Classifier Invariant Approach to Learn from Positive-Unlabeled Data

Amit Dhurandhar
IBM Research
Yorktown Heights, U.S.A.
adhuran@us.ibm.com

Karthik S. Gurumoorthy
Amazon Development Center
Bangalore, India
gurumoor@amazon.com

***Abstract*—Learning from positive (P) and unlabeled (U) data has a rich history as it finds use in multiple applications. In this paper, we provide a novel framework to tackle this problem in a model agnostic fashion. We say model agnostic since, our solution involves identifying and weighting positive as well as negative examples in an unsupervised manner which could then be passed as input to *any* standard classification algorithm. Moreover, based on our framework we provide approximation guarantees for our algorithm in terms of how well the identified positive examples from U along with their weights match the distribution of P. Such a principled approach has been missing for other methods that belong to the model agnostic category, not to mention that the current state-of-the-art are model dependent strategies that involve modifying the training algorithm. For Kernel Support Vector Machines, trained on a (non-negative) weighted dataset that as such is the output of our method, we derive generalization bounds. Given the advantages of having model agnostic methods (viz. use with (almost) any classifier, one time running cost), we show that our algorithm which possesses these benefits, is competitive with the best methods based on experiments on three real datasets. In fact, in a couple of cases we observe that our approach has better test performance than even standard supervised learning which has access to all positive as well as negative labels.**

***Index Terms*—model agnostic, pu learning**

I. Introduction

Learning effectively from positive (P) and unlabeled (U) data, commonly referred to as PU learning, is important in many applications such as outlier detection, text classification, information retrieval and even matrix completion [1], [2], [3], [4]. Many times, one type of labeled data is easier to gather than having labelled data points from both the positive and negative (N) classes as in the traditional supervised learning set up, referred to as the PN learning in our context. For instance in remote sensing applications [5], data samples collected from urban areas are easier to categorize and label than those collected from multiple rural areas. PU learning also from the unsupervised setting, which is effectively U learning with no labelled information and also from the partially labelled semi-supervised approaches, which could be termed as PNU learning.

The techniques developed to handle PU data can be broadly classified into two kinds. The first [3], [6] tries to identify negative (N) examples in U, following which any binary classifier can applied to the new PN data. The second [1], models the U data as weighted N data. A subclass of the second strategy is the unbiased PU learning [7], [8] which are regarded as the state-of-the-art methods. Although the first strategy has been out of favor in recent years, mainly due to the presence of heuristics which makes them less preferred compared to the second strategy [7], it still has certain advantages over the latter. For instance, the second strategy invariable involves modifying the learning algorithm, while for the first any *off-the-shelf* binary classification method could potentially be used. While the techniques in the second category require estimation/knowledge of the prior distribution of P or N data in U, no such requirement exists for methods in the first category.

Given these favorable aspects of approaches in the first category, we would like to revive interest in this space by proposing a novel method with strong theoretical grounding for PU learning that belongs to this group of model agnostic algorithms. Our framework involves the following two steps:

(i) Define a maximum mean discrepancy (MMD) objective function [9] that can be used to select examples from U that best match the distribution of the P data, thus identifying candidate P (and consequently N) in U.

(ii) Learn non-negative weights for each selected example indicative of its likelihood of being a representative example of P. Non-negativity is desirable as it is counter-intuitive to have an example selected as a representative for the P class but have a (large) negative weight associated with it.

A. Advantages of our framework

Our approach hence has the following advantages over methods in the second category:

(i) Knowledge/estimation of class priors in U are not needed. It is important to note here that our approach actually creates a labeled dataset with weights from U and as such is distinct and more informative than just a prior estimation procedure [10].

(ii) A classification algorithm of one's own choice can be used to learn from the weighted labeled data created by our method. The learned weights could either be used to weight the loss of the classifier or create an expanded training set where the number of copies of an example is proportional to its weight.

(iii) Weights for examples that are known or identified to be positive are classifier agnostic. As such, this makes sense since labelling an example as positive should be independent of the learning algorithm chosen subsequently for classification.

(iv) Computation of weights is more efficient as it does not depend on the learning algorithm and is a one time cost. Moreover, as we will see, our algorithm only performs gradient evaluations and not function evaluations which is known to have significantly lower computational cost [11].

(v) In addition to being formal, we provide constant factor approximation guarantees for our method and thereby drawing a clear distinction with prior work on methods in this first category. We also derive a generalization bound for kernel support vector machines trained using the weighted labelled dataset output by our method.

(vi) Last but not the least, our method could be used in both case-control and censoring settings [12] as we do not make any assumptions on the distribution of U in relation to the true data distribution.

We show that along with these advantages, our method empirically achieves state-of-the-art performance on the Image Segment dataset, while being competitive with the best on MNIST and Letters datasets, thus possibly reviving interest in the first category methods.

II. Problem Definition

Let $X \times Y$ denote the input-output space, where for any positive integer d, $X \in \mathbb{R}^d$ and $Y \in \{+1, -1\}$. Given an underlying joint distribution $p(x, y)$ on this space, let $p_+(x) = p(x|y = +1)$.

In the case-control PU learning setting we have two datasets: the first dataset denoted by P contains n_+ independent and identically distributed (i.i.d.) samples such that $P = \{x_1^+, ..., x_{n_+}^+\} \sim p_+(x)$ and a second dataset denoted by U which contains n_u i.i.d. samples where $U = \{x_1^u, ..., x_{n_u}^u\} \sim p(x)$. Thus, the first dataset has only samples from the positive class, while the second dataset has a mixture of the two classes with no information about which examples belong to which particular class. The censoring PU setting is similar except that the U dataset may not be a representative sample from $p(x)$ and hence may contain significantly different proportion of positive and negative examples compared to a representative set. Given these datasets P and U, our goal is to learn a binary classifier $h : X \to Y$ that has low expected error, in other words generalizes well.

III. Methodology

We now provide our solution to tackle the PU learning problem from a model agnostic standpoint, in the sense that we identify potentially positive and negative examples and also associate non-negative weights with each of them that can then be passed to any binary classification algorithm of one's choice. The identified examples and the associated weights are thus fixed irrespective of the classification algorithm used. We first define our framework followed by a description of our algorithm. We then prove constant factor bounds for the quality of our selection as well as providing a generalization bound.

A. Framework

Let $P, U \in X$ be two datasets and $\mathcal{K}$ be the Reproducing Kernel Hilbert Space (RKHS) corresponding to a kernel $k : X \times X \to \mathcal{R}$. Considering the function $\phi_x(y) = k(x, y) \in \mathcal{K}$, the kernel inner product between two data points x_i and x_j is then defined as $k(x_i, x_j) = \langle \phi_{x_i}, \phi_{x_j} \rangle$.

