0.108	$\mathbf{0.354 \pm 0.171}$	0.328 ± 0.042
Sonar	0.794 ± 0.066	0.011 ± 0.025	0.038 ± 0.117	0.414 ± 0.110	0.555 ± 0.150	0.116 ± 0.111	$\mathbf{0.624 \pm 0.089}$
Spambase	0.902 ± 0.011	0.248 ± 0.061	0.379 ± 0.081	0.790 ± 0.025	0.179 ± 0.108	$\mathbf{0.814 \pm 0.029}$	0.801 ± 0.014
Vehicle	0.909 ± 0.029	0.186 ± 0.093	0.324 ± 0.117	$\mathbf{0.756 \pm 0.088}$	0.582 ± 0.167	0.276 ± 0.199	0.646 ± 0.044
Waveform	0.787 ± 0.013	0.346 ± 0.086	0.243 ± 0.165	0.746 ± 0.017	0.408 ± 0.064	$\mathbf{0.794 \pm 0.017}$	0.757 ± 0.015
Wdbc	0.967 ± 0.014	0.270 ± 0.132	0.389 ± 0.164	0.589 ± 0.095	0.331 ± 0.153	$\mathbf{0.772 \pm 0.261}$	0.749 ± 0.037
Yeast	0.139 ± 0.032	0.002 ± 0.005	0.175 ± 0.219	0.540 ± 0.030	0.485 ± 0.025	0.222 ± 0.136	$\mathbf{0.562 \pm 0.031}$

Table 13: Average (± standard deviation) F1 for all methods on the UCI datasets under labeling mechanism **S4**.

Dataset	Oracle	SAR-EM	PUe	LBE	PGLin	PUSB	NTC-τMI
Abalone	0.000 ± 0.000	0.101 ± 0.035	0.117 ± 0.125	0.290 ± 0.014	0.279 ± 0.013	0.337 ± 0.020	$\mathbf{0.346 \pm 0.029}$
Banknote	0.990 ± 0.006	0.044 ± 0.058	0.502 ± 0.144	$\mathbf{0.867 \pm 0.072}$	0.661 ± 0.180	0.258 ± 0.191	0.848 ± 0.049
Breast-w	0.944 ± 0.025	0.286 ± 0.141	0.263 ± 0.251	0.751 ± 0.081	0.761 ± 0.122	0.209 ± 0.093	$\mathbf{0.784 \pm 0.031}$
Diabetes	0.607 ± 0.056	0.014 ± 0.015	0.257 ± 0.221	0.588 ± 0.041	0.582 ± 0.061	0.375 ± 0.118	$\mathbf{0.635 \pm 0.053}$
Haberman	0.044 ± 0.057	0.005 ± 0.021	0.229 ± 0.196	$\mathbf{0.425 \pm 0.073}$	0.408 ± 0.054	0.174 ± 0.118	0.385 ± 0.075
Heart	0.791 ± 0.051	0.034 ± 0.058	0.305 ± 0.262	0.616 ± 0.091	0.652 ± 0.076	0.309 ± 0.181	$\mathbf{0.687 \pm 0.070}$
Ionosphere	0.917 ± 0.023	0.013 ± 0.032	0.118 ± 0.209	0.765 ± 0.070	$\mathbf{0.800 \pm 0.125}$	0.121 ± 0.165	0.797 ± 0.054
Isolet	0.951 ± 0.014	0.014 ± 0.028	0.000 ± 0.000	0.161 ± 0.071	0.078 ± 0.061	$\mathbf{0.492 \pm 0.259}$	0.144 ± 0.012
Jm1	0.184 ± 0.019	0.134 ± 0.034	0.135 ± 0.132	0.328 ± 0.009	0.094 ± 0.081	$\mathbf{0.398 \pm 0.012}$	0.397 ± 0.009
Kc1	0.326 ± 0.045	0.152 ± 0.101	0.224 ± 0.109	0.322 ± 0.042	0.285 ± 0.115	$\mathbf{0.399 \pm 0.073}$	0.391 ± 0.026
Madelon	0.555 ± 0.017	0.129 ± 0.033	0.000 ± 0.000	0.495 ± 0.042	0.257 ± 0.054	0.252 ± 0.028	$\mathbf{0.528 \pm 0.024}$
Musk	0.761 ± 0.018	0.198 ± 0.068	0.000 ± 0.000	$\mathbf{0.620 \pm 0.021}$	0.476 ± 0.105	0.509 ± 0.075	0.449 ± 0.017
Segment	0.987 ± 0.009	0.124 ± 0.134	0.215 ± 0.142	$\mathbf{0.882 \pm 0.067}$	0.790 ± 0.161	0.732 ± 0.117	0.446 ± 0.025
Semeion	0.871 ± 0.024	0.052 ± 0.078	0.000 ± 0.000	0.196 ± 0.088	0.137 ± 0.092	0.147 ± 0.166	$\mathbf{0.318 \pm 0.048}$
Sonar	0.794 ± 0.066	0.034 ± 0.045	0.038 ± 0.117	0.489 ± 0.117	0.574 ± 0.136	0.198 ± 0.090	$\mathbf{0.629 \pm 0.083}$
Spambase	0.902 ± 0.011	0.098 ± 0.052	0.402 ± 0.087	0.643 ± 0.031	0.183 ± 0.125	0.291 ± 0.143	$\mathbf{0.658 \pm 0.023}$
Vehicle	0.909 ± 0.029	0.203 ± 0.097	0.326 ± 0.117	$\mathbf{0.771 \pm 0.093}$	0.629 ± 0.181	0.345 ± 0.173	0.637 ± 0.048
Waveform	0.787 ± 0.013	0.440 ± 0.101	0.244 ± 0.165	0.735 ± 0.019	0.548 ± 0.078	$\mathbf{0.767 \pm 0.029}$	0.750 ± 0.015
Wdbc	0.967 ± 0.014	0.016 ± 0.023	0.443 ± 0.149	0.192 ± 0.095	0.148 ± 0.123	0.027 ± 0.055	$\mathbf{0.466 \pm 0.073}$
Yeast	0.139 ± 0.032	0.003 ± 0.006	0.175 ± 0.219	0.533 ± 0.025	0.487 ± 0.020	0.431 ± 0.078	$\mathbf{0.577 \pm 0.036}$

