# Positive-Unlabeled Learning in the Face of Labeling Bias

Noah Youngs
*CY Data Science*
*New York, USA*
*Simons Center For Data Analysis*
*New York, USA*
*nyoungs@cydatascience.com*

Dennis Shasha
*Dep.of Computer Science, NYU*
*New York, USA*
*shasha@nyu.edu*

Richard Bonneau
*Dep. of Computer Science, NYU*
*New York, USA*
*CGSB, Dep. of Biology, NYU*
*New York, USA*
*Simons Center For Data Analysis*
*New York, USA*
*bonneau@nyu.edu*

***Abstract*—Positive-Unlabeled (PU) learning scenarios are a class of semi-supervised learning where only a fraction of the data is labeled, and all available labels are positive. The goal is to assign correct (positive and negative) labels to as much data as possible. Several important learning problems fall into the PU-learning domain, as in many cases the cost and feasibility of obtaining negative examples is prohibitive. In addition to the positive-negative disparity the overall cost of labeling these datasets typically leads to situations where the number of unlabeled examples greatly outnumbers the labeled. Accordingly, we perform several experiments, on both synthetic and real-world datasets, examining the performance of state of the art PU-learning algorithms when there is significant bias in the labeling process. We propose novel PU algorithms and demonstrate that they outperform the current state of the art on a variety of benchmarks. Lastly, we present a methodology for removing the costly parameter-tuning step in a popular PU algorithm.**

## 1. Introduction

With the number and size of uncurated datasets growing exponentially, more and more learning scenarios provide few labeled examples. Labeling data is often either time-consuming, or costly in terms of domain-expertise required to correctly perform the task. In addition, there are a number of contexts in which labeling the positive class is easier than the negative. For example, when tagging documents with topics it is far easier to curate a database of tagged topics, than to ask a user to list all topics a document is *not* about. While one could assume that the absence of a topic implies a negative example, this is only correct if the user tagging the document was provided with a *complete* set of topics before the tagging process. If an additional topic is added to the topic set, the lack of an annotation for that topic in previously labeled documents can no longer be equated with a negative example. This scenario also extends quite easily to image tagging, and several other important learning problems. This particular branch of semi-supervised learning, where only a fraction of examples are labeled and those labels consist only of the positive class, is known as positive-unlabeled learning (PU learning).

Algorithms and heuristics have both been applied to PU scenarios in diverse fields, including: text classification [7], [10], [15], protein function prediction [12], [13], [16], gene identification [14], data stream classification [5], remote-sensing [4], novelty detection [1], and many more. Despite the proliferation of PU scenarios, the problem has not been generalized until fairly recently. In the work of Elkan and Noto [2], a rigorous theoretical framework is presented which allows for PU scenarios to be formalized independently of their domain.

We continue the general treatment of the PU problem, propose a novel framework that leads to a new algorithm, and investigate the effect of biased labeling. Due to the expense of the labeling process, efforts are often made to maximize the number of positive examples, by examining unlabeled examples that are highly correlated with already-found positive examples. While this can be a way to cheaply grow the size of the positive label set, it also introduces significant bias into labeling process, which can have a negative impact on the performance of many existing PU algorithms. In addition to our novel algorithms and bias analysis, we adapt an existing state of the art PU algorithm, Biased SVM [7]removing the need for the computationally intensive tuning process.

## 2. Background and Related Work

The seminal work of Liu et al. [7], formalized PU learning as a specific and noteworthy form of semi-supervised learning. The work compared several existing PU algorithms in the context of text classification, as well as proposed a novel algorithm. This algorithm, Biased Support Vector Machines (Biased SVM), demonstrated superior performance over existing algorithms on several sample datasets.

Later work by Elkan and Noto [2] compared Biased SVM to their own novel algorithms, based on an important theoretical result: Elkan and Noto showed that the decision boundary of a classifier trained to differentiate between positive and unlabeled examples will produce predictions

978-1-4673-8493-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDMW.2015.207

that obey the same rank ordering as a traditional classifier that attempts to discriminate between positive and negative examples. Additionally, the decision threshold of such a classifier can be transformed, such that its accuracy is also in-line with a traditional classifier [2]. This result forms the basis for two algorithms: one in which a classifier is trained to discriminate between positive and unlabeled examples and then transformed, and another in which a similar methodology is used to calculate class probabilities, and then all unlabeled examples are duplicated in the training set, with each replicate weighted according to the calculated class membership probability, and passed to a classifier that can learn on weighted data.