The maximum mean discrepancy (MMD) is a measure of difference between two distributions p and q. Letting $\mu_p = \mathbb{E}_{x\sim p}[\phi_x]$, MMD is defined as:

$$MMD(\mathcal{K}, p, q) \equiv \sup_{\psi \in \mathcal{K}} \left(\mathbb{E}_{x\sim p}[\psi(x)] - \mathbb{E}_{y\sim q}[\psi(y)]\right) = \sup_{\psi \in \mathcal{K}} \langle \psi, \mu_p - \mu_q \rangle.$$

Our goal is to approximate μ_p by a weighted combination of n_p sub-samples $Z \subseteq U$ drawn from the distribution q, i.e., $\mu_p(x) \approx \sum_{j:z_j \in Z} w_j k(z_j, x)$, where w_j is the associated weight of the sample $z_j \in U$. Conceptually, this equals selecting the correct positive examples from U and estimating their importance in matching the distribution of positives in P. We thus need to choose a set $Z \subseteq U$ of cardinality $(|.|)$ n_p and learn the weights w_j that minimizes the finite sample MMD metric with the *non-negativity constraint* on the weights, as given below [9]:

$$\begin{aligned} &\widehat{MMD}(\mathcal{K}, P, Z, w) \\ &= \frac{1}{(n_+)^2} \sum_{x_i, x_j \in P} k(x_i, x_j) - \frac{2}{n_+} \sum_{z_j \in Z} w_j \sum_{x_i \in P} k(x_i, z_j) \\ &+ \sum_{z_i, z_j \in Z} w_i w_j k(z_i, z_j); \text{ subject to } w_j \geq 0, \forall z_j \in Z \end{aligned} \tag{1}$$

where $n_+ = |P|$. Index the elements in U from 1 to $n_u = |U|$ and for any $Z \subseteq U$ let $L_Z \subseteq [n_u] = \{1, 2, \ldots, n_u\}$ be the set containing its indices. Discarding the constant terms in (1) that do not depend on Z and w, we define the function

$$l(w) \equiv w^T \mu_p - \frac{1}{2} w^T K w = \frac{1}{2} \left[\|y\|_2^2 - \|Aw - y\|_2^2 \right] \tag{2}$$

where $K_{i,j} = k(y_i, y_j)$ and $\mu_{p,j} = \frac{1}{n_+} \sum_{x_i \in P} k(x_i, y_j); \forall y_j \in U$. The last inequality is obtained by expressing $K = A^T A$ as it is symmetric and positive-definite, and letting $\mu_p = A^T y$. Our goal then is to find a index set L_Z with $|L_Z| \leq n_p$ and a corresponding w such that the set function $f : 2^{[n_u]} \to \mathbb{R}^+$ defined as:

$$f(L_Z) \equiv \max_{w: supp(w) \in L_Z, w \geq 0} l(w) \tag{3}$$

attains maximum. Here $supp(w) = \{j : w_j > 0\}$. We will denote the maximizing weight values for the set L_Z by $\boldsymbol{\zeta}^{(L_Z)}$ and m-size set at which $f(.)$ attains maximum as L^*.

Accurately specifying n_p is equivalent to knowing the proportion of positives in U which is usually not known. Contrary

Algorithm 1 Model Agnostic PU Learn (MAPUL)

Input: P, U and lower bound ϵ on increase in $l(.)$

Let $L_{u+} = \emptyset$, $\boldsymbol{\zeta}^{(L_{u+})} = \mathbf{0}$, $L_+ = [n_+]$ {L_+ contains the indices of examples in P.}
$\mathbf{g} = \nabla l(\mathbf{0}) = \mu_p$
Let $\boldsymbol{\zeta}^{(L_+)} = \underset{\mathbf{w}:supp(\mathbf{w})\in L_+, \mathbf{w}\geq 0}{\operatorname{argmax}} \widehat{MMD}(\mathcal{K}, P, P, w)$
{Compute weights for all examples in P.}
while exit condition is not met {i.e. increase in successive $f(.) \geq \epsilon$ } **do**
  $v = \underset{j\in[n_u]\setminus L_{u+}}{\operatorname{argmax}} g_j$
  $L_{u+} = L_{u+} \cup \{v\}$
  $\boldsymbol{\zeta}^{(L_{u+})} = \underset{\mathbf{w}:supp(\mathbf{w})\in L_{u+}, \mathbf{w}\geq 0}{\operatorname{argmax}} l(\mathbf{w})$ {Find positive examples and their weight from U.}
  $\mathbf{g} = \nabla l\left(\boldsymbol{\zeta}^{(L_{u+})}\right) = \mu_p - K\boldsymbol{\zeta}^{(L_{u+})}$
end while
Let $w_{\text{med}} = median\{w | w \in \boldsymbol{\zeta}^{(L_{u+})} \cup \boldsymbol{\zeta}^{(L_+)}\}$ {Weight for potentially negative examples.}
return L_{u+}, $\boldsymbol{\zeta}^{(L_{u+})}$, $\boldsymbol{\zeta}^{(L_+)}$ and w_{med}

to requiring n_p, we overcome this issue by selecting examples along with computing their weights until the *increment* in the set function $f(.)$ in eq.(3) is less than some pre-specified ϵ. This approach is cogent as $f(.)$ being monotonically non-decreasing over containment on the support of w and bounded above by $0.5\|y\|_2^2$, converges over iterations by the Monotone Convergence theorem. Hence the non-negative increments in the function value will approach zero. We thus specify a small ϵ which leads us to choosing m examples indexed as in L_{u+} from U. As, (i) we do not try to determine the true prior of positives in U but rather match the P distribution, and (ii) our technique described in Algorithm 1 *incrementally* and *selectively* adds examples to L_{u+}, the set size m will most likely lower bound n_p and the selected examples will primarily be positive even for very small $\epsilon = 10^{-11}$ as witnessed in our experiments.

B. Algorithmic Description

Algorithm 1 is our model agnostic PU learning method abbreviated as *MAPUL*. As the first step, we compute non-negative weights for each example in P indicative of their importance. There is no selection at this point, rather just computation of weights. The next part is the crux of our algorithm. Here we select examples from U that best match the distribution of examples in P. We keep selecting examples and determine their weights till the increase in the set function $f(.)$ is less than a pre-specified ϵ. We finally get m examples from U (most likely positive) with corresponding non-negative weights.

We label the remaining examples in U as negative and assign a weight of w_{med} (i.e. median of all learned weights) to all of them. We consider the median weight, as it is possible that some of the remaining examples in U might still belong to P and we do not want to force the classifier to classify them as negative by assigning the maximum or very high weight. Since our method looks at incremental gain, it is quite selective in choosing examples that match P and the redundant examples have low likelihood of being selected. This also makes the algorithm efficient as observed in the experiments. The estimated weights can subsequently be used to weigh the loss of each example during training, and implementations of most popular classifiers [13], [14] can accommodate this when a weighted dataset is passed as input to them.

