Learning Classifiers from Only Positive and Unlabeled Data

Charles Elkan
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404
elkan@cs.ucsd.edu

Keith Noto
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0404
knoto@cs.ucsd.edu

ABSTRACT

The input to an algorithm that learns a binary classifier normally consists of two sets of examples, where one set consists of positive examples of the concept to be learned, and the other set consists of negative examples. However, it is often the case that the available training data are an incomplete set of positive examples, and a set of unlabeled examples, some of which are positive and some of which are negative. The problem solved in this paper is how to learn a standard binary classifier given a nontraditional training set of this nature.

Under the assumption that the labeled examples are selected randomly from the positive examples, we show that a classifier trained on positive and unlabeled examples predicts probabilities that differ by only a constant factor from the true conditional probabilities of being positive. We show how to use this result in two different ways to learn a classifier from a nontraditional training set. We then apply these two new methods to solve a real-world problem: identifying protein records that should be included in an incomplete specialized molecular biology database. Our experiments in this domain show that models trained using the new methods perform better than the current state-of-the-art biased SVM method for learning from positive and unlabeled examples.

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications—
data mining.

General Terms

Algorithms, theory.

Keywords

Supervised learning, unlabeled examples, text mining, bioinformatics.

1. INTRODUCTION

The input to an algorithm that learns a binary classifier consists normally of two sets of examples. One set is positive examples x such that the label $y = 1$, and the other set is negative examples x such that $y = 0$. However, suppose the available input consists of just an incomplete set of positive examples, and a set of unlabeled examples, some of which are positive and some of which are negative. The problem we solve in this paper is how to learn a traditional binary classifier given a nontraditional training set of this nature.

Learning a classifier from positive and unlabeled data, as opposed to from positive and negative data, is a problem of great importance. Most research on training classifiers, in data mining and in machine learning assumes the availability of explicit negative examples. However, in many real-world domains, the concept of a negative example is not natural. For example, over 1000 specialized databases exist in molecular biology [7]. Each of these defines a set of positive examples, namely the set of genes or proteins included in the database. In each case, it would be useful to learn a classifier that can recognize additional genes or proteins that should be included. But in each case, the database does not contain any explicit set of examples that should *not* be included, and it is unnatural to ask a human expert to identify such a set. Consider the database that we are associated with, which is called TCDB [15]. This database contains information about over 4000 proteins that are involved in signaling across cellular membranes. If we ask a biologist for examples of proteins that are not involved in this process, the only answer is "all other proteins." To make this answer operational, we could take all proteins mentioned in a comprehensive unspecialized database such as SwissProt [1]. But these proteins are unlabeled examples, not negative examples, because some of them are proteins that should be in TCDB. Our goal is precisely to discover these proteins.

This paper is organized as follows. First, Section 2 formalizes the scenario of learning from positive and unlabeled examples, presents a central result concerning this scenario, and explains how to use it to make learning from positive and unlabeled examples essentially equivalent to learning from positive and negative examples. Next, Section 3 derives how to use the same central result to assign weights to unlabeled examples in a principled way, as a second method of learning using unlabeled examples. Then, Section 4 describes a synthetic example that illustrates the results of Section 2. Section 5 explains the design and findings of an experiment showing that our two new methods perform better than the

best previously suggested method for learning from positive and unlabeled data. Finally, Section 6 summarizes previous work in the same area, and Section 7 summarizes our findings.

2. LEARNING A TRADITIONAL CLASSIFIER FROM NONTRADITIONAL INPUT

Let x be an example and let $y \in \{0, 1\}$ be a binary label. Let $s = 1$ if the example x is labeled, and let $s = 0$ if x is unlabeled. Only positive examples are labeled, so $y = 1$ is certain when $s = 1$, but when $s = 0$, then either $y = 1$ or $y = 0$ may be true.

Formally, we view x, y, and s as random variables. There is some fixed unknown overall distribution $p(x, y, s)$ over triples $\langle x, y, s \rangle$. A nontraditional training set is a sample drawn from this distribution that consists of unlabeled examples $\langle x, s = 0 \rangle$ and labeled examples $\langle x, s = 1 \rangle$. The fact that only positive examples are labeled can be stated formally as the equation

$$p(s = 1|x, y = 0) = 0. \qquad (1)$$

In words, the probability that an example x appears in the labeled set is zero if $y = 0$.

There is a subtle but important difference between the scenario considered here, and the scenario considered in [21]. The scenario here is that the training data are drawn randomly from $p(x, y, s)$, but for each tuple $\langle x, y, s \rangle$ that is drawn, only $\langle x, s \rangle$ is recorded. The scenario of [21] is that two training sets are drawn independently from $p(x, y, s)$. From the first set all x such that $s = 1$ are recorded; these are called "cases" or "presences." From the second set all x are recorded; these are called the background sample, or contaminated controls, or pseudo-absences.