Elkan and Noto, however, note that their work relies on one crucial assumption: that the set of labeled positive examples is chosen uniformly at random from the set of all positive examples. While this assumption might appear innocuous, in many real-world PU applications, it is certainly violated. In protein function prediction, for example, annotations are often propagated via homology, a method that compares the sequence of an unknown protein to the sequences of known proteins. Thus the labeled positives of a function are not selected at random from all proteins with that function, but rather according to a bias based on similarity to the first proteins labeled with that function. It is not difficult to imagine other problem scenarios, such as text classification, where documents are given to curators based on search queries (such as a keyword-filtered subset of Twitter obtained using the Twitter streaming API), and thus the set of labeled positive examples are again biased rather than random.

## 2.1. Assumptions and Notation

We adopt the theoretical framework used in Elkan and Noto [2]. Let $x$ be an example with a binary label $y \in \{0,1\}$. Let $s$ be a second binary label for $x$, which indicates whether the value of $y$ is known. Thus the traditional learning goal can be stated as trying to compute: $p(y=1 \mid x, s=0)$, which is the probability that a given example is a positive example ($y=1$), given its features $x$, and that it is currently unlabeled ($s=0$). Since only positive examples are labeled in a PU scenario, we have the following axioms:

$$\begin{aligned} s=1 &\rightarrow y=1 \\ s=0 &\rightarrow y=1 \quad or \quad y=0 \\ p(s=1 \mid x, y=0) &= 0 \end{aligned} \tag{1}$$

Using this framework we break down our training examples into three sets: $P$, $Q$, and $N$, where $P$ are the positive examples ($y=1, s=1$), $Q$ are the unlabeled positive examples ($y=1, s=0$), and $N$ are the negative examples ($y=0, s=0$). Thus any algorithm attempting to learn in a PU scenario has access to the value of $s$ for all training examples, but only to the value of $y$ for the examples that are labeled ($s=1$). It is also natural to refer to the labeled and unlabeled subsets of examples, L and U. With the framework defined above, L = P (we use the term P in this work) , and U = N + Q.

The assumption stated in section 2, that the labeled examples are chosen uniformly at random from the set of positive examples, can be expressed by the following:

$$p(s=1 \mid x, y=1) = p(s=1 \mid y=1) = c \tag{2}$$

The probability that any given positive example is labeled is constant, regardless of its features $x$. In other words, knowing the features of a positive example will not improve your estimation of the probability of that example being labeled.

Elkan and Noto then propose to train a classifier $g(x) = p(s=1 \mid x)$, which can be related to $f(x) = p(y=1 \mid x)$ by the equation: $g(x) = cf(x)$. The authors also present several techniques for estimating $c$ from observed data, all of which also rely on the "selected at random" assumption stated above.

## 3. Novel Algorithms and Analysis

### 3.1. Biased Labeling Analysis

When the process for labeling a new positive example is at all dependent on the set of currently labeled examples, the "selected at random" assumption is no longer valid and the probability of a positive example being labeled is now some function of that example itself: $h(x) = p(s=1|y=1,x)$. The result of Elkan and Noto [2] now becomes:

$$\begin{aligned} g(x) &= h(x)f(x) \\ f(x) &= g(x))/(h(x) \end{aligned} \tag{3}$$

This new relation between f(x) and g(x) relies on the ability to estimate h(x), and the ranking assumption no longer holds, unless there are specific conditions placed upon h(x). Namely:

$$\begin{aligned} f(x_1) \geq f(x_2) \quad &\geq f(x_3) \leftrightarrow g(x_1) \geq \quad g(x_2) \geq g(x_3) \\ &iff \\ h(x_1) \geq \quad &h(x_2) \quad \geq h(x_3) \end{aligned} \tag{4}$$

This result states that the ranking assumption only holds if the labeling bias is identically ranked, i.e. if an example $i$ is more likely to be positive than example $j$ then $i$ must also more likely to be labeled than $j$. This is a reasonable restriction, as it is not difficult to imagine a scenario in which new examples are labeled according to their similarity to existing examples (as is often the case in bioinformatics), but the restriction is worth noting.

### 3.2. A Novel PU Algorithm

We use a Gaussian-based formalization of PU learning in our algorithm. Given $v$ training examples of dimension $d$, and remembering that each example $x$ in the training set belongs to one of the three sets: $P$ (the labeled positive examples), $Q$ (the unlabeled positive examples), and $N$ (the negative examples), we desire a function $D$, which satisfies:

$$\begin{aligned} D &: x \rightarrow \Re^l \quad (l < d) \quad s.t. && (5)\\ D(x_P) &\sim \mathcal{N}(\mu_P, \sigma_P) && (a)\\ D(x_N) &\sim \mathcal{N}(\mu_N, \sigma_N) && (b) \end{aligned}$$

That is to say, a function $D$ that maps the feature space from its original dimension, $d$, into a lower-dimensional space of dimension $l$. The mapping should also be such that when it is applied to the features of positive examples, $X_P$, the results appear normally distributed with mean $\mu_P$ and standard deviation $\sigma_P$, and when applied to features of the negative class, $X_N$, produces data distributed normally with mean $\mu_N$ and standard deviation $\sigma_N$. The more distinct the parameters are for each class, the more discriminative and therefore useful the particular mapping $D$ will be.