C. Theoretical Guarantees

In this section we prove constant factor bounds for our algorithm (Theorem III.3) in terms of optimizing the objective in equation 3, where we select potential positive examples from U and learn non-negative weights indicative of their importance in representing P. To accomplish this, we first need to define the concept of restricted strongly concave (RSC) and restricted smooth (RSM) functions which is done next. For better readability, we relegate all the proofs to the Appendix.

Definition 1 (RSC and RSM [15]): For any spatial dimension d, a function $l : \mathbb{R}^{d+} \to \mathbb{R}$ is said to be restricted strong concave with parameter c_Ω and restricted smooth with parameter C_Ω if $\forall x, y \in \Omega \subset \mathbb{R}^{d+}$;

$$-\frac{c_\Omega}{2}\|y-x\|_2^2 \geq l(y)-l(x)-\langle \nabla l(x), y-x\rangle \geq -\frac{C_\Omega}{2}\|y-x\|_2^2.$$

From the definition we observe that for any $l(.)$ in the RSC-RSM class of functions, its *curvature* in certain directions specified by the set Ω is both lower and upper bounded by placing appropriate bounds on the Bregman distance $l(y) - l(x) - \langle \nabla l(x), y - x\rangle$. For twice differentiable functions, RSC-RSM properties amounts to specifying lower and upper bounds on the eigenvalues of the Hessian $\nabla^2 l(x)$, holding uniformly in the subspace Ω.

Let the RSC and RSM parameters on the domain $\Omega_m = \{x : \|x\|_0 \leq m; x \geq 0\}$ of all m-sparse non-negative vectors be c_m and C_m respectively. $\mathbb{R}^{d+}$ denotes the d dimensional non-negative orthant, which is of particular interest to us since we have a non-negativity constraint on the learned weights. Also, let $\tilde{\Omega} = \{(x, y) : \|x - y\|_0 \leq 1\}$ with the corresponding smoothness parameter $\tilde{C}_1$.

With the above definition in place, we first establish that our MMD objective function in eq.(2) belongs to the RSC-RSM class of functions.

Lemma III.1 ($l(.)$ is RSC and RSM)**.** *Given a symmetric positive definite kernel matrix K, the function $l(w)$ in (2) has a positive RSC and finite RSM parameters.*

We now state the following lemma which upper bounds the increment in function value from the current set $L \subset U$ of elements selected as matching the P class distribution, when $|S|$ more elements are added to it to obtain the larger set $L \cap S$. This lemma is useful in proving our principal result stated below.

Lemma III.2. *Let c_k be the RSC constant for any two k sparse vectors x and $y \in \mathbb{R}^{d^+}$. Then for any two sets L and S with $L \cap S = \emptyset$ and $|L| + |S| = k$ we have*

$$l\left(\boldsymbol{\zeta}^{(L\cup S)}\right) - l\left(\boldsymbol{\zeta}^{(L)}\right) \leq \frac{1}{2c_k}\left\|\nabla l_S^+\left(\boldsymbol{\zeta}^{(L)}\right)\right\|^2.$$

We now state our main theoretical result which avers that the subset L_{u+} of elements from U chosen by MAPUL as belonging to P, is close to the optimal set selection L^*, where the closeness is determined by the RSC-RSM constants of the objective function $l(.)$.

Theorem III.3. *If L_{u+} be the $m_1 \leq m$ sparse set selected by our Algorithm 1 and L^* is the optimal m sparse set maximizing (3), then*

$$f\left(L_{u+}\right) \geq f\left(L^*\right)\left[1 - e^{-\frac{c_{2m}}{\tilde{C}_1}}\right], \tag{4}$$

where c_{2m} and $\tilde{C}_1$ are the RSC and RSM parameters.

Given that our selected positive set along with the learned weights are close to the optimal, we now prove a generalization bound for kernel SVMs that one might train on this now labelled dataset. Note that a critical part of the algorithm is the learned non-negative weights for the positive set as well as a non-negative weight determined for the negative examples. Such weights are typically used to weight the corresponding per sample loss of the classifier to be trained. Thus, our bound has to apply to kernel SVMs trained using a weighted loss function.

Generalization Bound: Let w_x be a non-negative weight associated with an example (x, y) based on our method. Let $h : X \to Y$ denote a classification function. Then the margin can be given by $g(x) = yh(x)$. Consider a weighted version of the truncated hinge loss with margin $\rho > 0$ that kernel SVMs typically optimize:

$$\begin{aligned}\mathcal{L}(x) &= 0, \text{ if } g(x) \geq \rho \\ &= w_x, \text{ if } g(x) < 0 \\ &= w_x\left(1 - \frac{1}{\rho}g(x)\right), \text{ otherwise}\end{aligned}$$

We want to bound the expectation of this loss. We do this by deriving a Rademacher bound [16], which involves bounding the sample dependent Rademacher complexity term given by $\hat{R}_n(\mathcal{L}) = \frac{2}{n}E_\sigma\left[\sup_{g\in\mathcal{G}}\sum_{i=1}^n \sigma_i\mathcal{L}(x_i)\right]$, where $\{x_1, ..., x_n\}$ denote the training inputs, $\{\sigma_1, ..., \sigma_n\}$ are $\{+1, -1\}$ random Rademacher variables and $\mathcal{G}$ is the function class of the margin.

Lemma III.4. *If w_i denotes the weight for a training example x_i and $\varpi = \max_i w_i$, then $\hat{R}_n(\mathcal{L}) \leq \frac{2\varpi}{\rho}\hat{R}_n(\mathcal{G})$*

Lemma III.5. *If $\mathcal{H}$ denotes a set of linear hypothesis in RKHS such that $\forall h \in \mathcal{H}$ we have $h : x \to \beta \cdot \phi(x)$, where β and $\phi(.)$ are the parameter vector and kernel mapping function respectively with $||\beta|| \leq \Omega$, kernel function $k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ and $K = diag(k(x_1, x_1), ..., k(x_n, x_n))$, then we have:*

$$\hat{R}_n(\mathcal{G}) \leq \frac{2\varpi\Omega}{n}\sqrt{trace(K)}$$

Pursuant to Lemmas III.4 and III.5, we have the following theorem.

Theorem III.6. *The sample dependent Rademacher complexity $\hat{R}_n(\mathcal{L})$ for kernel SVMs can be upper bounded as follows:*

$$\hat{R}_n(\mathcal{L}) \leq \frac{4\varpi^2\Omega}{n\rho}\sqrt{trace(K)}$$

Note that the weights can be normalized and passed to a classification algorithm for training in algorithm 1 leading to the same relative importance of the examples and hence the same classifier, in which case $\varpi \leq 1$ leading to the following corollary.