The single-training-set scenario considered here provides strictly more information than the case-control scenario. Obviously, both scenarios allow $p(x)$ to be estimated. However, the first scenario also allows the constant $p(s = 1)$ to be estimated in an obvious way, while the case-control scenario does not. This difference turns out to be crucial: it is possible to estimate $p(y = 1)$ only in the first scenario.

The goal is to learn a function $f(x)$ such that $f(x) = p(y = 1|x)$ as closely as possible. We call such a function f a traditional probabilistic classifier. Without some assumption about which positive examples are labeled, it is impossible to make progress towards this goal. Our basic assumption is the same as in previous research: that the labeled positive examples are chosen completely randomly from all positive examples. What this means is that if $y = 1$, the probability that a positive example is labeled is the same constant regardless of x. We call this assumption the "selected completely at random" assumption. Stated formally, the assumption is that

$$p(s = 1|x, y = 1) = p(s = 1|y = 1). \qquad (2)$$

Here, $c = p(s = 1|y = 1)$ is the constant probability that a positive example is labeled. This "selected completely at random" assumption is analogous to the "missing completely at random" assumption that is often made when learning from data with missing values [10, 17, 18]. Another way of stating the assumption is that s and x are conditionally independent given y.

So, a training set is a random sample from a distribution $p(x, y, s)$ that satisfies Equations (1) and (2). Such a training set consists of two subsets, called the "labeled" ($s = 1$) and "unlabeled" ($s = 0$) sets. Suppose we provide these two sets as inputs to a standard training algorithm. This algorithm will yield a function $g(x)$ such that $g(x) = p(s = 1|x)$ approximately. We call $g(x)$ a nontraditional classifier. Our central result is the following lemma that shows how to obtain a traditional classifier $f(x)$ from $g(x)$.

Lemma 1: Suppose the "selected completely at random" assumption holds. Then $p(y = 1|x) = p(s = 1|x)/c$ where $c = p(s = 1|y = 1)$.
Proof: Remember that the assumption is $p(s = 1|y = 1, x) = p(s = 1|y = 1)$. Now consider $p(s = 1|x)$. We have that

$$
\begin{aligned}
p(s = 1|x) &= p(y = 1 \wedge s = 1|x) \\
&= p(y = 1|x)p(s = 1|y = 1, x) \\
&= p(y = 1|x)p(s = 1|y = 1).
\end{aligned}
$$

The result follows by dividing each side by $p(s = 1|y = 1)$. ∎

Although the proof above is simple, the result has not been published before, and it is not obvious. The reason perhaps that the result is novel is that although the learning scenario has been discussed in many previous papers, including [3, 5, 11, 26, 6], and these papers do make the "selected completely at random" assumption either explicitly or implicitly, the scenario has not previously been formalized using a random variable s to represent the fact of an example being selected.

Several consequences of the lemma are worth noting. First, f is an increasing function of g. This means that if the classifier f is only used to rank examples x according to the chance that they belong to class $y = 1$, then the classifier g can be used directly instead of f.

Second, $f = g/p(s = 1|y = 1)$ is a well-defined probability $f \leq 1$ only if $g \leq p(s = 1|y = 1)$. What this says is that $g > p(s = 1|y = 1)$ is impossible. This is reasonable because the "positive" (labeled) and "negative" (unlabeled) training sets for g are samples from overlapping regions in x space. Hence it is impossible for any example x to belong to the "positive" class for g with a high degree of certainty.

The value of the constant $c = p(s = 1|y = 1)$ can be estimated using a trained classifier g and a validation set of examples. Let V be such a validation set that is drawn from the overall distribution $p(x, y, s)$ in the same manner as the nontraditional training set. Let P be the subset of examples in V that are labeled (and hence positive). The estimator of $p(s = 1|y = 1)$ is the average value of $g(x)$ for x in P. Formally the estimator is $e_1 = \frac{1}{n} \sum_{x \in P} g(x)$ where n is the cardinality of P.

We shall show that $e_1 = p(s = 1|y = 1) = c$ if it is the case that $g(x) = p(s = 1|x)$ for all x. To do this, all we need to show is that $g(x) = c$ for $x \in P$. We can show this as follows:

$$
\begin{aligned}
g(x) &= p(s = 1|x) \\
&= p(s = 1|x, y = 1)p(y = 1|x) \\
&\quad + p(s = 1|x, y = 0)p(y = 0|x) \\
&= p(s = 1|x, y = 1) \cdot 1 + 0 \cdot 0 \ \ \text{since} \ \ x \in P \\
&= p(s = 1|y = 1).
\end{aligned}
$$

A second estimator of c is $e_2 = \sum_{x \in P} g(x) / \sum_{x \in V} g(x)$.

This estimator is almost equivalent to e_1, because

$$E[\sum_{x \in V} g(x)] = p(s = 1) \cdot m = E[n|m]$$

where m is the cardinality of V.

A third estimator of c is $e_3 = \max_{x \in V} g(x)$. This estimator is based on the fact that $g(x) \leq c$ for all x.