Table 14: The 25% quartile, median and 75% quartile of training time (in seconds) per UCI dataset of the different methods. The values are calculated over the different labeling mechanisms.

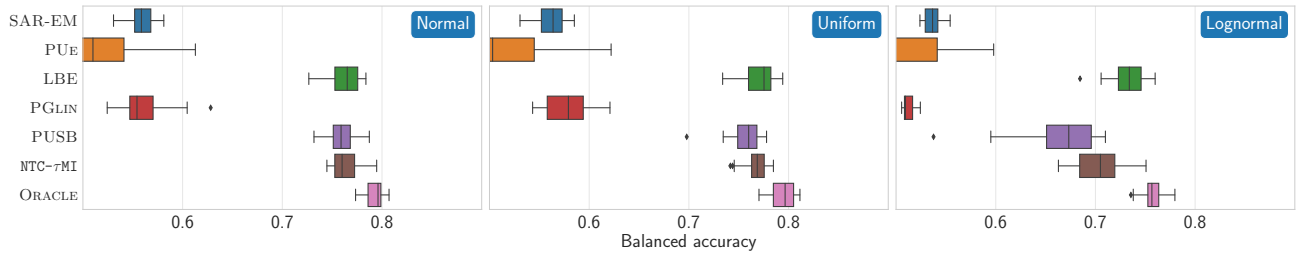
	SAR-EM			PUe			LBE			PGLin			PUSB			NTC-τMI		
L.S.	25%	Median	75%	25%	Median	75%	25%	Median	75%	25%	Median	75%	25%	Median	75%	25%	Median	75%
Abalone	7.72	11.97	16.85	695.66	1375.64	1832.64	44.84	62.39	64.98	0.02	0.02	0.03	920.72	1910.46	3149.45	0.14	0.21	0.31
Banknote	1.72	2.61	3.77	35.23	68.04	94.02	36.47	46.63	49.64	0.01	0.01	0.01	163.24	333.08	601.43	0.19	0.21	0.28
Breast-w	2.08	3.52	4.18	11.85	13.01	13.74	41.20	43.53	44.84	0.01	0.02	0.02	210.69	218.73	226.23	0.21	0.23	0.28
Diabetes	4.82	5.13	5.41	16.26	17.45	18.24	42.71	43.92	45.41	0.01	0.02	0.02	251.25	263.80	274.61	0.20	0.24	0.27
Haberman	1.19	1.66	2.06	0.60	1.02	1.32	26.41	38.82	40.78	0.01	0.01	0.01	24.25	51.16	83.13	0.12	0.18	0.24
Heart	1.78	2.45	2.83	0.54	0.94	1.06	29.68	40.22	42.67	0.01	0.01	0.02	34.92	54.71	72.25	0.12	0.20	0.27
Ionosphere	4.52	6.17	6.63	0.66	1.27	1.86	29.28	40.64	42.96	0.01	0.02	0.03	41.00	82.31	149.61	0.18	0.28	0.35
Isolet	145.90	392.35	6123.84	3946.94	6319.62	7868.23	1702.90	2821.20	3642.23	2.65	4.42	6.74	4124.84	7189.68	10124.58	2.10	5.17	7.52
Jm1	277.03	519.39	823.83	21109.16	32177.25	37832.76	99.35	162.93	209.58	0.14	0.24	0.28	21803.14	33476.98	41191.93	0.29	0.44	0.52
Kc1	22.46	24.68	27.68	253.27	275.31	288.70	54.06	56.28	57.78	0.03	0.04	0.05	972.89	1088.48	1117.43	0.26	0.30	0.33
Madelon	346.93	641.01	908.45	89.03	178.73	231.84	410.66	701.83	1088.25	0.78	1.28	2.09	328.22	637.23	1195.81	0.94	1.52	2.43
Musk	101.75	526.90	1377.33	2369.25	4779.14	5710.39	388.72	671.10	970.07	0.48	0.71	1.39	2716.04	5607.58	7681.91	0.54	0.92	1.74
Segment	13.82	24.68	27.00	124.52	246.03	335.96	44.20	57.01	60.40	0.02	0.03	0.03	299.47	656.61	1204.34	0.13	0.21	0.31
Semeion	2.56	6.69	7.83	34.04	62.92	94.60	124.88	221.31	317.19	0.04	0.08	0.18	174.68	361.97	662.25	0.17	0.28	0.36
Sonar	2.87	4.14	4.77	0.29	0.50	0.77	26.10	38.52	42.11	0.01	0.01	0.02	14.26	27.75	41.48	0.12	0.18	0.22
Spambase	479.64	561.52	644.29	1877.57	2111.36	2315.83	154.82	201.16	211.45	0.12	0.16	0.19	2432.10	3428.01	3986.52	0.36	0.42	0.48
Vehicle	4.39	5.70	8.44	7.79	14.26	22.91	32.15	45.42	47.80	0.02	0.02	0.03	105.71	200.36	338.64	0.13	0.20	0.29
Waveform	93.30	166.84	320.31	1101.49	2167.54	2633.88	67.26	116.68	157.02	0.03	0.07	0.08	1357.28	2787.59	4088.92	0.16	0.26	0.35
Wdbc	3.40	4.05	5.08	5.25	6.38	7.13	43.24	45.18	46.89	0.02	0.03	0.04	143.47	227.84	240.07	0.20	0.29	0.30
Yeast	4.42	6.51	7.37	29.65	63.96	95.21	34.93	47.59	49.64	0.01	0.02	0.02	156.56	364.50	661.08	0.13	0.22	0.26

Table 15: The 25% quartile, median and 75% quartile of NTC-τMI's speedup in single-core training time over each baseline per UCI dataset. The values are calculated over the different labeling mechanisms.