Condition $(a)$ allows us to estimate $\mu_P$ and $\sigma_P$, since we have a set of labeled positive examples in the training data, but $(b)$ implies we cannot easily estimate $\mu_N$ and $\sigma_N$, as all we have is a set of examples drawn from the distribution: $\mathcal{N}(\mu_U, \sigma_U) = \mathcal{N}(\mu_N, \sigma_N) + \mathcal{N}(\mu_Q, \sigma_Q)$, where $Q$ is the set of true positives in the unlabeled set. That is to say, we can estimate the parameters of the distribution on the unlabeled training examples, but the results will be a mixture of Gaussians, where the bias away from $\mathcal{N}(\mu_N, \sigma_N)$ will be greater when more true positives are included in the unlabeled set (the greater the size of $Q$ relative to $N$).

Once a function $D$ is chosen, estimates are obtained for parameters of the positive and unlabeled distributions. For each example, the class membership probability is computed, which is the probability that the example belongs to $P$, or to our proxy for $N$ (which is trained on $U = N + Q$), via another function $A$. We assume for simplicity that $\mu_P > \mu_Q$, but if the choice of $D$ implies the opposite, the following discussion still applies by reversing the necessary inequalities.

Once the probabilities have been approximated, they can be used as weights during the training of a classifier, whereby all unlabeled training examples are duplicated, with each replicate given weight equal to its class membership probability [2].

In summary, given a training set of $v$ examples with $d$ features, our approach utilizes $D$ to map the original features $X$ into a smaller-dimensional, but more separable space $Z$, $A$ to calculate class membership probabilities, and then a weighting scheme $W$ to transform $X$ into $X_w$, a final weighted set of training examples, and to create a corresponding label vector, $\vec{\lambda}$.

$$\begin{aligned} D &: X^{v\times d} \rightarrow Z^{v\times l} && (6)\\ A &: Z^{v\times l} \rightarrow \vec{\rho}, \qquad where \quad \vec{\rho} = p(y=1 \mid X) &&\\ W &: X^{v\times d}, \vec{\rho} \rightarrow X_w^{g\times d}, \vec{\lambda} \in \Re^g && \end{aligned}$$

**3.2.1. Choice of the $D$ Function: SNOB.** For the $D$ function we choose an algorithm already shown to have discriminatory power in PU scenarios: SNOB [12].

SNOB was originally formulated in terms of discrete, binary features, but the extension to continuous features is straightforward. Given $v$ training examples with $d$ features, represented by the matrix $X^{v\times d}$, we compute a co-feature-mass vector $\vec{\gamma} \in \Re^d$, such that:

$$\vec{\gamma}_j = \frac{\sum_{x_i \in P} X_{i,j}}{\sum_i X_{i,j}} \qquad (7)$$

We then calculate $D(X) = X\vec{\gamma}$, and normalize by the row sums of $X$.

In the case of binary features, the value of $D(X)$ is easily interpretable as the average empirical probability of an example being positive, given its features. Each entry in the $\vec{\gamma}$ vector, $\vec{\gamma}_j$, is the number of times a positive example has feature $j$, divided by the total number of times that any example has feature $j$ (the empirical probability of an example being positive conditional on feature $j$). Adding together the $\vec{\gamma}$ values for which a particular example has features, and dividing by the number of features for that example, as indicated, yields the average empirical probability: $p(y = 1|x)$. For continuous features, the procedure is the same, though the probabilistic interpretation does not hold.