Which of these three estimators is best? The first estimator is exactly correct if $g(x) = p(s = 1|x)$ precisely for all x, but of course this condition never holds in practice. We can have $g(x) \neq p(s = 1|x)$ for two reasons: because g is learned from a random finite training set, and/or because the family of models from which $g(x)$ is selected does not include the true model. For example, if the distributions of x given $y = 0$ and x given $y = 1$ are Gaussian with different covariances, as in Section 4 below, then logistic regression can model $p(y = 1|x)$ and $p(s = 1|x)$ approximately, but not exactly. In the terminology of statistics, logistic regression is mis-specified in this case.

In practice, when $g(x) \neq p(s = 1|x)$, the first estimator is still the best one to use. Compared to the third estimator, it will have much lower variance because it is based on averaging over m examples instead of on just one example. It will have slightly lower variance than the second estimator because the latter is exposed to additional variance via its denominator. Note that in principle any single example from P is sufficient to determine c, but that in practice averaging over all members of P is preferable.

3. WEIGHTING UNLABELED EXAMPLES

There is an alternative way of using Lemma 1. Let the goal be to estimate $E_{p(x,y,s)}[h(x, y)]$ for any function h, where $p(x, y, s)$ is the overall distribution. To make notation more concise, write this as $E[h]$. We want an estimator of $E[h]$ based on a nontraditional training set of examples of the form $\langle x, s \rangle$.

Clearly $p(y = 1|x, s = 1) = 1$. Less obviously,

$$
\begin{aligned}
p(y = 1|x, s = 0) &= \frac{p(s = 0|x, y = 1)p(y = 1|x)}{p(s = 0|x)} \\
&= \frac{[1 - p(s = 1|x, y = 1)]p(y = 1|x)}{1 - p(s = 1|x)} \\
&= \frac{(1 - c)p(y = 1|x)}{1 - p(s = 1|x)} \\
&= \frac{(1 - c)p(s = 1|x)/c}{1 - p(s = 1|x)} \\
&= \frac{1 - c}{c} \frac{p(s = 1|x)}{1 - p(s = 1|x)}.
\end{aligned}
$$

By definition

$$
\begin{aligned}
E[h] &= \int_{x,y,s} h(x, y)p(x, y, s) \\
&= \int_x p(x) \sum_{s=0}^{1} p(s|x) \sum_{y=0}^{1} p(y|x, s)h(x, y) \\
&= \int_x p(x) \Big(p(s = 1|x)h(x, 1) \\
&\qquad + p(s = 0|x)[p(y = 1|x, s = 0)h(x, 1) \\
&\qquad\qquad + p(y = 0|x, s = 0)h(x, 0)] \Big).
\end{aligned}
$$

The plugin estimate of $E[h]$ is then the empirical average

$$\frac{1}{m} \Big(\sum_{\langle x,s=1 \rangle} h(x, 1) + \sum_{\langle x,s=0 \rangle} w(x)h(x, 1) + (1 - w(x))h(x, 0) \Big)$$

where

$$w(x) = p(y = 1|x, s = 0) = \frac{1 - c}{c} \frac{p(s = 1|x)}{1 - p(s = 1|x)} \qquad (3)$$

and m is the cardinality of the training set. What this says is that each labeled example is treated as a positive example with unit weight, while each unlabeled example is treated as a combination of a positive example with weight $p(y = 1|x, s = 0)$ and a negative example with complementary weight $1 - p(y = 1|x, s = 0)$. The probability $p(s = 1|x)$ is estimated as $g(x)$ where g is the nontraditional classifier explained in the previous section.

There are two ways in which the result above on estimating $E[h]$ can be used to modify a learning algorithm in order to make it work with positive and unlabeled training data. The first method is to express the learning algorithm so that it uses the training set only via the computation of averages, and then to use the result above to estimate each of these averages. The second method is to modify the learning algorithm so that training examples have individual weights. Then, positive examples are given unit weight and unlabeled examples are duplicated; one copy of each unlabeled example is made positive with weight $p(y = 1|x, s = 0)$ and the other copy is made negative with weight $1 - p(y = 1|x, s = 0)$. This second method is the one used in experiments below.

As a special case of the result above, consider $h(x, y) = y$. We obtain

$$
\begin{aligned}
E[y] &= p(y = 1) \\
&= \frac{1}{m} \Big(\sum_{\langle x,s=1 \rangle} 1 + \sum_{\langle x,s=0 \rangle} w(x)1 + (1 - w(x))0 \Big) \\
&= \frac{1}{m} \Big(n + \sum_{\langle x,s=0 \rangle} w(x) \Big) \qquad (4)
\end{aligned}
$$

where n is the cardinality of the labeled training set, and the sum is over the unlabeled training set. This result solves an open problem identified in [26], namely how to estimate $p(y = 1)$ given only the type of nontraditional training set considered here and the "selected completely at random" assumption.