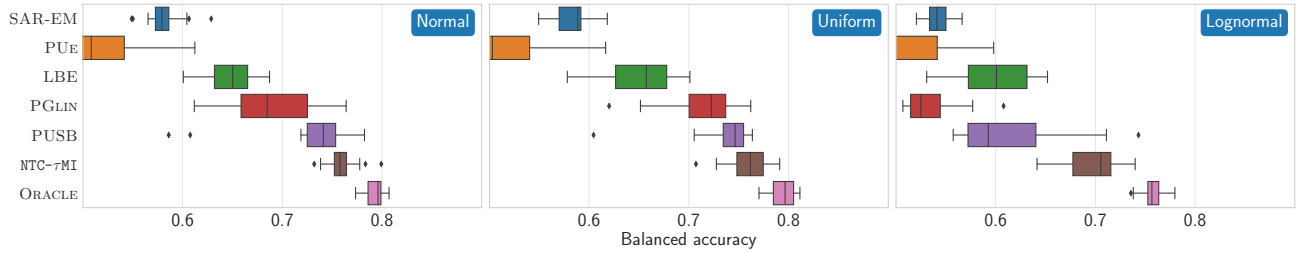
	SAR-EM			PUe			LBE			PGLin			PUSB		
L.S.	25%	Median	75%	25%	Median	75%	25%	Median	75%	25%	Median	75%	25%	Median	75%
Abalone	48.44	56.92	69.47	4104.69	5631.86	6773.71	193.35	223.36	300.77	0.08	0.10	0.12	6420.57	9147.91	10590.78
Banknote	7.93	11.47	16.71	170.26	324.65	441.14	135.72	216.08	270.60	0.03	0.05	0.07	842.48	1832.51	2739.11
Breast-w	9.74	14.07	16.11	47.58	51.06	59.85	158.72	177.38	209.89	0.05	0.06	0.07	782.65	829.05	1088.71
Diabetes	18.48	21.36	24.19	59.65	73.01	86.02	162.67	183.20	217.91	0.05	0.07	0.09	940.40	1095.27	1295.30
Haberman	7.67	9.07	11.20	4.51	5.28	6.36	158.00	201.37	231.47	0.04	0.05	0.06	204.75	292.32	357.16
Heart	9.52	11.51	16.91	3.16	4.05	4.88	152.96	170.60	236.44	0.06	0.07	0.08	202.00	266.62	293.69
Ionosphere	15.82	21.53	23.77	2.82	5.64	6.60	109.85	145.12	158.28	0.05	0.08	0.11	180.08	372.94	524.19
Isolet	43.38	68.58	765.61	785.24	1042.91	1275.68	412.37	490.50	647.89	0.79	0.91	1.02	1024.39	1206.07	1580.56
Jm1	768.72	1204.54	1561.79	47349.56	64874.62	78609.64	291.63	349.24	407.31	0.40	0.50	0.62	49960.26	69219.96	85596.81
Kc1	74.87	84.75	97.85	804.66	912.67	1048.04	171.44	183.29	213.59	0.11	0.15	0.18	3218.94	3496.29	4005.49
Madelon	301.04	374.29	572.72	74.63	100.46	131.21	315.30	397.39	541.14	0.67	0.82	1.02	303.66	403.71	595.15
Musk	63.88	779.80	1388.81	2394.47	3711.28	5459.98	438.23	614.40	799.08	0.67	0.80	0.95	2921.83	4867.86	6607.75
Segment	78.85	93.85	129.38	738.53	1034.64	1197.13	180.33	218.83	282.94	0.09	0.12	0.13	2220.61	2959.14	3970.84
Semeion	10.63	15.60	28.30	135.37	187.12	280.66	519.15	631.33	981.26	0.25	0.30	0.48	833.33	1035.85	1894.91
Sonar	18.32	22.39	25.24	2.46	2.78	3.63	159.49	205.40	226.05	0.06	0.08	0.09	117.31	145.19	182.49
Spambase	1101.57	1400.46	1635.89	4301.09	4901.32	5790.62	399.93	448.25	508.66	0.28	0.37	0.48	6070.96	7858.95	9416.28
Vehicle	21.89	29.19	43.92	50.00	68.08	79.32	154.19	172.13	247.87	0.08	0.11	0.12	652.32	947.67	1161.56
Waveform	496.20	720.10	879.62	4753.05	6943.93	8809.83	377.37	416.06	463.79	0.18	0.22	0.27	7196.99	10272.45	12498.08
Wdbc	12.89	17.63	23.02	22.14	24.00	27.32	146.46	156.99	229.19	0.10	0.12	0.14	602.48	782.11	856.90
Yeast	25.08	29.11	36.67	213.24	288.31	358.03	163.27	215.39	242.22	0.06	0.08	0.11	1155.22	1923.50	2292.11

Table 16: Average ($\pm$ standard deviation) balanced accuracy for all methods on the image datasets under S1-S4.