Corollary III.7. *If the weights w_i for training examples x_i are normalized across all training examples and returned by algorithm 1, then the sample dependent Rademacher complexity $\hat{R}_n(\mathcal{L})$ for kernel SVMs can be upper bounded as follows:*

$$\hat{R}_n(\mathcal{L}) \leq \frac{4\Omega}{n\rho}\sqrt{trace(K)}$$

One could also possibly bound the error extending ideas from [17], where the authors bound the individual treatment error w.r.t. integral probability metrics as measured between treatment and control groups.

IV. Experiments

We conducted experiments on three public datasets Image segmentation [18], MNIST [19] and Letters [20]. The details of the datasets are in the first 4 columns of Table I. Only MNIST is an image dataset and the others are tabular.

A. Setup

We performed three kinds of studies. First, taking the whole datasets into account and roughly assigning half of the examples to each class. Second, we skewed the class priors where the positive class was much smaller than the negative class. Third, we took only certain pairs of classes and saw how the different methods performed. This is similar to previous studies [21], [7], [8]. A Support Vector Machine with radial basis function (SVM-RBF) kernel was the classifier in most of the experiments, which has been the method of choice in recent studies [21] as well as relevant to our generalization result. In Table V and Figure 4, we see results using a 3-layered fully connected neural network ($100 \times 50 \times 50$) with ReLU activations and softmax. The estimated weights for each example were used in a standard way to weight the loss of that example and then learn a classifier. This can be done by passing these weights along with the dataset to well established implementations (viz. LIBSVM, weighted cross-entropy loss in tensorflow). In all studies, statistically significant results based on a 95% confidence interval are reported.

For the first kind of study, we randomly split the datasets into 75% U, 20% testing and 5% from which the P set is created as described below. Then for Image segment which has 7 classes namely; grass, path, window, cement, foliage, sky and brickface numbered to 1 to 7 for ease of reference, we assigned the first 3 classes as negative and the last 4 as

Dataset	Features	Size	Classes	Fully labeled	MAPUL	uPU (R)	uPU (H)	bSVM	nnPU	wPU
Image Seg.	19	2310	7	24.67	**32.68**	33.94	34.84	39.82	38.92	38.73
MNIST	784	70K	10	10.02	13.51	**13.03**	13.11	15.01	13.93	13.98
Letters	16	20K	26	26.01	29.22	**28.28**	29.17	29.20	29.74	29.31

TABLE I

ABOVE WE SEE THE PERCENTAGE *misclassification rate* FOR THE DIFFERENT METHODS. A SVM-RBF USED IN RECENT STUDIES WAS THE BASE CLASSIFIER. FULLY LABELED IS THE STANDARD SUPERVISED SETUP, WHERE ALL TRAINING EXAMPLES ARE LABELED. BEST (STATISTICALLY SIGNIFICANT) PU METHOD RESULTS ARE IN BOLD FOR ALL TABLES.

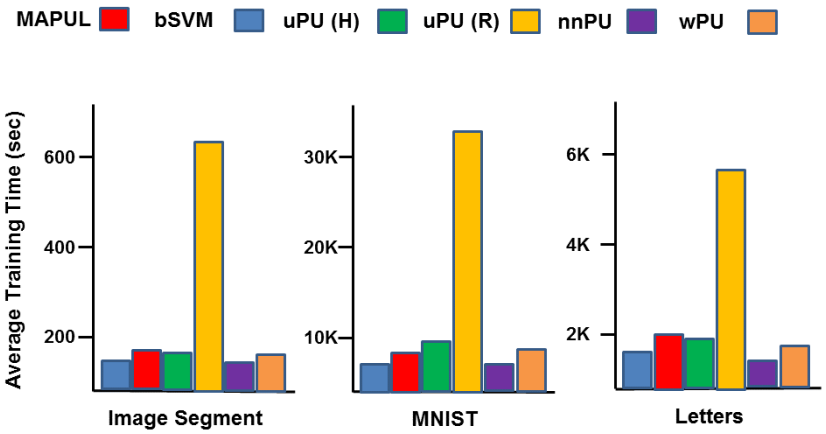

Fig. 1. Above we see the training time for the different methods using a SVM-RBF classifier on the three datasets. Note that for MAPUL (most of the) training time is a one time cost since we are learning algorithm agnostic, which is not the case with the other methods.

Dataset	# of Classes	Classes in P	Class Prior	Fully labeled	MAPUL	uPU (R)	uPU (H)	bSVM	nnPU	wPU
Image Segment	7	6, 7	0.14	1.73	**2.35**	2.63	2.78	**2.38**	2.86	2.49
MNIST	10	0, 2	0.1	3.62	4.57	**4.13**	4.91	4.53	4.97	4.56
Letters	26	A, B	0.04	0.88	**1.03**	1.18	1.26	1.28	1.23	1.25

TABLE II

ABOVE WE SEE THE *misclassification rates* WHEN THE CLASS PRIORS ARE SKEWED, WHERE (THE INDICATED) TWO CLASSES FORM THE POSITIVE CLASS WHILE THE REMAINING CLASSES FORM THE NEGATIVE CLASS.

Method	Class 2 vs 1	Class 3 vs 1	Class 4 vs 1	Class 5 vs 1	Class 6 vs 1	Class 7 vs 1
Fully labeled	0.57	2.58	2.47	2.94	0.01	0.43
MAPUL	**1.72**	6.45	**1.65***	**0.98***	**0.01**	**1.29**
uPU (R)	**1.73**	**5.16**	1.76*	1.23*	1.69	**1.31**
uPU (H)	**1.74**	6.44	1.81*	1.29*	1.74	**1.33**
bSVM	**1.72**	**5.16**	2.47	2.94	1.59	**1.29**
nnPU	**1.73**	5.43	1.84*	1.86*	1.72	**1.32**
wPU	**1.72**	**5.16**	2.47	2.94	1.08	**1.29**

TABLE III

ABOVE WE SEE THE PERCENTAGE *misclassification rate* ON IMAGE SEGMENT DATASET, WHERE ONLY TWO CLASSES ARE CONSIDERED AT A TIME WITH CLASS 1 BEING THE NEGATIVE CLASS. * INDICATES RESULTS BETTER THAN FULLY LABELED CASE.

Method	0 vs 1	0 vs 2	0 vs 3	0 vs 4	0 vs 5	0 vs 6	0 vs 7	0 vs 8	0 vs 9
Fully labeled	0.06	1.08	0.67	0.40	0.79	0.98	0.28	0.83	0.47
MAPUL	1.53	2.17	**2.42**	1.99	**3.02**	3.12	1.66	2.49	2.61
uPU (R)	**1.29**	**1.91**	2.45	**1.96**	**3.01**	2.94	**1.56**	**2.27**	2.39
uPU (H)	**1.25**	**1.92**	2.48	1.97	3.05	2.99	1.61	2.36	2.39
bSVM	1.40	2.31	2.56	1.99	3.28	**2.90**	1.80	2.74	**1.73**
nnPU	1.38	2.02	2.57	1.99	3.16	2.98	1.68	2.55	2.43
wPU	1.39	2.28	2.47	1.98	3.21	**2.90**	1.68	2.61	**1.73**

TABLE IV

ABOVE WE SEE THE PERCENTAGE *misclassification rate* ON MNIST DATASET, WHERE ONLY TWO CLASSES ARE CONSIDERED AT A TIME WITH CLASS 0 BEING THE POSITIVE CLASS.