There is an alternative way to estimate $p(y = 1)$. By definition

$$c = p(s = 1|y = 1) = \frac{p(s = 1 \wedge y = 1)}{p(y = 1)}$$

so

$$p(y = 1) = \frac{p(s = 1 \wedge y = 1)}{c}.$$

The obvious estimator of $p(s = 1 \wedge y = 1)$ is n/m, which yields the estimator

$$\frac{n}{m} \frac{n}{\sum_{x \in P} g(x)} = \frac{n^2}{m \sum_{x \in P} g(x)}$$

for $p(y = 1)$. Note that both this estimator and (4) are greater than n/m, as is expected for $p(y = 1)$.

The expectation $E[y] = p(y = 1)$ is the prevalence of y among all x, both labeled and unlabeled. The reasoning above shows how to estimate this prevalence assum-
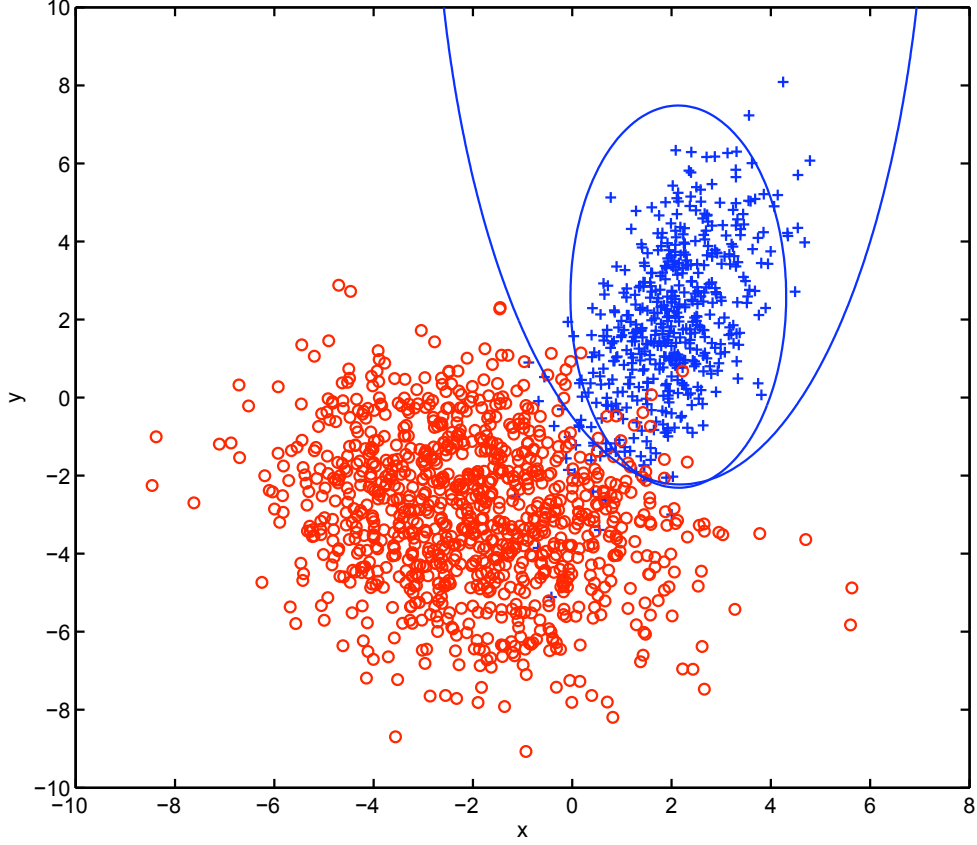
Figure 1: Data points lie in two dimensions. Blue pluses are positive examples, while red circles are negative examples. The large ellipse is the result of logistic regression trained on all the data. It shows the set of points x for which $p(y|x) = 0.5$ is estimated. The small ellipse is the result of logistic regression trained on positive labeled data versus all other data, then transformed following Lemma 1. The two ellipses represent similar classifiers in practice since they agree closely everywhere that data points of both classes have high density.

ing the single-training-set scenario. In the case-control scenario, where labeled training examples are obtained separately from unlabeled training examples, it can be proved that $p(y = 1)$ cannot be identified [21].

The arguments of this section and the preceding one are derivations about probabilities, so they are only applicable to the outputs of an actual classifier if that classifier produces correct probabilities as its output. A classifier that produces approximately correct probabilities is called well-calibrated. Some learning methods, in particular logistic regression, do give well-calibrated classifiers, even in the presence of mis-specification and finite training sets. However, many other methods, in particular naive Bayes, decision trees, and support vector machines (SVMs), do not. Fortunately, the outputs of these other methods can typically be postprocessed into calibrated probabilities.

The two most common postprocessing methods for calibration are isotonic regression [25], and fitting a one-dimensional logistic regression function [14]. We apply the latter

method, which is often called Platt scaling, to SVM classifiers in Section 5 below.

4. AN ILLUSTRATION

To illustrate the method proposed in Section 2 above, we generate 500 positive data points and 1000 negative data points, each from a two-dimensional Gaussian as shown in Figure 1. We then train two classifiers: one using all the data, and one using 20% of the positive data as labeled positive examples, versus all other data as negative examples.