**3.2.2. Choice of the $A$ Function: Monte Carlo Probability Approximation.** In order to map $D$ into probability space, we utilize an $A$ function based on Monte Carlo approximation. First, we generate $n$ points from the distribution $\mathcal{N}(\mu_P, \sigma_P)$, and $n$ points from the distribution $\mathcal{N}(\mu_U, \sigma_U)$. An estimate for the probability that $x$ is in the positive class is then given by:

$$A(z) = p(y=1|x) \sim \frac{\sum_{i=1}^n \imath(k_i; z)}{\sum_{i=1}^n \imath(k_i; z) + \sum_{i=1}^n \jmath(l_i; z)} \qquad (8)$$

Where $k_i$ is the $\imath$th Monte Carlo point generated from $\mathcal{N}(\mu_P, \sigma_P)$, $l_i$ is the ith Monte Carlo point generated from $\mathcal{N}(\mu_U, \sigma_U)$, $z = D(x)$, and $\imath(m; x)$ and $\jmath(m; x)$ are indicator functions such that:

$$\imath(m; l) = \begin{cases} 1 & : m < z \\ 0 & : m \geq z \end{cases}$$

$$\jmath(m; l) = \begin{cases} 1 & : m > z \\ 0 & : m \leq z \end{cases}$$

Intuitively, if $z$ (the value of our mapping applied to the features of a particular test example, $x$) were so large that it was greater than all Monte Carlo points generated from both $\mathcal{N}(\mu_P, \sigma_P)$ and $\mathcal{N}(\mu_U, \sigma_U)$ the approximate probability of $x$ coming from the positive class would be: $p(y = 1 \mid x) = \frac{n}{n+0} = 1$. Conversely, if $z$ were so small that it was less than all Monte Carlo points generated from both $\mathcal{N}(\mu_P, \sigma_P)$ and $\mathcal{N}(\mu_U, \sigma_U)$ the approximate probability of $x$ coming from the positive class would be: $p(y = 1 \mid x) = \frac{0}{0+n} = 0$. Finally, if $z$ were larger than exactly half of the Monte Carlo points generated from both distributions, then the approximate probability of $x$ coming from the positive class would be: $p(y = 1 \mid x) = \frac{\frac{n}{2}}{\frac{n}{2}+\frac{n}{2}} = 0.5$.

### 3.3. Biased SVM without Tuning

The Biased SVM algorithm proposed in [7] demonstrates strong performance on many data sets, but relies on a tuning process to calculate values for the parameters $C_P$ and $C_U$, which control the difference in cost between misclassifying positive and unlabeled examples in the SVM. This tuning process requires hundreds of calculations of SVMs on a subset of the training set in order to find the optimal pair of parameters, which is a computationally intensive process, and problematic when the training set is small and difficult to further subdivide. We provide an alternative to this tuning process, utilizing a Gaussian Mixture Model (GMM) applied to the output of SNOB (see section 3.3.1).

The inspiration for the Biased SVM algorithm was the issue of class imbalance [9], in which the two cost parameters were set such that:

$$\frac{C_+}{C_-} = \frac{|P| + |Q|}{|N|} \tag{9}$$

In a PU scenario, the number of true positives and negatives are unknown (due to the fact that we only know $|P|$ and $|U|$, but $|U| = |Q| + |N|$), but we can use our class probabilities to calculate $E(|P|+|Q|)$ and $E(|N|)$. This is done by simply summing up the probabilities of belonging to each class, for all unlabeled examples, and then adding to $E(|P|)$ the number of labeled positive examples. We then set the cost parameters:

$$\begin{aligned} C_P &= \frac{2E(|N|)}{E(|P|+|Q|) + E(|N|)} \qquad (10)\\ C_U &= \frac{E(|P|+|Q|)(1-C^p) + E(|N|)}{E(|N|)} \end{aligned}$$

So that the expected value of the ratio of cost parameters is the desired class imbalance ratio, and the average misclassification cost is equal to 1.

**3.3.1. Probability Inference via Mixture Models.** Assuming that we can find a function $D$ that satisfies the desired properties specified in Equation 5, the result of applying $D$ to the underlying data will be a mixture of two Gaussians. Thus we can train a Gaussian Mixture Model using the EM-algorithm [8]. We seed the GMM algorithm with the labeled positive examples as the initial members of one component and the unlabeled examples as the members of the other, with corresponding initial mixing proportions. It would also be logical to restrict the labeled positive examples to remaining in the positive component for the duration of the algorithm, but we leave this modification for future work.

Once we have estimates for $(\mu_P, \sigma_P)$ and $(\mu_N, \sigma_N)$, we obtain the probability that a particular $x$ was generated from each of the two distributions. The final probability that $x$ belongs to the positive class must also take into account the mixing proportions, and is given by:

$$A(z) = p(y=1 \mid x) = \frac{\varphi(l;\mu_P,\sigma_P)\pi_P}{\varphi(l;\mu_P,\sigma_P)\pi_P + \varphi(l;\mu_N,\sigma_N)\pi_N} \tag{11}$$

Where $\varphi(x;\mu,\sigma)$ is the pdf of a normal distribution with mean $\mu$ and $\sigma$, and $\pi$ is the mixing proportion of a component.