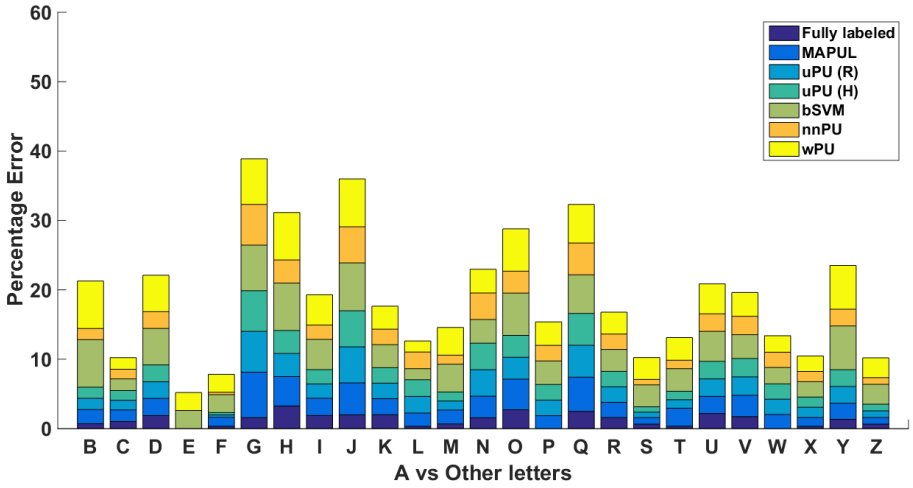

Fig. 2. Above we see the percentage *misclassification rate* for the Letters dataset, where only two classes are considered at a time with the letter A being the positive class.

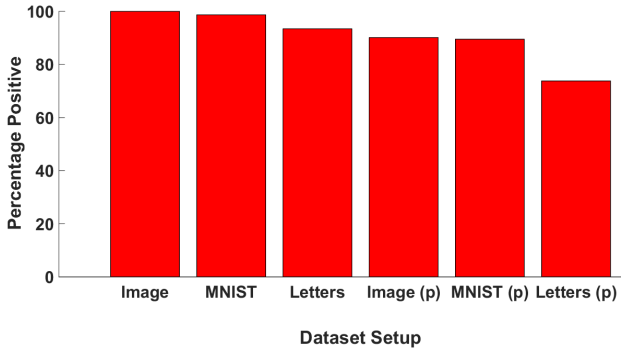

Fig. 3. Above we see the percentage of instances that MAPUL selected from U that actually belonged to the positive class. We see overall results (study 1) for the different datasets as well as (averaged) results involving the study where we considered only pairs of a classes indicated by the (p) suffix. We see above that we mostly pick the correct instances for all the datasets, with the best results for Image Segment. These results seem to correlate with our classification performance relative to the other methods on the three datasets in the respective studies.

positive. For MNIST, we assigned all the even digits to be the positive class, while the odd digits were the negative class. For Letters, we assigned the first 13 letters (i.e. A to M) to be the positive class and the remaining to be the negative class. The P datasets were created by selecting examples from the positive class in the 5% split. Results were averaged over 10 random splits.

For the second kind of study, we formed the P sets in analogous manner as above. Two classes from the original datasets were randomly selected to form the positive class, while the examples in the remaining classes formed the negative class. This creates unbalanced datasets ($\pi = 0.14, 0.1, 0.04$) with small P sets as seen in Table II.

For the third kind of study, we arbitrarily picked one class to be positive and chose each of the other classes individually to indicate negative examples. We again formed the same fraction splits as above and the same procedure to form the P sets. This was repeated 10 times. The results of this study on Image Segment and MNIST are in Tables III and IV. The results for the same study on the Letters dataset is in Figure 2. We plotted these results since a table would contain 175 values (25 (other letters) x 7 (methods)) which would be difficult for the reader to parse.

We compared our method MAPUL with other state-of-the-art methods namely; non-negative risk estimator based PU learning (nnPU) [8], unbiased PU learning with ramp loss (uPU (R)), unbiased PU learning with hinge loss (uPU (H)), weighted PU learning (wPU) [1] and biased support vector machine (bSVM). We also report results for the fully labeled case, which is the standard supervised learning setup that has access to all the labels, so that we have a measure of how far off from the best results we are likely to achieve.

For MAPUL, we used a Gaussian radial basis kernel to compute $l(.)$ whose width was set to a standard value of median of distances [9]. The threshold ϵ to detect convergence was set to 10^{-11}. Note that as our algorithm is quite selective, and hence ϵ can be set to a small value without worrying that the algorithm might choose the entire U as positives. This is another benefit of our algorithm where we obtain good quality results without needing to fine tune these parameters using a validation set. For the competing methods that need priors, we favor them by specifying the *true* data priors. All other parameters for the different methods were found by cross validation.

Our misclassification rate metric is exactly same as the one that is traditionally used and equals the average zero-one loss over the test set ($\times$ 100), i.e., if $\lambda(.,.)$ is a zero-one loss function and $\zeta(.)$ is a classifier, misclassification rate for the classifier over a dataset $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ is $\frac{100}{n} \sum_{i=1}^{n} \lambda(\zeta(x_i), y_i)$.

Dataset	Features	Size	Classes	Fully labelled	MAPUL	uPU (R)	uPU (H)	nnPU	wPU
Image Seg.	19	2310	7	21.43	**28.34**	30.83	30.92	37.92	37.87
MNIST	784	70K	10	1.32	**3.16**	**3.15**	3.97	3.99	3.98
Letters	16	20K	26	21.72	23.52	**22.82**	23.57	24.02	23.51

TABLE V

ABOVE, WE SEE THE PERCENTAGE *misclassification rate* FOR THE DIFFERENT METHODS. A 3-LAYER FULLY CONNECTED NEURAL NETWORK WAS THE BASE CLASSIFIER. FULLY LABELED IS THE STANDARD SUPERVISED SETUP, WHERE ALL TRAINING EXAMPLES ARE LABELED. BIASED SVM IS NOT APPLICABLE HERE AS THESE RESULTS USE A NEURAL NETWORK CLASSIFIER.