Figure 1 shows the ideal trained classifier as a large ellipse. Each point on this ellipse has predicted probability 0.5 of belonging to the positive class. The transformed non-traditional classifier is the small ellipse; the transformation following Lemma 1 uses the estimate e_1 of $p(s = 1|y = 1)$. Based on a validation set of just 20 labeled examples, this estimated value is $e_1 = 0.1928$, which is very close to the true value 0.2. Although the two ellipses are visually different, they correspond closely in the area where both positive

and negative data points have high density, so they represent similar classifiers for this application.

Given a data point $\langle x_1, x_2 \rangle$, both classifiers use the representation $\langle x_1, x_2, x_1^2, x_2^2 \rangle$ as input in order to allow the contours $p(y = 1|x) = 0.5$ to be quadratic sections, as they are in Figure 1. Expanding the input representation in this way is similar to using a nonlinear kernel with a support vector machine. Because the product $x_1 x_2$ is not part of the input representation, the ellipses are constrained to be axis-parallel. Logistic regression with this input representation is therefore mis-specified, i.e. not capable of representing exactly the distributions $p(y = 1|x)$ and $p(s = 1|x)$. Analogous mis-specification is likely to occur in real-world domains.

5. APPLICATION TO REAL-WORLD DATA

One common real-world application of learning from positive and unlabeled data is in document classification. Here we describe experiments that use documents that are records from the SwissProt database.

We call the set of positive examples P. This set consists of 2453 records obtained from a specialized database named TCDB [15]. The set of unlabeled examples U consists of 4906 records selected randomly from SwissProt excluding its intersection with TCDB, so U and P are disjoint. Domain knowledge suggests that perhaps about 10% of the records in U are actually positive.

This dataset is useful for evaluating methods for learning from positive and unlabeled examples because in previous work we did in fact manually identify the subset of actual positive examples inside U; call this subset Q. The procedure used to identify Q, which has 348 members, is explained in [2]. Let $N = U \backslash Q$ so the cardinality of N is 4558. The three sets of records N, P, and Q are available at `www.cs.ucsd.edu/users/elkan/posonly`.

The P and U datasets were obtained separately, and U is a sample from the whole population, as opposed to $P \cup U$. Hence this experiment is an example of the case-control scenario explained in Section 2, not of the single-training-set scenario. However, we can still apply the methods suggested in that section and evaluate their success. When applied in practice, $P \cup U$ will be all of SwissProt, so the scenario will be the single-training-set one.

Our experiments compare four approaches: (i) standard learning from $P \cup Q$ versus N, (ii) learning from P versus U with adjustment of output probabilities, (iii) learning from P and U after double weighting of U, and (iv) the biased SVM method from [11] explained in Section 6 below. Approach (i) is the baseline, that is the ideal: learning a standard classifier from fully labeled positive and negative training sets. We expect its accuracy to be the highest; we hope that approaches (ii) and (iii) can achieve almost as good accuracy. Approach (iv) is the most successful method described in previous research. To make comparisons fair, all four methods are based on soft-margin SVMs with linear kernels.

Each of the four methods yields a classifier that assigns numerical scores to test examples, so for each method we can plot its receiver operating characteristic (ROC) curve. Each method also has a natural threshold that can be used to convert numerical scores into yes/no predictions for test examples. For methods (i) and (iii) this natural threshold is 0.5, since these methods yield scores that are well-calibrated probabilities. For method (iv) the natural threshold is zero,

since this method is an SVM. For method (ii) the natural threshold is $0.5c$ where $c = p(s = 1|y = 1)$. As in Section 4 above, we use the estimator e_1 for c.

For each of the four methods, we do cross-validation with ten folds, and obtain a combined confusion matrix from the ten testing subsets. With one confusion matrix for each approach, we compute its recall and precision. All these numbers are expected to be somewhere around 95%. Cross-validation proceeds as follows. Partition P, Q, and N randomly into ten subsets each of size as equal as possible. For example Q will have eight subsets of size 35 and two of size 34. For each of the ten trials, reserve one subset of P, Q, and N for testing. Use the other nine subsets of each for training. In every trial, each approach gives one final classifier. In trial number i, this classifier is applied to the testing subsets P_i, Q_i, and N_i, yielding a confusion matrix of the following form:

	positive	negative	predicted
$P_i \cup Q_i$	a_i	b_i	
N_i	c_i	d_i	
actual			

The combined confusion matrix reported for each approach has entries $a = \sum_{i=1}^{10} a_i$ etc. In this matrix a is the number of true positives, b is the number of false negatives, c is the number of false positives, and d is the number of true negatives. Finally, precision is defined as $p = a/(a + c)$ and recall as $r = a/(a + b)$.

The basic learning algorithm for each method is an SVM with a linear kernel as implemented in libSVM [9]. For approach (ii) we use Platt scaling to get probability estimates which are then adjusted using Lemma 1. For approach (iii) we run libSVM twice. The first run uses Platt scaling to get probability estimates, which are then converted into weights following Equation (3) at the end of Section 3. Next, these weights are used for the second run. Although the official version of libSVM allows examples to be weighted, it requires all examples in one class to have the same weight. We use the modified version of libSVM by Ming-Wei Chang and Hsuan-Tien Lin, available at `www.csie.ntu.edu.tw/-`$\sim$`cjlin/libsvmtools/#15`, that allows different weights for different examples.