## 4. Experiments and Results

We focus our experiments on learning scenarios in which $|P| << |Q|$ , as this the most common case in real-world data where the source of the PU scenario is due to the cost of labeling training examples. We examine the impact of bias in label choice via a synthetic example, as well as the performance of several algorithms on three real-world datasets.

The structure of our formalism allows for several combinatorial variations of algorithms, using potentially different choices of $D$ and different methods of computing class probabilities (choices of $A$). We tested many such combinations, but for clarity present only the results from one algorithm, which we refer to as SNOB_MC_Double. In addition, we present results from our Tuneless Biased SVM (TBSVM) using a Mixture Model trained on the output of SNOB.

Alongside our algorithms, we compare the Biased SVM method of [7] (BSVM), and the two algorithms proposed in [2] (EN_Single and EN_Double).

All algorithms use a SVM with linear kernel for the classification task, and Average F Score and AUC ROC values are calculated using 5-fold cross-validation.

### 4.1. Synthetic Data with Controlled Bias

In order to test the effect of bias on the PU learning scenario, we create 2-dimensional synthetic data from two Gaussian distributions. We generate positive and negative examples from distributions with mean $\mu_P = [4, 8]$, $\mu_N = [1, 5]$, and covariance matrices:

$$\Sigma_P = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix} \qquad \Sigma_N = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 2.5 \end{pmatrix}$$

.

We generate $|P| = 100$ labeled positive examples, $|Q| = 400$ unlabeled positive examples, and $|N| = 500$ negative examples. Additionally, when selecting the labeled positive examples from the set of all true positives, we either select uniformly at random, or according to a bias function: $bias(x) = \sqrt{x_1^2 + x_2^2}$. Thus examples which are closer to the origin are selected first, and since this direction is roughly perpendicular to the decision boundary between positive and negative values, creates a scenario in which the probability of being labeled moves *oppositely* to the probability of being positive. In order to explore the effect of differing amount of bias, we choose some labeled examples at random, and others according to the bias function. For example, if $|P|$ = 100, and we set the bias amount to 40%, then 60 of the labeled positive examples will be chosen uniformly at random from the set of all true positives, while the remaining 40 will be chosen according to the bias function described above. Results are presented in Figure 1.

For most algorithms, we see degradation of performance

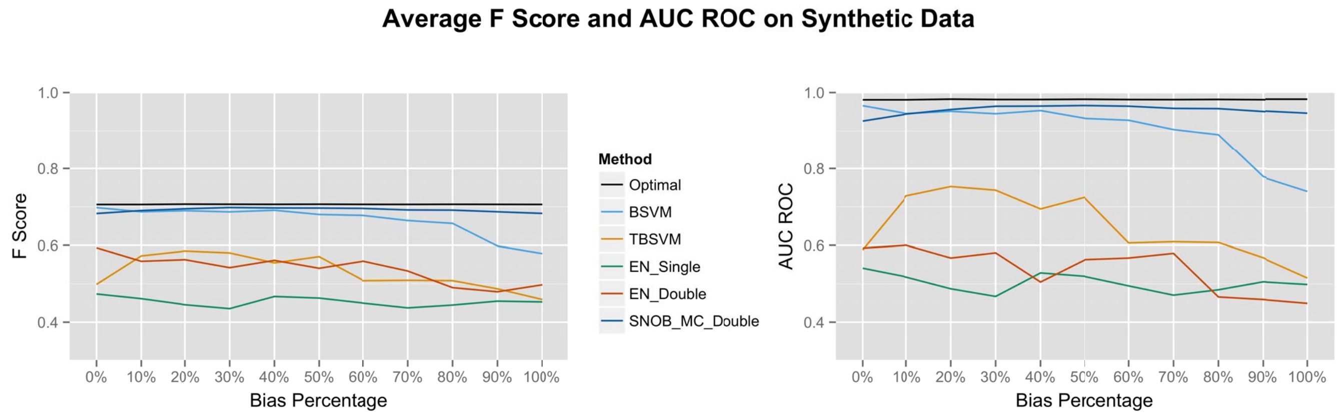

Figure 1. Results on synthetic data generated from two-dimensional Gaussians, with increasing amounts of label bias.

as the percentage of bias labels increases, albeit a slight one. Other than at 0% bias our SNOB_MC_Double performs the best in terms of both average F Score and AUC ROC, with BSVM taking a close second, followed by TBSVM (and also EN_Double in terms of average F Score).