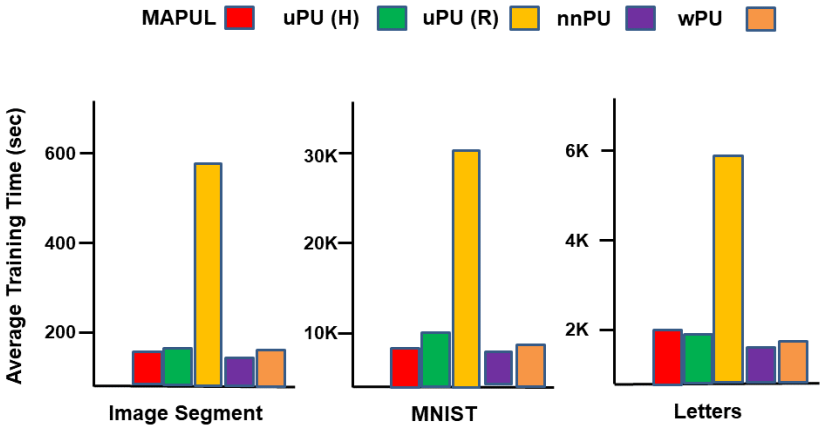

Fig. 4. Above we see the training time for the different methods using a neural network classifier on the three datasets. Note that for MAPUL (most of the) training time is a one time cost since we are learning algorithm agnostic, which is not the case with the other methods.

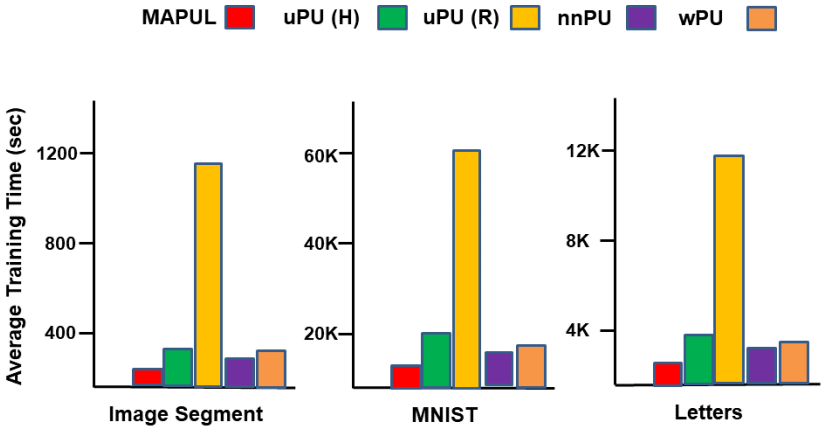

Fig. 5. Above we see the training time for the different methods on the three datasets that is a sum total of first training a SVM-RBF classifier and then the neural network classifier. The important thing to note here is the advantage of MAPUL where the time is more or less similar to training just one classifier since, the identified (and weighted) positive and negative sets have to be computed just once and can be used to train both the SVM-RBF as well as the neural network. This is not the case for the other methods.

B. Observations

We see in Table I that our method is the best on Image Segment while less than a percent worse than the best on the other datasets. We also see in Figure 1 that our method is quite efficient, not to mention that for a particular dataset, the identification of potential positive examples from U and learning weights for all the examples is a one time cost.

In Table II, we see that our method performs well even when the datasets are skewed and the size of the P set is small. Here our method performs the best on Image Segment and Letters, while being competitive with the other methods on MNIST. These two studies are a testament to the robustness and efficiency of our approach.

In Tables III and IV and Figure 2, we show the results for the pairwise studies where only two classes were considered. We observe on Image Segment that we have state-of-the-art performance in five of the six cases. In fact, in two cases (columns 4 and 5) we are *better than even fully labeled supervised learning*. We conjecture that our learned weights in these cases must be informative, where assigning uniform

weight to all examples of P forces the classifier to fit outliers or hard examples that adversely affects its generalizability. This intuition is also consistent when we observe the performance of the other methods, where the uPU methods are closer in performance to us, while biased SVM and wPU is no better than the fully labeled case.

On MNIST in Table IV, uPU(R) does quite well with state-of-the-art performance in six of the nine cases. However, here too MAPUL is competitive, with state-of-the-art performance in two cases and within 0.3% of the best in all the other cases. On Letters in Figure 2, we also observe similar qualitative results, where we have state-of-the-art performance in 9 cases (E, K, M, Q, R, S, U, V and Z), while we are competitive with the best (within a couple of percent) in all others.

We also computed the percentage of positive examples picked by MAPUL from U. The percentages are upward of 90% (98.12% for Image Segment, 96.89% for MNIST and 91.13% for Letters), which shows that we mostly pick the correct examples. These numbers also correlate with the observed performance. A plot indicating the percentage of positive examples picked by MAPUL from U that are actually positive for the overall and pairwise studies are shown in Figure 3. We observe that we mostly pick the correct examples and our classification performance relative to the other methods reported here seems to correlate with the percentage we pick correctly in the respective studies.

In Table V, we observe that even with a neural network classifier the results are qualitatively similar to those with SVM-RBF. Given the main benefits of our method namely, being model agnostic and not requiring priors or distributional assumptions as outlined before, these results where we have comparable (or better) performance to the state-of-the-art strongly motivates MAPUL. Moreover, in Figure 5 we see the computational advantage of MAPUL where the now labelled and weighted dataset using MAPUL can be used to train both the SVM-RBF as well as the neural network classifier. Other methods require relearning the weights (and labels) for each classifier.

V. Time Complexity

As seen in the experiments, our method is highly competitive w.r.t. computational time, given that it is a one time cost independent of the particular classifier to identify the positive examples and the example weights, which is not the case for any of the other methods. This is apparent from Figure 5. The point is once the positives are selected and weights determined by our method, it can be used by many classifiers directly. This is the primary reason for MAPUL's low computational time complexity. Thus, in general more the classifiers that are trained more significant the difference in training times will be.

If $n_+ = |P|$, $n_u = |U|$ and m the number of examples identified by MAPUL as positive from U, then the time complexity is $O(n_+n_u + n_um^2 + m^4)$, where $m << n_u$ as our method is highly selective in choosing examples that match the P class even for small ϵ, making the learned weights critical for performance. The reported times include training of the classifier and as mentioned before in our case finding the positives and learning the weights is a one time cost.

VI. Related Work and Discussion

PU learning can be dated back to the late 90s [22], [23], [24] where it was mostly of theoretical interest. However, since then it has been heavily studied as it finds use in multiple applications ranging from remote sensing [5], to novelty detection [4], to even matrix completion [2]. It has also applications in text categorization and information retrieval [3]. Recently, a multiclass extension of PU learning was introduced [21] for applications in predicting multiple classes rather than just positive or negative. There are mainly two settings, case-control and censoring PU setting [12]. In the first, U is assumed to be sampled from $p(x)$, which may not be the case for the latter. Although we have applied our method to the first setting, it can be readily applied to the latter. This is an additional advantage of MAPUL as the other methods are typically designed either for one or the other.

