∝SVM for Learning with Label Proportions

Felix X. Yu[†] YUXINNAN@EE.COLUMBIA.EDU
Dong Liu[†] DONGLIU@EE.COLUMBIA.EDU
Sanjiv Kumar[§] SANJIVK@GOOGLE.COM
Tony Jebara[†] JEBARA@CS.COLUMBIA.EDU
Shih-Fu Chang[†] SFCHANG@CS.COLUMBIA.EDU
[†]Columbia University, New York, NY 10027
[§]Google Research, New York, NY 10011

Abstract

We study the problem of learning with label proportions in which the training data is provided in groups and only the proportion of each class in each group is known. We propose a new method called proportion-SVM, or ∝SVM, which explicitly models the latent unknown instance labels together with the known group label proportions in a large-margin framework. Unlike the existing works, our approach avoids making restrictive assumptions about the data. The ∝SVM model leads to a non-convex integer programming problem. In order to solve it efficiently, we propose two algorithms: one based on simple alternating optimization and the other based on a convex relaxation. Extensive experiments on standard datasets show that ∝SVM outperforms the state-of-the-art, especially for larger group sizes.

1. Introduction

The problem of learning with label proportions has recently drawn attention in the learning community (Quadrianto et al., 2009; Rüeping, 2010). In this setting, the training instances are provided as groups or "bags". For each bag, only the proportions of the labels are available. The task is to learn a model to predict labels of the individual instances.

Learning with label proportions raises multiple issues. On one hand, it enables interesting applications such as modeling voting behaviors from aggregated proportions across different demographic regions. On the other hand, the feasibility of such a learning method also raises concerns about the sensitive personal information that could potentially be leaked simply by observing label proportions.

To address this learning setting, this article explicitly models the unknown instance labels as latent variables. This alleviates the need for making restrictive assumptions on the data, either parametric or generative. We introduce a large-margin framework called proportion-SVM, or ∝SVM[1], which jointly optimizes over the unknown instance labels and the known label proportions (Section 3). In order to solve ∝SVM efficiently, we propose two algorithms - one based on simple alternating optimization (Section 4), and the other based on a convex relaxation (Section 5). We show that our approach outperforms the existing methods for various datasets and settings (Section 6). The gains are especially higher for more challenging settings when the bag size is large.

2. Related Works

MeanMap: Quadrianto et al. (2009) proposed a theoretically sound method to estimate the mean of each class using the mean of each bag and the label proportions. These estimates are then used in a conditional exponential model to maximize the log likelihood. The key assumption in MeanMap is that the class-conditional distribution of data is independent of the bags. Unfortunately, this assumption does not hold for many real world applications. For example, in modeling voting behaviors, in which the bags are different demographic regions, the data distribution can be highly dependent on the bags.

Inverse Calibration (InvCal): Rüeping (2010) pro-

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28.

[1] ∝ is the symbol for "proportional-to".

posed treating the mean of each bag as a "super-instance", which was assumed to have a soft label corresponding to the label proportion. The "super-instances" can be poor in representing the properties of the bags. Our work also utilizes a large-margin framework, but we explicitly model the instance labels. Section 3.3 gives a detailed comparison with InvCal.

Figure 1 provides a toy example to highlight the problems with MeanMap and InvCal, which are the state-of-the-art methods.

Related Learning Settings: In semi-supervised learning, Mann & McCallum (2007) and Bellare et al. (2009) used an expectation regularization term to encourage model predictions on the unlabeled data to match the given proportions. Similar ideas were also studied in the generalized regularization method (Gillenwater et al., 2011). Li et al. (2009a) proposed a variant of semi-supervised SVM to incorporate the label mean of the unlabeled data. Unlike semi-supervised learning, the learning setting we are considering requires no instance labels for training.

As an extension to multiple-instance learning, Kuck & de Freitas (2005) designed a hierarchical probabilistic model to generate consistent label proportions. Besides the inefficiency in optimization, the method was shown to be inferior to MeanMap (Quadrianto et al., 2009). Similar ideas have also been studied by Chen et al. (2006) and Musicant et al. (2007).

Stolpe & Morik (2011) proposed an evolutionary strategy paired with a labeling heuristic for clustering with label proportions. Different from clustering, the proposed ∝SVM framework jointly optimizes the latent instance labels and a large-margin classification model. The ∝SVM formulation is related to large-margin clustering (Xu et al., 2004), with an additional objective to utilize the label proportions. Specifically, the convex relaxation method we used is inspired by the works of Li et al. (2009a) and Xu et al. (2004).

3. The ∝SVM Framework

3.1. Learning Setting

We consider a binary learning setting as follows. The training set $\{\mathbf{x}_i\}_{i=1}^N$ is given in the form of K bags,

$$\{\mathbf{x}_i | i \in \mathcal{B}_k\}_{k=1}^K, \quad \cup_{k=1}^K \mathcal{B}_k = \{1 \cdots N\}. \tag{1}$$

In this paper, we assume that the bags are disjoint, *i.e.*, $\mathcal{B}_k \cap \mathcal{B}_l = \emptyset$, $\forall k \neq l$. The k-th bag is with label proportion p_k:

$$\forall_{k=1}^K, \quad p_k := \frac{|\{i | i \in \mathcal{B}_k, y_i^* = 1\}|}{|\mathcal{B}_k|}, \tag{2}$$

in which $y_i^* \in \{1, -1\}$ denotes the *unknown* ground-truth label of $\mathbf{x}_i$, $\forall_{i=1}^N$. We use $f(x) = \text{sign}(\mathbf{w}^T \varphi(\mathbf{x}) + b)$ for predicting the binary label of an instance $\mathbf{x}$, where $\varphi(\cdot)$ is a map of the input data.