For approach (iv) we run libSVM many times using varying weights for the negative and unlabeled examples. The details of this method are summarized in Section 6 below and are the same as in the paper that proposed the method originally [11]. With this approach different weights for different examples are not needed, but a validation set to choose the best settings is needed. Given that the training set in each trial consists of 90% of the data, a reasonable choice is to use 70% of the data for training and 20% for validation in each trial. After the best settings are chosen in each trial, libSVM is rerun using all 90% for training, for that trial. Note that different settings may be chosen in each of the ten trials.

The results of running these experiments are shown in Table 5. Accuracies and F1 scores are calculated by thresholding trained models as described above, while areas under the ROC curve do not depend on a specific threshold. "Relative time" is the number of times an SVM must be trained for one fold of cross-validation.

As expected, training on $P \cup Q$ versus N, method (i), performs the best, measured both by F1 and by area under the
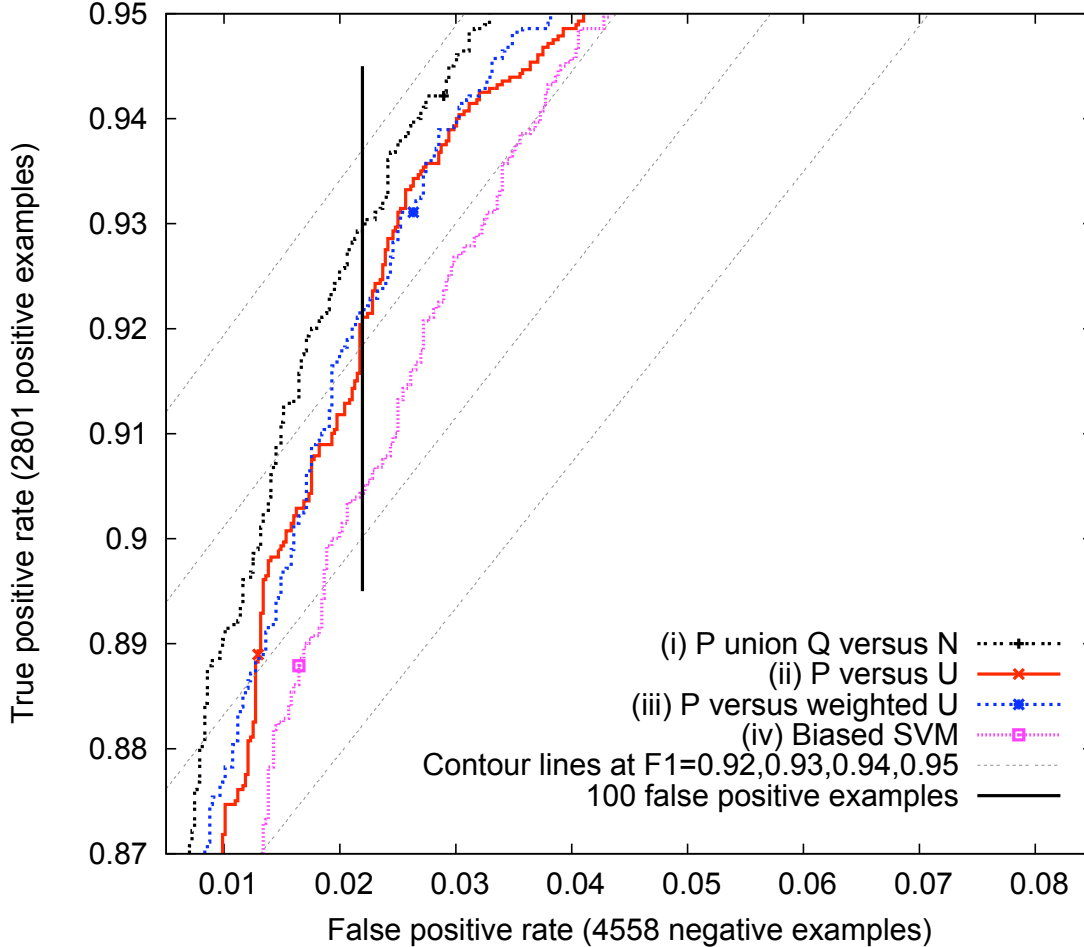
Figure 2: Results for the experiment described in Section 5. The four ROC curves are produced by four different methods. (i) Training on $P \cup Q$ versus N; the point on the line uses the threshold $p(y = 1|x) = 0.5$. (ii) Training on P versus U; the point shown uses the threshold $p(s = 1|x)/e_1 = 0.5$ where e_1 is an estimate of $p(s = 1|y = 1)$. (iii) Training on P versus U, where each example in U is labeled positive with a weight of $p(y = 1|x, z = 0)$ and also labeled negative with complementary weight. (iv) Biased SVM training, with penalties C_U and C_P chosen using validation sets. Note that, in order to show the differences between methods better, only the important part of the ROC space is shown.