An early effective solution for this problem was to model it as a cost sensitive SVM classification problem [6], where the examples in P are weighted high, while the examples in U are considered to belong to the negative class and are weighted lower. The lower weight acknowledges the existence of positive examples in U allowing the classifier to classify examples in U that are most definitely positive as positive without too much penalty. One of the paradigm shifts in the proposed approaches was [1], which showed that under a random selection assumption, the probability of classifying an example into the positive class is just a constant scaling of the probability of classifying it as part of U as opposed to P. This led to an approach which considered examples in U as both positive and negative but with different weights. The weights were functions of the classification algorithm and would be tuned using a validation set. One of the unrealistic assumptions of this work was that the weights for the examples in P were assumed to be all one, which may not be the case. This has led to the recent state-of-the-art unbiased PU learning [7] which learns weights for all examples and can be shown to be an unbiased estimator of the risk. For rich hypothesis classes, an improvement based on a non-negative risk estimator has been proposed [8]. However, in all these methods the underlying algorithm is dependent on how they weight the samples. This can be computationally expensive as the weights have to be tuned using a validation set. Moreover, they require knowledge or estimation of the prior distribution of examples in U, which may or may not be easy to obtain.

Our work moves away from these limitations as we do not require knowledge of the prior distribution, nor are our weights and identified positive examples classifier dependent. Thus, computing weights and identifying examples is a one time cost. In addition, they can be passed as input to any classification algorithm that can train on weighted inputs. On the theoretical side, we prove approximation guarantees in terms of how well the identified examples in U match the target P set. Since our

objective is RSC and RSM, it would be weakly submodular and hence a greedy algorithm that performs function evaluations should provide a constant factor guarantee [25]. However, our method involves only gradient evaluations and not function evaluations which makes it highly efficient [11]. Moreover, we prove a constant factor bound even with the non-negativity constraint that would be the same as the standard greedy [25], thus not loosing out even in performance, which may be a result of independent interest. We also empirically see the benefit of our method where it is competitive with the best even though it is model agnostic.

In the future, it would be interesting to extend our methodology to multi-PU learning. We could fit U independently to each class in P and obtain candidates along with their weights. Here however, we would need a way to handle examples that were identified to belong to two or more classes, i.e., examples that overlap and somehow resolve them in a reasonable fashion. It would be nice in this case too, to have some theoretically strong arguments to justify our decisions.

APPENDIX

A. Proof of Lemma III.1

For the concave function $l(w) = -\frac{1}{2}w^T K w + w^T \mu_p$, we calculate $l(w_1) - l(w_2) - \nabla\langle l(w_2), w_1 - w_2\rangle = -0.5(w_1 - w_2)^T K (w_1 - w_2)$. If w_1 and w_2 are k_1 and k_2 sparse vectors respectively, then $\Delta w = w_1 - w_2$ has a *maximum* of $k \leq k_1 + k_2$ non-zero entries. For the constants c and C satisfying $-c\|\Delta w\|^2 \geq -\Delta w^T K \Delta w \geq -C\|\Delta w\|^2$ we obtain the bounds: $c \geq k$-sparse smallest eigen value of K and $C \leq k$-sparse largest eigen value of K. In particular, when $supp(w_2) \subset supp(w_1)$, $\|\Delta w\|_0 \leq k_1$ providing tighter bounds for c and C. □

B. Proof of Lemma III.2

By the definition of RSC constant c_k we find

$$
\begin{aligned}
& l\left(\boldsymbol{\zeta}^{(L\cup S)}\right) - l\left(\boldsymbol{\zeta}^{(L)}\right) \\
& \leq \left\langle \nabla l\left(\boldsymbol{\zeta}^{(L)}\right), \boldsymbol{\zeta}^{(L\cup S)} - \boldsymbol{\zeta}^{(L)} \right\rangle - \frac{c_k}{2}\left\|\boldsymbol{\zeta}^{(L\cup S)} - \boldsymbol{\zeta}^{(L)}\right\|^2 \\
& \leq \max_{\mathbf{v}:\mathbf{v}_{(L\cup S)^c}=0, \mathbf{v}>=0} \left\langle \nabla l\left(\boldsymbol{\zeta}^{(L)}\right), \mathbf{v} - \boldsymbol{\zeta}^{(L)} \right\rangle - \frac{c_k}{2}\left\|\mathbf{v} - \boldsymbol{\zeta}^{(L)}\right\|^2 .
\end{aligned} \tag{5}
$$

Observe that the KKT conditions at the optimum $\boldsymbol{\zeta}^{(L)}$ for the function $f(L)$ necessitates that $\forall j \in L$,

$$
\begin{aligned}
\boldsymbol{\zeta}_j^{(L)} > 0 \implies \nabla l_j\left(\boldsymbol{\zeta}^{(L)}\right) = 0 \text{ and} \\
\boldsymbol{\zeta}_j^{(L)} = 0 \implies \nabla l_j\left(\boldsymbol{\zeta}^{(L)}\right) \leq 0
\end{aligned}
$$

and hence we have $\mathbf{v}_j = \boldsymbol{\zeta}_j^{(L)}$. When $j \in S$, $\boldsymbol{\zeta}_j^{(L)} = 0$, and maximizing w.r.t. $\mathbf{v}_j$, the maximum occurs at $\mathbf{v}_j = \frac{\nabla l_j^+\left(\boldsymbol{\zeta}^{(L)}\right)}{c_k}$ where $\nabla l_j^+\left(\boldsymbol{\zeta}^{(L)}\right) = \max\left(\nabla l_j\left(\boldsymbol{\zeta}^{(L)}\right), 0\right)$. Plugging this maximum value of $\mathbf{v}$ in (5) we get our result. □

C. Proof of Theorem III.3

Let $L = L_i$ be the set of examples chosen by our algorithm 1 from U up to the iteration i such that the final set after exiting the while loop in m iterations is $L_{u+} = L_m$. Define the residual set $L_R = L^* \setminus L$ and let $D(i+1) = f(L \cup \{v\}) - f(L)$ where v is the index that would be selected in the next step. Defining $\mathbf{y}^{(\{v\})} = \boldsymbol{\zeta}^{(L)} + \alpha \mathbf{1}^{(\{v\})}$ for some $\alpha \geq 0$ and recalling that $\boldsymbol{\zeta}^{(L\cup\{v\})}$ is the maximizing point for $f(L \cup \{v\})$ we get

$$
\begin{aligned}
D(i+1) &\geq l\left(\mathbf{y}^{(\{v\})}\right) - l\left(\boldsymbol{\zeta}^{(L)}\right) \\
&\geq \left\langle \nabla l\left(\boldsymbol{\zeta}^{(L)}\right), \alpha \mathbf{1}^{(\{v\})} \right\rangle - \frac{\tilde{C}_1}{2}\alpha^2.
\end{aligned}
$$