3.2. Formulation

We explicitly model the unknown instance labels as $\mathbf{y} = (y_1, \cdots, y_N)^T$, in which $y_i \in \{-1, 1\}$ denotes the unknown label of $\mathbf{x}_i$, $\forall_{i=1}^N$. Thus the label proportion of the k-th bag can be straightforwardly modeled as

$$\tilde{p}_k(\mathbf{y}) = \frac{|\{i | i \in \mathcal{B}_k, y_i = 1\}|}{|\mathcal{B}_k|} = \frac{\sum_{i \in \mathcal{B}_k} y_i}{2|\mathcal{B}_k|} + \frac{1}{2}. \tag{3}$$

We formulate the ∝SVM under the large-margin framework as below.

$$\min_{\mathbf{y}, \mathbf{w}, b} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^N L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i) + b) + C_p \sum_{k=1}^K L_p\left(\tilde{p}_k(\mathbf{y}), p_k\right)$$
$$\text{s.t.} \ \ \forall_{i=1}^N, \ \ y_i \in \{-1, 1\}, \tag{4}$$

in which $L(\cdot) \geq 0$ is the loss function for classic supervised learning. $L_p(\cdot) \geq 0$ is a function to penalize the difference between the true label proportion and the estimated label proportion based on $\mathbf{y}$. The task is to simultaneously optimize the labels $\mathbf{y}$ and the model parameters $\mathbf{w}$ and b.

The above formulation permits using different loss functions for $L(\cdot)$ and $L_p(\cdot)$. One can also add weights for different bags. Throughout this paper, we consider $L(\cdot)$ as the hinge loss, which is widely used for large-margin learning: $L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i) + b) = \max\left(0, 1 - y_i(\mathbf{w}^T\varphi(\mathbf{x}_i) + b)\right)$. The algorithms in Section 4 and Section 5 can be easily generalized to different $L(\cdot)$.

Compared to (Rüeping, 2010; Quadrianto et al., 2009), ∝SVM requires no restrictive assumptions on the data. In fact, in the special case where no label proportions are provided, ∝SVM becomes large-margin clustering (Xu et al., 2004; Li et al., 2009a), whose solution depends only on the data distribution. ∝SVM can naturally incorporate any amount of supervised data without modification. The labels for such instances will be observed variables instead of being hidden. ∝SVM can be easily extended to the multi-class case, similar to (Keerthi et al., 2012).

3.3. Connections to InvCal

As stated in Section 2, the Inverse Calibration method (InvCal) (Rüeping 2010) treats the mean of each bag as a "super-instance", which is assumed to have a soft label corresponding to the label proportion. It is for-

mulated as below.

$$\begin{aligned}\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_p\sum_{k=1}^{K}(\xi_k+\xi_k^*) \\ \forall_{k=1}^K, \quad & \xi_k \geq 0, \quad \xi_k^* \geq 0 \\ \forall_{k=1}^K, \quad & \mathbf{w}^T\boldsymbol{m}_k + b \geq -\log(\frac{1}{p_k}-1) - \epsilon_k - \xi_k \\ & \mathbf{w}^T\boldsymbol{m}_k + b \leq -\log(\frac{1}{p_k}-1) + \epsilon_k + \xi_k^*,\end{aligned} \tag{5}$$

in which the k-th bag mean is $\boldsymbol{m}_k = \frac{1}{|\mathcal{B}_k|}\sum_{i\in\mathcal{B}_k}\varphi(\mathbf{x}_i)$, $\forall_{k=1}^K$. Unlike ∝SVM, the proportion of the k-th bag is modeled on top of this "super-instance" $\boldsymbol{m}_k$ as:

$$q_k := \left(1+\exp\left(-\mathbf{w}^T\boldsymbol{m}_k + b\right)\right)^{-1}. \tag{6}$$

The second term of the objective function (5) tries to impose $q_k \approx p_k$, $\forall_{k=1}^K$, albeit in an inverse way.

Though InvCal is shown to outperform other alternatives, including MeanMap (Quadrianto et al., 2009) and several simple large-margin heuristics, it has a crucial limitation. Note that (6) is not a good way of measuring the proportion predicted by the model, especially when the data has high variance, or the data distribution is dependent on the bags. In our formulation (4), by explicitly modeling the unknown instance labels $\mathbf{y}$, the label proportion can be directly modeled as $\tilde{p}_k(\mathbf{y})$ given in (3). The advantage of our method is illustrated in a toy experiment shown in Figure 1 (for details see Section 6.1).

3.4. Difficulties in Solving ∝SVM

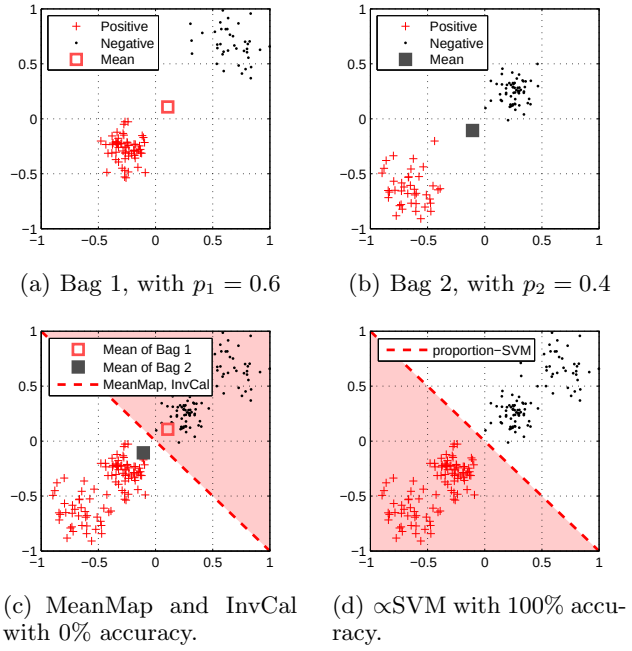
The ∝SVM formulation is fairly intuitive and straightforward. It is, however, a non-convex integer programming problem, which is NP-hard. Therefore, one key issue lies in how to find an efficient algorithm to solve it approximately. In this paper, we provide two solutions: a simple alternating optimization method (Section 4), and a convex relaxation method (Section 5).