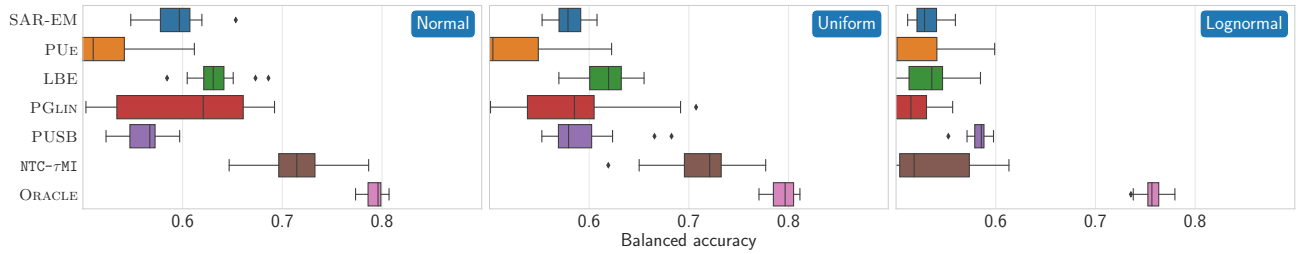
L.S.	Dataset	Oracle	SAR-EM	PUe	LBE	PGLin	PUSB	NTC-τMI
S1	USPS	0.959 ± 0.003	0.762 ± 0.062	0.500 ± 0.000	0.582 ± 0.045	0.701 ± 0.113	0.799 ± 0.066	$\mathbf{0.802 \pm 0.010}$
	MNIST	0.984 ± 0.001	0.743 ± 0.051	0.500 ± 0.000	0.577 ± 0.009	0.606 ± 0.021	$\mathbf{0.929 \pm 0.009}$	0.868 ± 0.010
	FashionMNIST	0.990 ± 0.001	0.862 ± 0.086	0.500 ± 0.000	0.674 ± 0.013	0.689 ± 0.052	$\mathbf{0.970 \pm 0.012}$	0.960 ± 0.004
	CIFAR10	0.921 ± 0.002	0.777 ± 0.053	0.500 ± 0.000	0.500 ± 0.000	0.710 ± 0.045	0.812 ± 0.020	$\mathbf{0.857 \pm 0.004}$
S2	USPS	0.959 ± 0.003	0.758 ± 0.062	0.500 ± 0.000	0.580 ± 0.045	0.700 ± 0.113	0.784 ± 0.102	$\mathbf{0.803 \pm 0.010}$
	MNIST	0.984 ± 0.001	0.743 ± 0.048	0.500 ± 0.000	0.577 ± 0.009	0.606 ± 0.021	$\mathbf{0.926 \pm 0.009}$	0.869 ± 0.010
	FashionMNIST	0.990 ± 0.001	0.846 ± 0.090	0.500 ± 0.000	0.673 ± 0.016	0.686 ± 0.049	$\mathbf{0.971 \pm 0.008}$	0.961 ± 0.003
	CIFAR10	0.921 ± 0.002	0.765 ± 0.055	0.500 ± 0.000	0.500 ± 0.000	0.694 ± 0.026	0.839 ± 0.009	$\mathbf{0.861 \pm 0.004}$
S3	USPS	0.959 ± 0.003	0.781 ± 0.066	0.500 ± 0.000	0.611 ± 0.052	0.715 ± 0.121	$\mathbf{0.854 \pm 0.013}$	0.825 ± 0.008
	MNIST	0.984 ± 0.001	0.746 ± 0.050	0.500 ± 0.000	0.606 ± 0.010	0.641 ± 0.026	$\mathbf{0.928 \pm 0.013}$	0.886 ± 0.009
	FashionMNIST	0.990 ± 0.001	0.878 ± 0.068	0.500 ± 0.000	0.700 ± 0.016	0.771 ± 0.051	$\mathbf{0.965 \pm 0.010}$	0.964 ± 0.002
	CIFAR10	0.921 ± 0.002	0.784 ± 0.056	0.500 ± 0.000	0.500 ± 0.000	0.784 ± 0.031	0.817 ± 0.013	$\mathbf{0.859 \pm 0.004}$
S4	USPS	0.959 ± 0.003	0.701 ± 0.070	0.500 ± 0.000	0.541 ± 0.034	0.646 ± 0.082	$\mathbf{0.776 \pm 0.090}$	0.732 ± 0.019
	MNIST	0.984 ± 0.001	0.748 ± 0.015	0.500 ± 0.000	0.505 ± 0.001	0.503 ± 0.001	$\mathbf{0.867 \pm 0.007}$	0.651 ± 0.015
	FashionMNIST	0.990 ± 0.001	0.853 ± 0.046	0.500 ± 0.000	0.535 ± 0.005	0.506 ± 0.002	$\mathbf{0.941 \pm 0.018}$	0.831 ± 0.020
	CIFAR10	0.921 ± 0.002	0.666 ± 0.049	0.500 ± 0.000	0.500 ± 0.000	0.601 ± 0.051	0.721 ± 0.009	$\mathbf{0.770 \pm 0.013}$



(a) Using labeling mechanism S2.



(b) Using labeling mechanism S3.



(c) Using labeling mechanism S4.

Figure 6: Balanced accuracy for artificial datasets using normal, uniform and lognormal feature distributions under S2, S3, and S4.