Setting $\alpha = \frac{\nabla l_v^+\left(\boldsymbol{\zeta}^{(L)}\right)}{\tilde{C}_1}$ we have

$$
\begin{aligned}
D(i+1) &\geq \frac{1}{2\tilde{C}_1}\left[\nabla l_v^+\left(\boldsymbol{\zeta}^{(L)}\right)\right]^2 \\
&\geq \frac{1}{2m\tilde{C}_1}\sum_{j\in L_R}\left[\nabla l_j^+\left(\boldsymbol{\zeta}^{(L)}\right)\right]^2
\end{aligned}
$$

where the last inequality follows from recalling that our algorithm chooses the coordinate v that maximizes the gradient value $\nabla l\left(\boldsymbol{\zeta}^{(L)}\right)$ and $|L_R| \leq m$. Letting $k = |L| + |L_R|$, $B(i) = f(L^*) - f(L)$ and setting $S = L_R$ in (4) we find

$$
mD(i+1) \geq \frac{c_k}{\tilde{C}_1}[f(L \cup L_R) - f(L)] \geq \frac{c_{2m}}{\tilde{C}_1}B(i)
$$

where use the inequalities that $c_k \geq c_{2m}$ as $k \leq 2m$ and $L^* \subseteq L \cup L_R$. Setting $\kappa = \frac{c_{2m}}{\tilde{C}_1 m}$ and noting that $D(i+1) = B(i) - B(i+1)$ we get the recurrence relation $B(i+1) \leq (1-\kappa)B(i)$ which when iterated i times starting from step 0 gives $B(i) \leq (1-\kappa)^i B(0)$. Plugging in $B(m) = f(L^*) - f(L_{u+})$ and $B(0) = f(L^*)$ gives us the required inequality

$$
f(L_{u+}) \geq f(L^*)\left[1 - (1-\kappa)^m\right] \geq f(L^*)\left[1 - e^{-\frac{c_{2m}}{\tilde{C}_1}}\right].
$$

□

D. Proof of Lemma III.4

This follows from our observation that the loss function $\mathcal{L}$ is $\frac{\varpi}{\rho}$-Lipschitz. Hence, by applying Talagrands lemma [26] we get the result. □

E. Proof of Lemma III.5

The derivation uses Cauchy-Schwartz inequality, Jensen's inequality and noticing that expectation of $\sigma_i\sigma_j$ is zero for

$i \neq j$.

$$\begin{aligned}
\hat{R}_n(\mathcal{G}) &= \frac{2}{n} E_\sigma \left[\sup_{||\beta|| \leq \Omega} \sum_{i=1}^{n} \sigma_i w_i \beta \cdot \phi(x_i) \right] \\
&= \frac{2}{n} E_\sigma \left[\sup_{||\beta|| \leq \Omega} \beta \cdot \sum_{i=1}^{n} \sigma_i w_i \phi(x_i) \right] \\
&\leq \frac{2\Omega}{n} E_\sigma \left[|| \sum_{i=1}^{n} \sigma_i w_i \phi(x_i) || \right] \\
&\leq \frac{2\Omega}{n} \left(E_\sigma \left[\sum_{i,j=1}^{n} \sigma_i \sigma_j w_i w_j k(x_i, x_j) \right] \right)^{\frac{1}{2}} \\
&= \frac{2\Omega}{n} \left(\sum_{i=1}^{n} w_i^2 k(x_i, x_i) \right)^{\frac{1}{2}} \\
&\leq \frac{2\varpi\Omega}{n} \sqrt{trace(K)}
\end{aligned}$$

□

REFERENCES

[1] 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*, 2008.

[2] C.-J. Hsieh, N. Natarajan, and I. S. Dhillon, "Pu learning for matrix completion," in *ICML*, 2015.

[3] B. Liu, W. S. Lee, P. S. Yu, and X. Li, "Partially supervised classification of text documents," in *ICML*, 2002.

[4] G. Blanchard, G. Lee, and C. Scott, "Semi-supervised novelty detection," in *Journal of Machine Learning Research*, 2010, p. 29733009.

[5] W. Li, Q. Guo, and C. Elkan, "A positive and unlabeled learning algorithm for oneclass classification of remote-sensing data," in *IEEE Trans. on Geoscience and Remote Sensing*, 2011.

[6] 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 Third IEEE International Conference on Data Mining*, 2003.

[7] D. Plessis, M. C, G. Niu, and M. Sugiyama, "Analysis of learning from positive and unlabeled data," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 703–711.

[8] R. Kiryo, G. Niu, D. Plessis, M. C, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 1675–1685.

[9] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A Kernel Method for the Two-Sample-Problem," in *Journal of Machine Learning Research*, 2008, pp. 723–773.

[10] M. C. Plessis, G. Niu, and M. Sugiyama, "Class-prior estimation for learning from positive and unlabeled data," *Mach. Learn.*, vol. 106, no. 4, pp. 463–492, Apr. 2017.

[11] S. Boyd and L. Vandenberghe, "Convex optimization," in *Cambridge University Press*, 2004.

[12] A. Menon, B. V. Rooyen, C. S. Ong, and B. Williamson, "Learning from corrupted binary labels via class-probability estimation," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, pp. 125–134.

[13] C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 2011.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[15] H. Zhang, "The restricted strong convexity revisited: Analysis of equivalence to error bound and quadratic growth," in *https://arxiv.org/abs/1511.01635*, 2016.

[16] P. L. Bartlett and S. Mendelson, "Rademacher and gaussian complexities: Risk bounds and structural results," *J. Mach. Learn. Res.*, 2003.

[17] U. Shalit, F. D. Johansson, and D. Sontag, "Estimating individual treatment effect: generalization bounds and algorithms," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[18] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[20] P. W. Frey and D. J. Slate, "Letter recognition using holland-style adaptive classifiers," *Machine Learning*, vol. 6, no. 2, 1991.

[21] Y. Xu, C. Xu, C. Xu, and D. Tao, "Multi-positive and unlabeled learning," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.

[22] F. Denis, "Pac learning from positive statistical queries," in *ALT*, 1998.

[23] F. D. Comit, F. Denis, R. Gilleron, and F. Letouzey, "Positive and unlabeled examples help learning," in *ALT*, 1999.

[24] F. Letouzey, F. Denis, and R. Gilleron, "Learning from positive and unlabeled examples," in *ALT*, 2000.

[25] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An Analysis of Approximations for Maximizing Submodular Set Functions," *Math. Program.*, vol. 14, pp. 265–294, December 1978.

[26] M. Talagrand, "Concentration of measure and isoperimetric inequalities in product spaces," *Mathematiques de L'IHES*, 1995.