4. The alter-∝SVM Algorithm

In ∝SVM, the unknown instance labels $\mathbf{y}$ can be seen as a bridge between supervised learning loss and label proportion loss. Therefore, one natural way for solving (4) is via alternating optimization as,

- For a fixed $\mathbf{y}$, the optimization of (4) *w.r.t* $\mathbf{w}$ and b becomes a classic SVM problem.
- When $\mathbf{w}$ and b are fixed, the problem becomes:

$$\begin{aligned}\min_{\mathbf{y}} \quad & \sum_{i=1}^{N} L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i)+b) + \frac{C_p}{C}\sum_{k=1}^{K} L_p\left(\tilde{p}_k(\mathbf{y}), p_k\right) \\ \text{s.t.} \quad & \forall_{i=1}^N, \quad y_i \in \{1,-1\}.\end{aligned} \tag{7}$$



(a) Bag 1, with $p_1 = 0.6$ (b) Bag 2, with $p_2 = 0.4$

(c) MeanMap and InvCal with 0% accuracy. (d) ∝SVM with 100% accuracy.

Figure 1. An example of learning with two bags to illustrate the drawbacks of the existing methods. (a) Data of bag 1. (b) Data of bag 2. (c) Learned separating hyperplanes of MeanMap and InvCal. (d) Learned separating hyperplane of ∝SVM (either alter-∝SVM or conv-∝SVM). More details are given in Section 6.1. Note that the algorithms do not have access to the individual instance labels.

We show that the second step above can be solved efficiently. Because the influence of each bag $\{y_i | i \in \mathcal{B}_k\}$, $\forall_{k=1}^K$ on the objective is independent, we can optimize (7) on each bag separately. In particular, solving $\{y_i | i \in \mathcal{B}_k\}$ yields the following optimization problem:

$$\begin{aligned}\min_{\{y_i|i\in\mathcal{B}_k\}} \quad & \sum_{i\in\mathcal{B}_k} L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i)+b) + \frac{C_p}{C} L_p\left(\tilde{p}_k(\mathbf{y}), p_k\right) \\ \text{s.t.} \quad & \forall i \in \mathcal{B}_k, \quad y_i \in \{1,-1\}.\end{aligned} \tag{8}$$

Proposition 1 *For a fixed $\tilde{p}_k(\mathbf{y}) = \theta$, (8) can be optimally solved by the steps below.*

- Initialize $y_i = -1$, $i \in \mathcal{B}_k$. The optimal solution can be obtained by flipping the signs as below.
- By flipping the sign of y_i, $i \in \mathcal{B}_k$, suppose the reduction of the first term in (8) is δ_i. Sort δ_i, $i \in \mathcal{B}_k$. Then flip the signs of the top-R y_i's which have the highest reduction δ_i. $R = \theta|\mathcal{B}_k|$.

For bag $\mathcal{B}_k$, we only need to sort the corresponding δ_i, $i \in \mathcal{B}_k$ once. Sorting takes $\mathcal{O}(|\mathcal{B}_k|\log(|\mathcal{B}_k|))$ time. After that, for each $\theta \in \{0, \frac{1}{|\mathcal{B}_k|}, \frac{2}{|\mathcal{B}_k|}, \cdots, 1\}$, the optimal solution can be computed incrementally, each taking $\mathcal{O}(1)$ time. We then pick the solution with the smallest objective value, yielding the optimal solution of (8).

Algorithm 1 alter-∝SVM

Randomly initialize $y_i \in \{-1, 1\}$, $\forall_{i=1}^N$. $C^* = 10^{-5}C$.
while $C^* < C$ **do**
  $C^* = \min\{(1+\Delta)C^*, C\}$
  repeat
    Fix $\mathbf{y}$ to solve $\mathbf{w}$ and b.
    Fix $\mathbf{w}$ and b to solve $\mathbf{y}$ (Eq. (7) with $C \leftarrow C^*$).
  until The decrease of the objective is smaller than a threshold (10^{-4})
end while

Proposition 2 *Following the above steps, (7) can be solved in $\mathcal{O}(N\log(J))$ time, $J = \max_{k=1\cdots K}|\mathcal{B}_k|$.*

The proofs of the above propositions are given in the supplementary material.

By alternating between solving ($\mathbf{w}$, b) and $\mathbf{y}$, the objective is guaranteed to converge. This is due to the fact that the objective function is lower bounded, and non-increasing. In practice, we terminate the procedure when the objective no longer decreases (or if its decrease is smaller than a threshold). Empirically, the alternating optimization typically terminates fast within tens of iterations, but one obvious problem is the possibility of local solutions.

To alleviate this problem, similar to T-SVM (Joachims, 1999; Chapelle et al., 2008), the proposed alter-∝SVM algorithm (Algorithm 1) takes an additional annealing loop to gradually increase C. Because the nonconvexity of the objective function mainly comes from the second term of (4), the annealing can be seen as a "smoothing" step to protect the algorithm from sub-optimal solutions. Following (Chapelle et al., 2008), we set $\Delta = 0.5$ in Algorithm 1 throughout this work. The convergence and annealing are further discussed in the supplementary material.