ROC curve. However, this is the ideal method that requires knowledge of all true labels. Our two methods that use only positive and unlabeled examples, (ii) and (iii), perform better than the current state-of-the-art, which is method (iv). Although methods (ii) and (iv) yield almost the same area under the ROC curve, Figure 2 shows that the ROC curve for method (ii) is better in the region that is important from an application perspective. Method (iv) is the slowest by far because it requires exhaustive search for good algorithm settings. Since methods (ii) and (iii) are mathematically well-principled, no search for algorithm settings is needed.

The new methods (ii) and (iii) do require estimating $c = p(s = 1|y = 1)$. We do this with the e_1 estimator described in Section 2. Because of cross-validation, a different estimate e_1 is computed for each fold. All these estimates are within 0.15% of the best estimate we can make using knowledge of the true labels, which is $p(s = 1|y = 1) = \frac{|P|}{|P \cup Q|} = 0.876$.

Consider the ROC curves in Figure 2; note that only part of the ROC space is shown in this figure. Each of the four

curves is the result of one of the four methods. The points highlighted on the curves show the false positive/true positive trade-off at the thresholds described above. While the differences in the ROC curves may seem small visually, they represent a substantial practical difference. Suppose that a human expert will tolerate 100 negative records. This is represented by the black line in Figure 2. Then the expert will miss 9.6% of positive records using the biased SVM, but only 7.6% using the reweighting method, which is a 21% reduction in error rate, and a difference of 55 positive records in this case.

6. RELATED WORK

Several dozen papers have been published on the topic of learning a classifier from only positive and unlabeled training examples. Two general approaches have been proposed previously. The more common approach is (i) to use heuristics to identify unlabeled examples that are likely to be negative,

Table 1: Measures of performance for each of four methods.

method	accuracy	F1 score	area under ROC curve	relative time
(i) Ideal: Training on $P \cup Q$ versus N	0.9600	0.9472	0.9912	1
(ii) Training on P versus U	0.9497	0.9308	0.9895	**1**
(iii) Training on P versus weighted U	**0.9568**	**0.9422**	**0.9899**	2
(iv) Biased SVM	0.9465	0.9279	0.9895	621

and then (ii) to apply a standard learning method to these examples and the positive examples; steps (i) and (ii) may be iterated. Papers using this general approach include [24, 20, 23], and the idea has been rediscovered independently a few times, most recently in [22, Section 2.4]. The approach is sometimes extended to identify also additional positive examples in the unlabeled set [6]. The less common approach is to assign weights somehow to the unlabeled examples, and then to train a classifier with the unlabeled examples interpreted as weighted negative examples. This approach is used for example by [8, 12].

The first approach can be viewed as a special case of the second approach, where each weight is either 0 or 1. The second approach is similar to the method we suggest in Section 3 above, with three important differences. First, we view each unlabeled example as being both a weighted negative example *and* a weighted positive example. Second, we provide a principled way of choosing weights, unlike previous papers. Third, we assign different weights to different unlabeled examples, whereas previous work assigns the same weight to every unlabeled example.

A good paper that evaluates both traditional approaches, using soft-margin SVMs as the underlying classifiers, is [11]. The finding of that paper is that the approach of heuristically identifying likely negative examples is inferior. The weighting approach that is superior solves the following SVM optimization problem:

$$\text{minimize } \frac{1}{2}||w||^2 + C_P \sum_{i \in P} z_i + C_U \sum_{i \in U} z_i$$

subject to $y_i(w \cdot x + b) \geq 1 - z_i$ and $z_i \geq 0$ for all i.

Here P is the set of labeled positive training examples and U is the set of unlabeled training examples. For each example i the hinge loss is z_i. In order to make losses on P be penalized more heavily than losses on U, $C_P > C_U$. No direct method is suggested for setting the constants C_P and C_U. Instead, a validation set is used to select empirically the best values of C_P and C_U from the ranges $C_U = 0.01, 0.03, 0.05, \ldots, 0.61$ and $C_P/C_U = 10, 20, 30, \ldots, 200$. This method, called biased SVM, is the current state-of-the-art for learning from only positive and unlabeled documents. Our results in the previous section show that the two methods we propose are both superior.

The assumption on which the results of this paper depend is that the positive examples in the set P are selected completely at random. This assumption was first made explicit by [3] and has been used in several papers since, including [5]. Most algorithms based on this assumption need $p(y = 1)$ to be an additional input piece of information; a recent paper emphasizes the importance of $p(y = 1)$ for learning from positive and unlabeled data [4]. In Section 3 above, we show how to estimate $p(y = 1)$ empirically.