Table 17: Balanced accuracy (± standard deviation) for different approaches to setting a threshold on the NTC's output under **S1**. τ`MIcalibration` is our proposed approach, $\hat{\pi}$ estimates the class prior using the KM2 algorithm, and π denotes using the ground-truth class prior. Optimal refers to picking the threshold that maximizes the balanced accuracy on a fully labeled training set.

Dataset	Optimal	τ`MIcalibration`	$\hat{\pi}$	π
Abalone	0.561 ± 0.038	$\mathbf{0.552 \pm 0.040}$	0.536 ± 0.037	0.534 ± 0.029
Banknote	0.982 ± 0.010	0.927 ± 0.018	0.780 ± 0.046	$\mathbf{0.981 \pm 0.010}$
Breast-w	0.955 ± 0.025	0.863 ± 0.034	0.864 ± 0.078	$\mathbf{0.957 \pm 0.016}$
Diabetes	0.710 ± 0.042	$\mathbf{0.709 \pm 0.042}$	0.634 ± 0.048	0.699 ± 0.040
Haberman	0.537 ± 0.051	$\mathbf{0.535 \pm 0.062}$	0.523 ± 0.036	0.527 ± 0.054
Heart	0.732 ± 0.064	$\mathbf{0.742 \pm 0.064}$	0.611 ± 0.063	0.732 ± 0.060
Ionosphere	0.781 ± 0.037	0.761 ± 0.041	0.636 ± 0.057	$\mathbf{0.784 \pm 0.035}$
Isolet	0.918 ± 0.017	0.760 ± 0.006	$\mathbf{0.866 \pm 0.029}$	0.851 ± 0.027
Jm1	0.636 ± 0.019	0.626 ± 0.017	$\mathbf{0.633 \pm 0.018}$	0.615 ± 0.019
Kc1	0.684 ± 0.039	$\mathbf{0.683 \pm 0.035}$	0.675 ± 0.048	0.637 ± 0.038
Madelon	0.516 ± 0.014	$\mathbf{0.518 \pm 0.016}$	0.505 ± 0.009	0.517 ± 0.016
Musk	0.837 ± 0.017	0.767 ± 0.010	0.742 ± 0.035	$\mathbf{0.807 \pm 0.024}$
Segment	0.978 ± 0.013	0.792 ± 0.013	0.857 ± 0.068	$\mathbf{0.973 \pm 0.018}$
Semeion	0.856 ± 0.044	0.762 ± 0.022	0.742 ± 0.079	$\mathbf{0.796 \pm 0.047}$
Sonar	0.595 ± 0.083	0.573 ± 0.096	0.545 ± 0.055	$\mathbf{0.578 \pm 0.089}$
Spambase	0.850 ± 0.021	0.834 ± 0.014	0.822 ± 0.028	$\mathbf{0.841 \pm 0.026}$
Vehicle	0.863 ± 0.050	0.804 ± 0.026	0.691 ± 0.039	$\mathbf{0.846 \pm 0.060}$
Waveform	0.835 ± 0.012	$\mathbf{0.833 \pm 0.011}$	0.832 ± 0.013	0.797 ± 0.017
Wdbc	0.820 ± 0.051	0.789 ± 0.052	0.810 ± 0.045	$\mathbf{0.818 \pm 0.044}$
Yeast	0.656 ± 0.031	$\mathbf{0.654 \pm 0.034}$	0.565 ± 0.041	0.608 ± 0.040

Table 18: Balanced accuracy (± standard deviation) for different approaches to setting a threshold on the NTC's output under **S2**. τ`MIcalibration` is our proposed approach, $\hat{\pi}$ estimates the class prior using the KM2 algorithm, and π denotes using the ground-truth class prior. Optimal refers to picking the threshold that maximizes the balanced accuracy on a fully labeled training set.