In the implementation of alter-∝SVM, we consider $L_p(\cdot)$ as the absolute loss: $L_p(\tilde{p}_k(\mathbf{y}), p_k) = |\tilde{p}_k(\mathbf{y}) - p_k|$. Empirically, each alter-∝SVM loop given an annealing value C^* terminates within a few iterations. From Proposition 2, optimizing $\mathbf{y}$ has linear complexity in N (when J is small). Therefore the overall complexity of the algorithm depends on the SVM solver. Specifically, when linear SVM is used (Joachims, 2006), alter-∝SVM has linear complexity. In practice, to further alleviate the influence of the local solutions, similar to clustering, *e.g.*, kmeans, we repeat alter-∝SVM multiple times by randomly initializing $\mathbf{y}$, and then picking the solution with the smallest objective value.

5. The conv-∝SVM Algorithm

In this section, we show that with proper relaxation of the ∝SVM formulation (4), the objective function can be transformed to a convex function of $\boldsymbol{M} := \mathbf{y}\mathbf{y}^T$. We then relax the solution space of $\boldsymbol{M}$ to its convex hull, leading to a convex optimization problem of $\boldsymbol{M}$. The conv-∝SVM algortihm is proposed to solve the relaxed problem. Unlike alter-∝SVM, conv-∝SVM does not require multiple initializations. This method is motivated by the techniques used in large-margin clustering (Li et al., 2009b; Xu et al., 2004).

5.1. Convex Relaxation

We change the label proportion term in the objective function (4) as a constraint $\mathbf{y} \in \mathcal{Y}$, and we drop the bias term b[2]. Then, (4) is rewritten as:

$$\min_{\mathbf{y}\in\mathcal{Y}} \min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^N L(y_i, \mathbf{w}^T\varphi(\mathbf{x}_i)) \tag{9}$$
$$\mathcal{Y} = \left\{\mathbf{y} \middle| |\tilde{p}_k(\mathbf{y}) - p_k| \le \epsilon, y_i \in \{-1,1\}, \forall_{k=1}^K\right\},$$

in which ϵ controls the compatibility of the label proportions. The constraint $\mathbf{y} \in \mathcal{Y}$ can be seen as a special loss function:

$$L_p(\tilde{p}_k(\mathbf{y}), p_k) = \begin{cases} 0, & \text{if } |\tilde{p}_k(\mathbf{y}) - p_k| < \epsilon, \\ \infty, & \text{otherwise.} \end{cases} \tag{10}$$

We then write the inner problem of (9) as its dual:

$$\min_{\mathbf{y}\in\mathcal{Y}} \max_{\boldsymbol{\alpha}\in\mathcal{A}} -\frac{1}{2}\boldsymbol{\alpha}^T\left(\boldsymbol{\mathcal{K}} \odot \mathbf{y}\mathbf{y}^T\right)\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{1}, \tag{11}$$

in which $\boldsymbol{\alpha} \in \mathbb{R}^N$, $\odot$ denotes pointwise-multiplication, $\boldsymbol{\mathcal{K}}$ is the kernel matrix with $\mathcal{K}_{ij} = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$, $\forall_{i,j=1}^N$, and $\mathcal{A} = \{\boldsymbol{\alpha} | 0 \le \boldsymbol{\alpha} \le C\}$.

The objective in (11) is non-convex in $\mathbf{y}$, but convex in $\boldsymbol{M} := \mathbf{y}\mathbf{y}^T$. So, following (Li et al., 2009b; Xu et al., 2004), we instead solve the optimal $\boldsymbol{M}$. However, the feasible space of $\boldsymbol{M}$ is

$$\mathcal{M}_0 = \{\mathbf{y}\mathbf{y}^T | \mathbf{y} \in \mathcal{Y}\}, \tag{12}$$

which is a non-convex set. In order to get a convex optimization problem, we relax $\mathcal{M}_0$ to its convex hull, the tightest convex relaxation of $\mathcal{M}_0$:

$$\mathcal{M} = \left\{ \sum_{\mathbf{y}\in\mathcal{Y}} \mu_{(\mathbf{y})}\mathbf{y}\mathbf{y}^T \middle| \boldsymbol{\mu} \in \mathcal{U}\right\}, \tag{13}$$

in which $\mathcal{U} = \{\boldsymbol{\mu} | \sum_{\mathbf{y}\in\mathcal{Y}} \mu_{(\mathbf{y})} = 1, \mu_{(\mathbf{y})} \ge 0\}$.

[2] If the bias term is not dropped, there will be constraint $\boldsymbol{\alpha}^T\mathbf{y} = 0$ in the dual, leading to non-convexity. Such difficulty has also been discussed in (Xu et al., 2004). Fortunately, the effect of removing the bias term can be alleviated by zero-centering the data or augmenting the feature vector with an additional dimension with value 1.

Thus solving the relaxed $\boldsymbol{M}$ is identical to finding $\boldsymbol{\mu}$:

$$\min_{\boldsymbol{\mu}\in\mathcal{U}}\max_{\boldsymbol{\alpha}\in\mathcal{A}} -\frac{1}{2}\boldsymbol{\alpha}^T\left(\sum_{\mathbf{y}\in\mathcal{Y}}\mu_{(\mathbf{y})}\boldsymbol{\mathcal{K}}\odot\mathbf{y}\mathbf{y}^T\right)\boldsymbol{\alpha}+\boldsymbol{\alpha}^T\mathbf{1}. \quad (14)$$

(14) can be seen as Multiple Kernel Learning (MKL) (Bach et al., 2004), which is a widely studied problem. However, because $|\mathcal{Y}|$ is very large, it is not tractable to solve (14) directly.

5.2. Cutting Plane Training

Fortunately, we can assume that at optimality only a small number of $\mathbf{y}$'s are active in (14). Define $\mathcal{Y}_{active}\subset\mathcal{Y}$ as the set containing all the active $\mathbf{y}$'s. We show that $\mathbf{y}\in\mathcal{Y}_{active}$ can be incrementally found by the cutting plane method.