The most similar previous work to ours is [26]. Their approach also makes the "selected completely at random" assumption and also learns a classifier directly from the positive and unlabeled sets, then transforms its output following a lemma that they prove. Two important differences are that they assume the outputs of classifiers are binary as opposed to being estimated probabilities, and they do not suggest a method to estimate $p(s = 1|y = 1)$ or $p(y = 1)$. Hence the algorithm they propose for practical use uses a validation set to search for a weighting factor, like the weighting methods of [8, 12], and like the biased SVM approach [11]. Our proposed methods are orders of magnitude faster because correct weighting factors are computed directly.

The task of learning from positive and unlabeled examples can also be addressed by ignoring the unlabeled examples, and learning only from the labeled positive examples. Intuitively, this type of approach is inferior because it ignores useful information that is present in the unlabeled examples. There are two main approaches of this type. The first approach is to do probability density estimation, but this is well-known to be a very difficult task for high-dimensional data. The second approach is to use a so-called one-class SVM [16, 19]. The aim of these methods is to model a region that contains most of the available positive examples. Unfortunately, the outcome of these methods is sensitive to the values chosen for tuning parameters, and no good way is known to set these values [13]. Moreover, the biased SVM method has been reported to do better experimentally [11].

7. CONCLUSIONS

The central contribution of this paper is Lemma 1, which shows that if positive training examples are labeled at random, then the conditional probabilities produced by a model trained on the labeled and unlabeled examples differ by only a constant factor from the conditional probabilities produced by a model trained on fully labeled positive and negative examples.

Following up on Lemma 1, we show how to use it in two different ways to learn a classifier using only positive and unlabeled training data. We apply both methods to an important biomedical classification task whose purpose is to find new data instances that are relevant to a real-world molecular biology database. Experimentally, both methods lead to classifiers that are both hundreds of times faster and more accurate than the current state-of-the-art SVM-based method. These findings hold for four different definitions of accuracy: area under the ROC curve, F1 score or error rate using natural thresholds for yes/no classification, and recall at a fixed false positive rate that makes sense for a human expert in the application domain.

8. ACKNOWLEDGMENTS

9. REFERENCES

[1] B. Boeckmann, A. Bairoch, R. Apweiler, M. Blatter, A. Estreicher, E. Gasteiger, M. Martin, K. Michoud, C. O'Donovan, I. Phan, et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1):365–370, 2003.

[2] S. Das, M. H. Saier, and C. Elkan. Finding transport proteins in a general protein database. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 4702 of *Lecture Notes in Computer Science*, pages 54–66. Springer, 2007.

[3] F. Denis. PAC learning from positive statistical queries. In *Proceedings of the 9th International Conference on Algorithmic Learning Theory (ALT'98), Otzenhausen, Germany*, volume 1501 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 1998.

[4] F. Denis, R. Gilleron, and F. Letouzey. Learning from positive and unlabeled examples. *Theoretical Computer Science*, 348(1):70–83, 2005.

[5] F. Denis, R. Gilleron, and M. Tommasi. Text classification from positive and unlabeled examples. In *Proceedings of the Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002)*, pages 1927–1934, 2002.

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

[7] M. Galperin. The Molecular Biology Database Collection: 2008 update. *Nucleic Acids Research*, 36(Database issue):D2, 2008.

[8] W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003), Washington, DC*, pages 448–455, 2003.

[9] H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276, 2007.

[10] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., second edition, 2002.

[11] 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 (ICDM 2003)*, pages 179–188, 2003.

[12] Z. Liu, W. Shi, D. Li, and Q. Qin. Partially supervised classification - based on weighted unlabeled samples support vector machine. In *Proceedings of the First International Conference on Advanced Data Mining and Applications (ADMA 2005), Wuhan, China*, volume 3584 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2005.

[13] L. M. Manevitz and M. Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[14] J. C. Platt. Probabilities for SV machines. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–73. MIT Press, 1999.

[15] 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:D181–D186, 2006.

[16] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

[17] A. Smith and C. Elkan. A Bayesian network framework for reject inference. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 286–295, 2004.

[18] A. Smith and C. Elkan. Making generative classifiers robust to selection bias. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 657–666, 2007.

[19] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2004.

[20] C. Wang, C. Ding, R. F. Meraz, and S. R. Holbrook. PSoL: a positive sample only learning algorithm for finding non-coding RNA genes. *Bioinformatics*, 22(21):2590–2596, 2006.

[21] G. Ward, T. Hastie, S. Barry, J. Elith, and J. R. Leathwick. Presence-only data and the EM algorithm. *Biometrics*, 2008. In press.

[22] F. Wu and D. S. Weld. Autonomously semantifying Wikipedia. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal*, pages 41–50, Nov. 2007.

[23] H. Yu. Single-class classification with mapping convergence. *Machine Learning*, 61(1–3):49–69, 2006.

[24] H. Yu, J. Han, and K. C.-C. Chang. PEBL: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81, 2004.

[25] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining*, pages 694–699. AAAI Press (distributed by MIT Press), Aug. 2002.

[26] D. Zhang and W. S. Lee. A simple probabilistic approach to learning from positive and unlabeled examples. In *Proceedings of the 5th Annual UK Workshop on Computational Intelligence (UKCI)*, pages 83–87, Sept. 2005.