## 4.2. Real-World Datasets

**4.2.1. TCBD Database.** The TCBD Database [11], used as the original benchmark for the algorithms presented Elkan and Noto, consists of text documents representing proteins. The set includes 2453 labeled positive examples, 348 unlabeled positive examples, and 4558 negative examples. We test on the original dataset, as well on the dataset with P and Q exchanged.

In addition, we examine the effects of label bias from a source external to the dataset. We choose positive examples as proteins from one species, or a combination of species, and let the rest of the positive examples go unlabeled. We repeat this process 5 times, with positive-label species chosen so that the number of labeled positive examples ranges from 54 - 1527. These species-biased results are then averaged together, with results are presented in Figure 2.

The original TCBD task was not a difficult classification, with most algorithms achieving very high F Score and AUC ROC. When the labeled and unlabeled positives are reversed, however, the task becomes much more difficult (now only 5% of the data is labeled, as opposed to the original 33%). In this case, our SNOB_MC_Double algorithm performs markedly better than all other algorithms in both F Score and AUC ROC, followed by BSVM and TBSVM.

**4.2.2. Internet Advertisements Dataset.** The Internet Advertisments Dataset was obtained from the UCI machine learning repository [6], and first appeared in [3]. The dataset consists of 3 real-valued features, which have been removed, and 1555 categorical features. The data contains 458 positive examples (advertisements), 20% of which were labeled by choosing a random seed positive example, and then labeling other examples that were most highly correlated with the seed. This process was repeated 10 times with 10 different seed advertisements, and the results averaged. The remaining 366 positive examples were included with the 2821 negative examples (non-advertisements) as the unlabeled set. For the unbiased scenario, we again labeled 20% of the true positives, but this time by selecting uniformly at random from all true positives (again repeating the random selection 10 times). Results are presented in Figure 3.

Once again SNOB_MC_Double outperforms all other algorithms by both measures, with 2nd place going to EN_Double in the biased labeling scenario, and BSVM in the unbiased scenario. It is also interesting to note that TBSVM outperforms BSVM in the biased labeling scenario, but the reverse is true in the unbiased scenario.

**4.2.3. Reuters Dataset.** We also perform tests on the ubiquitous Reuters-21578 dataset, which contains 21578 Reuters newswire documents tagged with 135 topics. We test on only the 10 most populated categories, creating 10 learning scenarios. For each topic, all documents without that topic tag are treated as negative. We once again seed with a random positive example, and choose the rest of the labels via true positives with the highest correlated feature vector to the seed positive. We choose P to have size equal to 20% of the true positives, with the rest of the true positives and the negatives forming the unlabeled set for the algorithms to train on. Results are presented in Table 1.

SNOB_MC_Double has the highest F Score and AUC ROC in nearly every topic category, and the highest average values of both, with the performance especially strong in

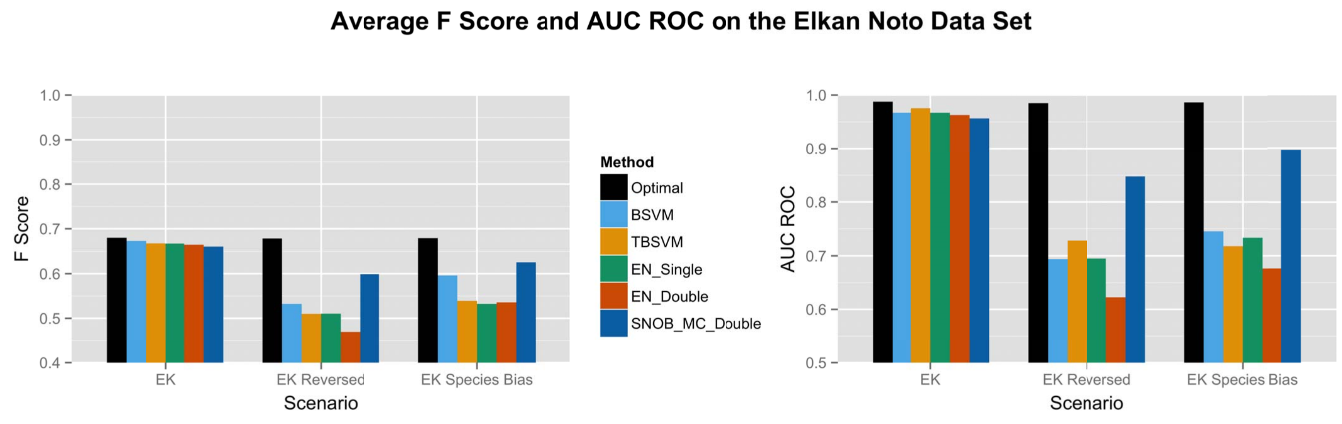

Figure 2. Elkan Noto TCBD Data Set Results, showing average F Score and AUC ROC results on the original data split, a split with the P and Q sets reversed, and then an average of 5 splits chosen with label-bias.