Because the objective function of (14) is convex in $\boldsymbol{\mu}$, and concave in $\boldsymbol{\alpha}$, it is equivalent to (Fan, 1953),

$$\max_{\boldsymbol{\alpha}\in\mathcal{A}}\min_{\boldsymbol{\mu}\in\mathcal{U}} -\frac{1}{2}\boldsymbol{\alpha}^T\left(\sum_{\mathbf{y}\in\mathcal{Y}}\mu_{(\mathbf{y})}\boldsymbol{\mathcal{K}}\odot\mathbf{y}\mathbf{y}^T\right)\boldsymbol{\alpha}+\boldsymbol{\alpha}^T\mathbf{1}. \quad (15)$$

It is easy to verify that the above is equivalent to:

$$\max_{\boldsymbol{\alpha}\in\mathcal{A},\beta} \quad -\beta \quad (16)$$
$$\text{s.t.} \quad \beta\geq\frac{1}{2}\boldsymbol{\alpha}^T\left(\boldsymbol{\mathcal{K}}\odot\mathbf{y}\mathbf{y}^T\right)\boldsymbol{\alpha}+\boldsymbol{\alpha}^T\mathbf{1},\forall\mathbf{y}\in\mathcal{Y}.$$

This form enables us to apply the cutting plane method (Kelley Jr, 1960) to incrementally include the most violated $\mathbf{y}$ into $\mathcal{Y}_{active}$, and then solve the MKL problem, (14) with $\mathcal{Y}$ replaced as $\mathcal{Y}_{active}$. The above can be repeated until no violated $\mathbf{y}$ exists.

In the cutting plane training, the critical step is to obtain the most violated $\mathbf{y}\in\mathcal{Y}$:

$$\arg\max_{\mathbf{y}\in\mathcal{Y}}\frac{1}{2}\boldsymbol{\alpha}^T\left(\boldsymbol{\mathcal{K}}\odot\mathbf{y}\mathbf{y}^T\right)\boldsymbol{\alpha}+\boldsymbol{\alpha}^T\mathbf{1}, \quad (17)$$

which is equivalent to

$$\arg\max_{\mathbf{y}\in\mathcal{Y}}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j\varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j). \quad (18)$$

This is a 0/1 concave QP, for which there exists no efficient solution. However, instead of finding the most violated constraint, if we find any violated constraint $\mathbf{y}$, the objective function still decreases. We therefore relax the objective in (18), which can be solved efficiently. Note that the objective of (18) is equivalent to a ℓ_2 norm $\sum_{i=1}^{N}\parallel\alpha_i y_i\varphi(\mathbf{x}_i)\parallel_2$. Following (Li et al., 2009b), we approximate it as the ℓ_∞ norm:

$$\sum_{i=1}^{N}\parallel\alpha_i y_i\varphi(\mathbf{x}_i)\parallel_\infty\equiv\max_{j=1\cdots d}\left|\sum_{i=1}^{N}\alpha_i y_i x_i^{(j)}\right|, \quad (19)$$

Algorithm 2 conv-∝SVM

Initialize $\alpha_i=1/N,\forall_{i=1}^N$. $\mathcal{Y}_{active}=\emptyset$. Output: $\boldsymbol{M}\in\mathcal{M}$
repeat
  Compute $\mathbf{y}\in\mathcal{Y}$ based on (18) − (21).
  $\mathcal{Y}_{active}\leftarrow\mathcal{Y}_{active}\cup\{\mathbf{y}\}$.
  Solve the MKL problem in (14) with $\mathcal{Y}_{active}$ to get $\mu_{(\mathbf{y})}$, $\mathbf{y}\in\mathcal{Y}_{active}$.
until The decrease of the objective is smaller than a threshold (10^{-4})

in which $x_i^{(j)}$ is the j-th dimension of the i-th feature vector. These can be obtained by eigendecomposition of the kernel matrix $\boldsymbol{\mathcal{K}}$, when a nonlinear kernel is used. The computational complexity is $\mathcal{O}(dN^2)$. In practice, we choose d such that 90% of the variance is preserved. We further rewrite (19) as:

$$\max_{j=1\cdots d}\max\left(\sum_{i=1}^{N}\alpha_i y_i x_i^{(j)},-\sum_{i=1}^{N}\alpha_i y_i x_i^{(j)}\right) \quad (20)$$
$$=\max_{j=1\cdots d}\max\left(\sum_{k=1}^{K}\sum_{i\in\mathcal{B}_k}\alpha_i y_i x_i^{(j)},\sum_{k=1}^{K}\sum_{i\in\mathcal{B}_k}-\alpha_i y_i x_i^{(j)}\right).$$

Therefore the approximation from (18) to (19) enables us to consider each dimension and each bag separately. For the j-th dimension, and the k-th bag, we only need to solve two sub-problems $\max_{\mathbf{y}\in\mathcal{Y}}\sum_{i\in\mathcal{B}_k}\alpha_i y_i x_i^{(j)}$, and $\max_{\mathbf{y}\in\mathcal{Y}}-\sum_{i\in\mathcal{B}_k}\alpha_i y_i x_i^{(j)}$. The former, as an example, can be written as

$$\min_{\{y_i|i\in\mathcal{B}_k\}}\sum_{i\in\mathcal{B}_k}\left(-\alpha_i x_i^{(j)}\right)y_i,\quad|\tilde{p}_k(\mathbf{y})-p_k|\leq\epsilon. \quad (21)$$

This can be solved in the same way as (8), which takes $\mathcal{O}(|\mathcal{B}_k|\log|\mathcal{B}_k|)$ time. Because we have d dimensions, similar to Proposition 2, one can show that:

Proposition 3 *(18) with the ℓ_2 norm approximated as the ℓ_∞ norm can be solved in $\mathcal{O}(dN\log(J))$ time, $J=\max_{k=1\cdots K}|\mathcal{B}_k|$.*

5.3. The Algorithm

The overall algorithm, called conv-∝SVM, is shown in Algorithm 2. Following (Li et al., 2009b), we use an adapted SimpleMKL algorithm (Rakotomamonjy et al., 2008) to solve the MKL problem.