Dataset	Optimal	τ`MIcalibration`	$\hat{\pi}$	π
Abalone	0.542 ± 0.027	0.505 ± 0.040	0.510 ± 0.017	$\mathbf{0.513 \pm 0.022}$
Banknote	0.983 ± 0.009	0.930 ± 0.017	0.806 ± 0.066	$\mathbf{0.981 \pm 0.008}$
Breast-w	0.966 ± 0.014	0.880 ± 0.028	0.853 ± 0.056	$\mathbf{0.957 \pm 0.019}$
Diabetes	0.707 ± 0.041	$\mathbf{0.708 \pm 0.039}$	0.593 ± 0.057	0.693 ± 0.051
Haberman	0.541 ± 0.061	0.538 ± 0.053	0.519 ± 0.038	$\mathbf{0.552 \pm 0.062}$
Heart	0.758 ± 0.050	0.750 ± 0.048	0.587 ± 0.067	$\mathbf{0.770 \pm 0.050}$
Ionosphere	0.778 ± 0.056	0.748 ± 0.073	0.600 ± 0.066	$\mathbf{0.772 \pm 0.055}$
Isolet	0.927 ± 0.016	0.759 ± 0.006	$\mathbf{0.849 \pm 0.031}$	0.829 ± 0.030
Jm1	0.625 ± 0.015	$\mathbf{0.621 \pm 0.014}$	0.614 ± 0.018	0.607 ± 0.013
Kc1	0.683 ± 0.032	$\mathbf{0.678 \pm 0.036}$	0.623 ± 0.046	0.644 ± 0.043
Madelon	0.527 ± 0.020	$\mathbf{0.527 \pm 0.021}$	0.511 ± 0.008	0.525 ± 0.022
Musk	0.813 ± 0.012	0.751 ± 0.015	0.716 ± 0.034	$\mathbf{0.798 \pm 0.018}$
Segment	0.976 ± 0.011	0.790 ± 0.014	0.862 ± 0.083	$\mathbf{0.968 \pm 0.018}$
Semeion	0.825 ± 0.035	0.750 ± 0.023	0.707 ± 0.053	$\mathbf{0.782 \pm 0.041}$
Sonar	0.595 ± 0.074	$\mathbf{0.576 \pm 0.082}$	0.546 ± 0.051	0.571 ± 0.078
Spambase	0.861 ± 0.016	0.837 ± 0.011	0.782 ± 0.045	$\mathbf{0.857 \pm 0.017}$
Vehicle	0.855 ± 0.044	0.800 ± 0.038	0.666 ± 0.049	$\mathbf{0.841 \pm 0.045}$
Waveform	0.835 ± 0.010	$\mathbf{0.830 \pm 0.009}$	0.824 ± 0.016	0.798 ± 0.014
Wdbc	0.843 ± 0.042	0.813 ± 0.049	0.811 ± 0.066	$\mathbf{0.846 \pm 0.039}$
Yeast	0.633 ± 0.032	$\mathbf{0.635 \pm 0.034}$	0.529 ± 0.023	0.605 ± 0.031