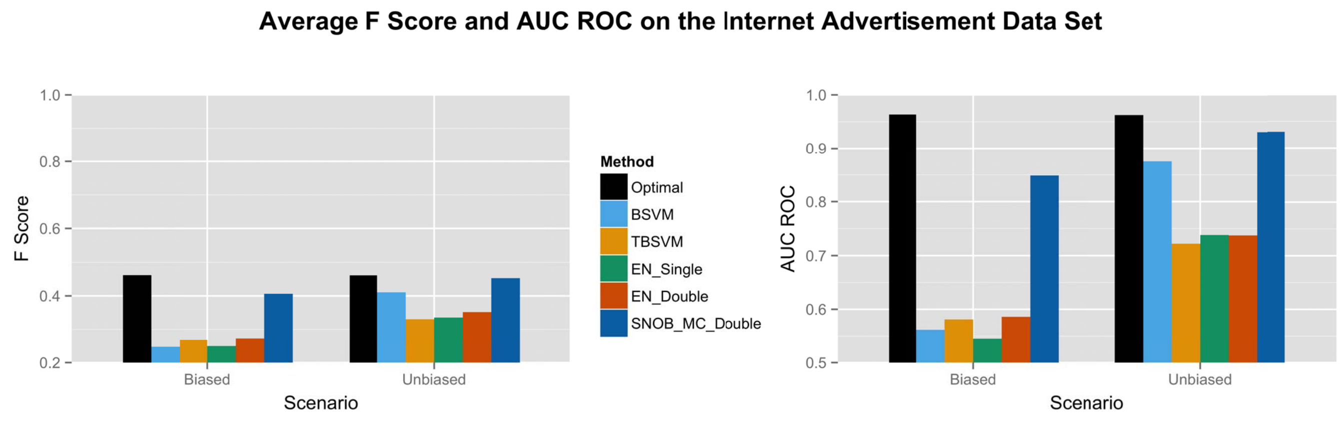

Figure 3. IA Database Results, showing accuracy percentage and AUC_ROC results on a scenario with labeling bias.

terms of AUC ROC. EN_Double ranks 2nd, outperforming SNOB_MC_Double in only two topics in terms of AUC ROC and only one in terms of F Score. The other three algorithms perform almost identically on average (including BSVM and TBSVM, despite the absence of the costly tuning procedure).

## 5. Conclusions

We have proposed a novel algorithm that performs a weighted classification, using class membership probabilities as weights. The weights are obtained by first applying the SNOB algorithm as a feature mapping, using a Monte Carlo probability approximation to calculate probabilities, and a double-weighting scheme for the unlabeled examples.

In addition to showing that our algorithms performed comparably or better than the current state of the art on several datasets, we also demonstrated that similar methodology for calculating the class membership probabilities (this time applying a Gaussian Mixture Model instead of Monte Carlo approximation) can be used to bypass the expensive parameter tuning step in the Biased SVM algorithm, resulting in a speedup of several orders of magnitude. We found that our method TBSVM, performed better than the original BSVM

TABLE 1. PERFORMANCE ON THE LARGEST 10 TOPICS IN THE REUTERS DATASET.