As an additional step, we need to recover $\mathbf{y}$ from $\boldsymbol{M}$. This is achieved by rank-1 approximation of $\boldsymbol{M}$ (as $\mathbf{y}\mathbf{y}^T$)[3]. Because of the convex relaxation, the comput-

[3]Note that $\mathbf{y}\mathbf{y}^T=(-\mathbf{y})(-\mathbf{y})^T$. This ambiguity can be resolved by validation on the training bags.

ed $\mathbf{y}$ is not binary. However, we can use the real-valued $\mathbf{y}$ directly in our prediction model (with dual):

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x})\right). \quad (22)$$

Similar to alter-∝SVM, the objective of conv-∝SVM is guaranteed to converge. In practice, we terminate the algorithm when the decrease of the objective is smaller than a threshold. Typically the SimpleMKL converges in less than 5 iterations, and conv-∝SVM terminates in less than 10 iterations. The SimpleMKL takes $\mathcal{O}(N^2)$ (computing the gradient) time, or the complexity of SVM, whichever is higher. Recovering $\mathbf{y}$ takes $\mathcal{O}(N^2)$ time and computing eigendecomposition with the first d singular values takes $\mathcal{O}(dN^2)$ time.

6. Experiments

MeanMap (Quadrianto et al., 2009) was shown to outperform alternatives including kernel density estimation, discriminative sorting and MCMC (Kuck & de Freitas, 2005). InvCal (Rüeping, 2010) was shown to outperform MeanMap and several large-margin alternatives. Therefore, in the experiments, we only compare our approach with MeanMap and InvCal.

6.1. A Toy Experiment

To visually demonstrate the advantage of our approach, we first show an experiment on a toy dataset with two bags. Figure 1 (a) and (b) show the data of the two bags, and Figure 1 (c) and (d) show the learned separating hyperplanes from different methods. Linear kernel is used in this experiment. For this specific dataset, the restrictive data assumptions of MeanMap and InvCal do not hold: the mean of the first bag (60% positive) is on the "negative side", whereas, the mean of the second bag (40% positive) is on the "positive side". Consequently, both MeanMap and InvCal completely fail, with the classification accuracy of 0%. On the other hand, our method, which does not make strong data assumptions, achieves the perfect performance with 100% accuracy.

6.2. Experiments on UCI/LibSVM Datasets

Datasets. We compare the performance of different techniques on various datasets from the UCI repository[4] and the LibSVM collection[5]. The details of the datasets are listed in Table 1.

In this paper, we focus on the binary classification settings. For the datasets with multiple classes (dna and satimage), we test the one-vs-rest binary classification performance, by treating data from one class as positive, and randomly selecting same amount of data from the remaining classes as negative. For each dataset, the attributes are scaled to $[-1, 1]$.

Dataset	Size	Attributes	Classes
heart	270	13	2
heart-c	303	13	2
colic	366	22	2
vote	435	16	2
breast-cancer	683	10	2
australian	690	14	2
credit-a	690	15	2
breast-w	699	9	2
a1a	1,605	119	2
dna	2,000	180	3
satimage	4,435	36	6
cod-rna.t	271,617	8	2

Table 1. Datasets used in experiments.

Experimental Setup. Following (Rüeping, 2010), we first randomly split the data into bags of a fixed size. Bag sizes of 2, 4, 8, 16, 32, 64 are tested. We then conduct experiments with 5-fold cross validation. The performance is evaluated based on the average classification accuracy on the individual test instances. We repeat the above processes 5 times (randomly selecting negative examples for the multi-class datasets, and randomly splitting the data into bags), and report the mean accuracies with standard deviations.

The parameters are tuned by an inner cross validation loop on the training subset of each partition of the 5-fold validation. Because no instance-level labels are available during training, we use the bag-level error on the validation bags to tune the parameters:

$$Err = \sum_{k=1}^{T} |\tilde{p}_k - p_k|, \quad (23)$$

in which $\tilde{p}_k$ and p_k denote the predicted and the ground-truth proportions for the k-th validation bag.

For MeanMap, the parameter is tuned from $\lambda \in \{0.1, 1, 10\}$. For InvCal, the parameters are tuned from $C_p \in \{0.1, 1, 10\}$, and $\epsilon \in \{0, 0.01, 0.1\}$. For alter-∝SVM, the parameters are tuned from $C \in \{0.1, 1, 10\}$, and $C_p \in \{1, 10, 100\}$. For conv-∝SVM, the parameters are tuned from $C \in \{0.1, 1, 10\}$, and $\epsilon \in \{0, 0.01, 0.1\}$. Two kinds of kernels are considered: linear and RBF. The parameter of the RBF kernel is tuned from $\gamma = \{0.01, 0.1, 1\}$.

We randomly initialize alter-∝SVM 10 times, and pick the result with the smallest objective value. Empirically, the influence of random initialization to other algorithms is minimal.