Table 19: Balanced accuracy (± standard deviation) for different approaches to setting a threshold on the NTC's output under **S3**. τMIcalibration is our proposed approach, $\hat{\pi}$ estimates the class prior using the KM2 algorithm, and π denotes using the ground-truth class prior. Optimal refers to picking the threshold that maximizes the balanced accuracy on a fully labeled training set.

Dataset	Optimal	τMIcalibration	$\hat{\pi}$	π
Abalone	0.593 ± 0.026	$\mathbf{0.588 \pm 0.025}$	0.586 ± 0.028	0.557 ± 0.025
Banknote	0.983 ± 0.011	0.918 ± 0.020	0.838 ± 0.062	$\mathbf{0.980 \pm 0.012}$
Breast-w	0.962 ± 0.011	0.867 ± 0.022	0.894 ± 0.053	$\mathbf{0.956 \pm 0.014}$
Diabetes	0.713 ± 0.041	$\mathbf{0.714 \pm 0.038}$	0.657 ± 0.056	0.701 ± 0.047
Haberman	0.549 ± 0.059	$\mathbf{0.555 \pm 0.057}$	0.530 ± 0.055	0.547 ± 0.069
Heart	0.725 ± 0.077	0.716 ± 0.089	0.619 ± 0.066	$\mathbf{0.716 \pm 0.088}$
Ionosphere	0.801 ± 0.054	0.787 ± 0.049	0.663 ± 0.080	$\mathbf{0.807 \pm 0.050}$
Isolet	0.926 ± 0.014	0.758 ± 0.007	$\mathbf{0.868 \pm 0.032}$	0.844 ± 0.034
Jm1	0.636 ± 0.015	$\mathbf{0.627 \pm 0.013}$	0.627 ± 0.011	0.616 ± 0.014
Kc1	0.692 ± 0.034	$\mathbf{0.688 \pm 0.038}$	0.687 ± 0.035	0.631 ± 0.035
Madelon	0.516 ± 0.019	$\mathbf{0.521 \pm 0.020}$	0.506 ± 0.011	0.520 ± 0.019
Musk	0.846 ± 0.015	0.772 ± 0.013	0.772 ± 0.020	$\mathbf{0.818 \pm 0.013}$
Segment	0.971 ± 0.023	0.785 ± 0.014	0.892 ± 0.057	$\mathbf{0.970 \pm 0.022}$
Semeion	0.854 ± 0.029	0.763 ± 0.020	0.781 ± 0.072	$\mathbf{0.794 \pm 0.046}$
Sonar	0.634 ± 0.071	$\mathbf{0.647 \pm 0.094}$	0.574 ± 0.052	0.639 ± 0.096
Spambase	0.855 ± 0.017	0.836 ± 0.014	0.838 ± 0.021	$\mathbf{0.846 \pm 0.022}$
Vehicle	0.876 ± 0.035	0.797 ± 0.025	0.744 ± 0.061	$\mathbf{0.862 \pm 0.030}$
Waveform	0.839 ± 0.011	0.831 ± 0.008	$\mathbf{0.839 \pm 0.010}$	0.803 ± 0.015
Wdbc	0.849 ± 0.065	0.813 ± 0.049	0.825 ± 0.078	$\mathbf{0.848 \pm 0.059}$
Yeast	0.672 ± 0.024	$\mathbf{0.671 \pm 0.026}$	0.604 ± 0.042	0.630 ± 0.031