| Topics | Algorithm | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Optimal | | TBSVM | | EN_Single | | EN_Double | | SNOB_MC_Double | | BSVM | |
| | F Score | AUC | F Score | AUC | F Score | AUC | F Score | AUC | F Score | AUC | F Score | AUC |
| Earn | 65.52 | 99.36 | 53.60 | 78.67 | 52.93 | 77.6 | 57.28 | 84.97 | 57.49 | 84.12 | 53.04 | 77.78 |
| Acq | 52.67 | 98.22 | 43.11 | 82.39 | 43.10 | 82.39 | 44.52 | 91.03 | 49.54 | 90.91 | 43.10 | 82.39 |
| Money-fx | 24.74 | 95.77 | 17.39 | 73.68 | 17.52 | 74.43 | 18.78 | 79.59 | 21.53 | 88.39 | 17.52 | 74.41 |
| Grain | 24.16 | 98.58 | 16.55 | 74.15 | 16.58 | 74.33 | 19.04 | 85.27 | 23.27 | 97.89 | 16.58 | 74.33 |
| Crude | 23.26 | 97.77 | 17.18 | 78.32 | 17.19 | 78.35 | 19.10 | 88.94 | 22.56 | 96.97 | 17.19 | 78.34 |
| Trade | 20.95 | 97.07 | 16.16 | 80.39 | 16.31 | 81.54 | 18.02 | 91.31 | 19.48 | 94.24 | 16.30 | 81.53 |
| Interest | 18.25 | 95.56 | 13.28 | 78.20 | 13.31 | 78.44 | 13.54 | 80.22 | 15.17 | 88.43 | 13.27 | 78.03 |
| Ship | 15.22 | 98.08 | 10.52 | 77.24 | 10.53 | 77.81 | 11.55 | 85.64 | 14.27 | 95.22 | 10.53 | 77.80 |
| Wheat | 14.82 | 97.76 | 11.34 | 82.12 | 11.34 | 82.12 | 12.18 | 91.61 | 14.36 | 98.74 | 11.34 | 82.12 |
| Corn | 12.44 | 98.59 | 8.43 | 75.28 | 8.42 | 75.23 | 9.72 | 88.41 | 10.84 | 95.22 | 8.42 | 75.23 |
| Average | 27.20 | 97.68 | 20.75 | 78.04 | 20.72 | 78.23 | 22.37 | 86.70 | 24.85 | 93.01 | 20.73 | 78.20 |

algorithm on some datasets, and worse on others. It is thus difficult to conclude whether TBSVM or BSVM has better accuracy, but the computational savings from our algorithm are substantial.

These results held across both biased and unbiased datasets, suggesting that our SNOB_MC_Double algorithm is a strong choice for any PU learning task.

## 6. Acknowledgments

We would like to thank the Simons Foundation, specifically the Simons Center for Data Analysis, and NYU-ITS, specifically Muataz Al-Barwani and the NYU Abu Dhabi ITS high performance computing team. RB was supported by the Simons Foundation and US National Science Foundation grants IOS-1126971, CBET-1067596 and CHE-1151554, and National Institutes of Health GM 32877-21/22, PN2-EY016586, IU54CA143907-01 and EY016586-06. DS was partially supported by the U.S. National Science Foundation under grants DBI-0445666, DBI-0421604, NSF IOS-0922738, MCB-0929339, and N2010 IOB-0519985 as well as NIH 2R01GM032877-25A1. This support is greatly appreciated.

## References

[1] G. Blanchard, G. Lee, and C. Scott. Semi-supervised novelty detection. *The Journal of Machine Learning Research*, 11:2973–3009, 2010.

[2] 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. ACM, 2008.

[3] N. Kushmerick. Learning to remove internet advertisements. In *Proceedings of the third annual conference on Autonomous Agents*, pages 175–181. ACM, 1999.

[4] W. Li, Q. Guo, and C. Elkan. A positive and unlabeled learning algorithm for one-class classification of remote-sensing data. *Geoscience and Remote Sensing, IEEE Transactions on*, 49(2):717–725, 2011.

[5] X. Li, S. Y. Philip, B. Liu, and S.-K. Ng. Positive unlabeled learning for data stream classification. In *SDM*, volume 9, pages 257–268. SIAM, 2009.

[6] M. Lichman. UCI machine learning repository, 2013.

[7] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 179–186. IEEE, 2003.

[8] G. McLachlan and D. Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[9] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach: a case study in intensive care monitoring. Technical report, Technical Report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund, 1999.

[10] J. J. Rocchio. Relevance feedback in information retrieval. 1971.

[11] M. H. Saier, C. V. Tran, and R. D. Barabote. Tcdb: the transporter classification database for membrane transport protein analyses and information. *Nucleic acids research*, 34(suppl 1):D181–D186, 2006.

[12] N. Youngs, D. Penfold-Brown, R. Bonneau, and D. Shasha. Negative example selection for protein function prediction: the nogo database. *PLoS computational biology*, 10(6):e1003644, 2014.

[13] N. Youngs, D. Penfold-Brown, K. Drew, D. Shasha, and R. Bonneau. Parametric bayesian priors and better choice of negative examples improve protein function prediction. *Bioinformatics*, page btt110, 2013.

[14] M. Yousef, S. Jung, L. C. Showe, and M. K. Showe. Learning from positive examples when the negative class is undetermined-microrna gene identification. *Algorithms for Molecular Biology*, 3(2), 2008.

[15] H. Yu, J. Han, and K. C.-C. Chang. Pebl: positive example based learning for web page classification using svm. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–248. ACM, 2002.

[16] X.-M. Zhao, Y. Wang, L. Chen, and K. Aihara. Gene function prediction using labeled and unlabeled data. *BMC bioinformatics*, 9(1):57, 2008.