[4] http://archive.ics.uci.edu/ml/

[5] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

Dataset	Method	2	4	8	16	32	64
heart	MeanMap	81.85±1.28	80.39±0.47	79.63±0.83	79.46±1.46	79.00±1.42	76.06±1.25
	InvCal	81.78±0.55	80.98±1.35	79.45±3.07	76.94±3.26	73.76±2.69	73.04±6.46
	alter-∝SVM	**83.41±0.71**	**81.80±1.25**	79.91±2.11	79.69±0.64	77.80±2.52	76.58±2.00
	conv-∝SVM	83.33±0.59	80.61±2.48	**81.00±0.75**	**80.72±0.82**	**79.32±1.14**	**79.40±0.72**
colic	MeanMap	80.00±0.80	76.14±1.69	75.52±0.72	74.17±1.61	76.10±1.92	76.74±6.10
	InvCal	81.25±0.24	78.82±3.24	77.34±1.62	74.84±4.14	69.63±4.12	69.47±6.06
	alter-∝SVM	**81.42±0.02**	**80.79±1.48**	**79.59±1.38**	**79.40±1.06**	**78.59±3.32**	**78.49±2.93**
	conv-∝SVM	81.42±0.02	80.63±0.77	78.84±1.32	77.98±1.14	77.49±0.66	76.94±1.07
vote	MeanMap	87.76±0.20	91.90±1.89	90.84±2.33	88.72±1.45	87.63±0.26	88.42±0.80
	InvCal	95.57±0.11	95.57±0.42	94.43±0.24	94.00±0.61	91.47±2.57	91.13±1.07
	alter-∝SVM	**95.62±0.33**	**96.09±0.41**	**95.56±0.47**	**94.23±1.35**	**91.97±1.56**	**92.12±1.20**
	conv-∝SVM	91.66±0.19	90.80±0.34	89.55±0.25	88.87±0.37	88.95±0.39	89.07±0.24
australian	MeanMap	**86.03±0.39**	85.62±0.17	84.08±1.36	83.70±1.45	83.96±1.96	82.90±1.96
	InvCal	85.42±0.28	**85.80±0.37**	84.99±0.68	83.14±2.54	80.28±4.29	80.53±6.18
	alter-∝SVM	85.42±0.30	85.60±0.39	85.49±0.78	84.96±0.96	85.29±0.92	84.47±2.01
	conv-∝SVM	85.51±0.00	85.54±0.08	**85.90±0.54**	**85.67±0.24**	**85.67±0.81**	**85.47±0.89**
dna-1	MeanMap	86.38±1.33	82.71±1.26	79.89±1.55	78.46±0.53	80.20±1.44	78.83±1.73
	InvCal	93.05±1.45	90.81±0.87	86.27±2.43	81.58±3.09	78.31±3.28	72.98±2.33
	alter-∝SVM	**94.93±1.05**	**94.31±0.62**	**92.86±0.78**	**90.72±1.35**	**90.84±0.52**	**89.41±0.97**
	conv-∝SVM	92.78±0.66	90.08±1.18	85.38±2.05	84.91±2.43	82.77±3.30	85.66±0.20
dna-2	MeanMap	88.45±0.68	83.06±1.68	78.69±2.11	79.94±5.68	79.72±3.73	74.73±4.26
	InvCal	93.30±0.88	90.32±1.89	87.30±1.80	83.17±2.18	79.47±2.55	76.85±3.42
	alter-∝SVM	**94.74±0.56**	**94.49±0.46**	**93.06±0.85**	**91.82±1.59**	**90.81±1.55**	**90.08±1.45**
	conv-∝SVM	94.35±1.01	92.08±1.48	89.72±1.26	88.27±1.87	87.58±1.54	86.55±1.18
satimage-2	MeanMap	97.21±0.38	96.27±0.77	95.85±1.12	94.65±0.31	94.49±0.37	94.52±0.28
	InvCal	88.41±3.14	94.65±0.56	94.70±0.20	94.49±0.31	92.90±1.05	93.82±0.60
	alter-∝SVM	**97.83±0.51**	**97.75±0.43**	**97.52±0.48**	**97.52±0.51**	**97.51±0.20**	**97.11±0.26**
	conv-∝SVM	96.87±0.23	96.63±0.09	96.40±0.22	96.87±0.38	96.29±0.40	96.50±0.38

Table 2. Accuracy with linear kernel, with bag size 2, 4, 8, 16, 32, 64.

Method	2^{11}	2^{12}	2^{13}
InvCal	88.79±0.21	88.20±0.62	87.89±0.79
alter-∝SVM	**90.32±1.22**	**90.28±0.94**	**90.21±1.53**

Table 3. Accuracy on cod-rna.t, with linear kernel, with bag size 2^{11}, 2^{12}, 2^{13}.

Results. Table 2 and Table 4 show the results with linear kernel, and RBF kernel, respectively. Additional experimental results are provided in the supplementary material. Our methods consistently outperform MeanMap and InvCal, with p-value < 0.05 for most of the comparisons (more than 70%). For larger bag sizes, the problem of learning from label proportions becomes more challenging due to the limited amount of supervision. For these harder cases, the gains from ∝SVM are typically even more significant. For instance, on the dna-2 dataset, with RBF kernel and bag size 64, alter-∝SVM outperforms the former works by 19.82% and 12.69%, respectively (Table 4).

A Large-Scale Experiment. We also conduct a large-scale experiment on the cod-rna.t dataset containing about 271K points. The performance of InvCal and alter-∝SVM with linear kernel are compared. The experimental setting is the same as for the other datasets. The results in Table 3 show that alter-∝SVM consistently outperforms InvCal. For smaller bag sizes also, alter-∝SVM outperforms InvCal, though the improvement margin reduces due to sufficient amount of supervision.