Table 20: Balanced accuracy (± standard deviation) for different approaches to setting a threshold on the NTC's output under **S4**. τMIcalibration is our proposed approach, $\hat{\pi}$ estimates the class prior using the KM2 algorithm, and π denotes using the ground-truth class prior. Optimal refers to picking the threshold that maximizes the balanced accuracy on a fully labeled training set.

Dataset	Optimal	τMIcalibration	$\hat{\pi}$	π
Abalone	0.630 ± 0.027	$\mathbf{0.637 \pm 0.026}$	0.633 ± 0.026	0.579 ± 0.018
Banknote	0.865 ± 0.064	0.850 ± 0.068	0.603 ± 0.038	$\mathbf{0.853 \pm 0.074}$
Breast-w	0.923 ± 0.029	0.838 ± 0.031	0.805 ± 0.084	$\mathbf{0.921 \pm 0.032}$
Diabetes	0.719 ± 0.033	$\mathbf{0.721 \pm 0.039}$	0.710 ± 0.038	0.709 ± 0.030
Haberman	0.568 ± 0.071	0.546 ± 0.067	$\mathbf{0.572 \pm 0.078}$	0.570 ± 0.079
Heart	0.670 ± 0.076	$\mathbf{0.667 \pm 0.077}$	0.616 ± 0.088	0.661 ± 0.092
Ionosphere	0.780 ± 0.050	0.752 ± 0.049	0.638 ± 0.062	$\mathbf{0.772 \pm 0.063}$
Isolet	0.896 ± 0.025	0.760 ± 0.008	$\mathbf{0.850 \pm 0.050}$	0.805 ± 0.039
Jm1	0.639 ± 0.015	$\mathbf{0.627 \pm 0.012}$	0.620 ± 0.013	0.617 ± 0.013
Kc1	0.707 ± 0.023	0.700 ± 0.023	$\mathbf{0.704 \pm 0.023}$	0.648 ± 0.032
Madelon	0.519 ± 0.017	0.517 ± 0.022	0.509 ± 0.014	$\mathbf{0.520 \pm 0.023}$
Musk	0.845 ± 0.017	0.772 ± 0.011	0.758 ± 0.035	$\mathbf{0.788 \pm 0.023}$
Segment	0.957 ± 0.030	0.788 ± 0.013	0.851 ± 0.099	$\mathbf{0.944 \pm 0.045}$
Semeion	0.832 ± 0.044	0.754 ± 0.034	0.713 ± 0.056	$\mathbf{0.774 \pm 0.059}$
Sonar	0.621 ± 0.090	0.620 ± 0.091	0.572 ± 0.076	$\mathbf{0.623 \pm 0.096}$
Spambase	0.711 ± 0.026	$\mathbf{0.701 \pm 0.029}$	0.592 ± 0.025	0.664 ± 0.020
Vehicle	0.857 ± 0.037	0.797 ± 0.029	0.738 ± 0.059	$\mathbf{0.840 \pm 0.050}$
Waveform	0.837 ± 0.009	0.830 ± 0.008	$\mathbf{0.831 \pm 0.009}$	0.799 ± 0.013
Wdbc	0.573 ± 0.059	0.533 ± 0.070	0.540 ± 0.047	$\mathbf{0.559 \pm 0.070}$
Yeast	0.686 ± 0.031	$\mathbf{0.685 \pm 0.028}$	0.653 ± 0.037	0.631 ± 0.031