7. Discussion

7.1. Robustness to $\{p_k\}_{k=1}^K$

In Section 6.2, because the bags were randomly generated, distribution of $\{p_k\}_{k=1}^K$ is approximately Gaussian for moderate to large K. It is intuitive that the performance will depend on the distribution of proportions $\{p_k\}_{k=1}^K$. If p_k is either 0 or 1, the bags are most informative, because this leads to the standard supervised learning setting. On the other hand, if p_k's are close to each other, the bags will be least informative. In fact, both MeanMap and InvCal cannot reach a numerically stable solution in such case. For MeanMap, the linear equations for solving class means will be ill-posed. For InvCal, because all the "super-instances" are assumed to have the same regression value, the result is similar to random guess.

∝SVM, on the other hand, can achieve good performance even in this challenging situation. For example, when using the vote dataset, with bag sizes 8 and 32, $p_k = 38.6\%$, $\forall_{k=1}^K$(same as prior), with linear kernel, alter-∝SVM has accuracies(%) 94.23 ± 1.02 and 86.71 ± 1.30, and conv-∝SVM has accuracies(%) 89.60 ± 0.59 and 87.69 ± 0.51, respectively. These re-

Dataset	Method	2	4	8	16	32	64
heart	MeanMap	82.69±0.71	80.80±0.97	79.65±0.82	79.44±1.21	80.03±2.05	77.26±0.85
	InvCal	**83.15±0.56**	81.06±0.70	80.26±1.32	79.61±3.84	76.36±3.72	73.90±3.00
	alter-∝SVM	83.15±0.85	**82.89±1.30**	**81.51±0.54**	80.07±1.21	79.10±0.96	78.63±1.85
	conv-∝SVM	82.96±0.26	82.20±0.52	81.38±0.53	**81.17±0.55**	**80.94±0.86**	**78.87±1.37**
colic	MeanMap	82.45±0.88	81.38±1.26	81.71±1.16	79.94±1.33	76.36±2.43	77.84±1.69
	InvCal	82.20±0.61	81.20±0.87	81.17±1.74	78.59±2.19	74.09±5.26	72.81±4.80
	alter-∝SVM	**83.28±0.50**	**82.97±0.39**	**82.03±0.44**	**81.62±0.46**	**81.53±0.21**	**81.39±0.34**
	conv-∝SVM	82.74±1.15	81.83±0.46	79.58±0.57	79.77±0.84	78.22±1.19	77.31±1.76
vote	MeanMap	91.15±0.33	90.52±0.62	91.54±0.20	90.28±1.63	89.58±1.09	89.38±1.33
	InvCal	95.68±0.19	94.77±0.44	93.95±0.43	**93.03±0.37**	87.79±1.64	86.63±4.74
	alter-∝SVM	**95.80±0.20**	**95.54±0.25**	**94.88±0.94**	92.44±0.60	**90.72±1.11**	**90.93±1.30**
	conv-∝SVM	92.99±0.20	92.01±0.69	90.57±0.68	88.98±0.35	88.74±0.43	88.62±0.60
australian	MeanMap	85.97±0.72	85.88±0.34	85.34±1.01	83.36±2.04	83.12±1.52	80.58±5.41
	InvCal	**86.06±0.30**	86.11±0.26	**86.32±0.45**	84.13±1.62	82.73±1.70	81.87±3.29
	alter-∝SVM	85.74±0.22	85.71±0.21	86.26±0.61	**85.65±0.43**	83.63±1.83	**83.62±2.21**
	conv-∝SVM	85.97±0.53	**86.46±0.23**	85.30±0.70	84.18±0.53	**83.69±0.78**	82.98±1.32
dna-1	MeanMap	91.53±0.25	90.58±0.34	86.00±1.04	80.77±3.69	77.35±3.59	68.47±4.30
	InvCal	89.32±3.39	92.73±0.53	87.99±1.65	81.05±3.14	74.77±2.95	67.75±3.86
	alter-∝SVM	**95.67±0.40**	**94.65±0.52**	**93.71±0.47**	**92.52±0.63**	**91.85±1.42**	**90.64±1.32**
	conv-∝SVM	93.36±0.53	86.75±2.56	81.03±3.58	75.90±4.56	76.92±5.91	77.94±2.48
dna-2	MeanMap	92.08±1.54	91.03±0.69	87.50±1.58	82.21±3.08	76.77±4.33	72.56±5.32
	InvCal	89.65±4.05	93.12±1.37	89.19±1.17	83.52±2.57	77.94±2.82	72.64±3.89
	alter-∝SVM	**95.63±0.45**	**95.05±0.75**	**94.25±0.50**	**93.95±0.93**	**92.74±0.93**	**92.46±0.90**
	conv-∝SVM	94.06±0.86	90.68±1.18	87.64±0.76	87.32±1.55	85.74±1.03	85.33±0.79
satimage-2	MeanMap	97.08±0.48	96.82±0.38	96.50±0.43	96.45±1.16	95.51±0.73	94.26±0.22
	InvCal	97.53±1.33	98.33±0.13	98.38±0.23	97.99±0.54	96.27±1.15	94.47±0.27
	alter-∝SVM	**98.83±0.36**	**98.69±0.37**	**98.62±0.27**	**98.72±0.37**	**98.51±0.22**	**98.25±0.41**
	conv-∝SVM	96.55±0.13	96.45±0.19	96.45±0.39	96.14±0.49	96.16±0.35	95.93±0.45

Table 4. Accuracy with RBF kernel, with bag size 2, 4, 8, 16, 32, 64.

sults are close to those obtained for randomly generated bags in Table 2. This indicates that our method is less sensitive to the distribution of $\{p_k\}_{k=1}^K$.

7.2. Choice of Algorithm

Empirically, when nonlinear kernel is used, the run time of alter-∝SVM is longer than that of conv-∝SVM, because we are repeating alter-∝SVM multiple times to pick the solution with the smallest objective value. For instance, on a machine with 4-core 2.5GHz CPU, on the vote dataset with RBF kernel and 5-fold cross validation, the alter-∝SVM algorithm (repeating 10 times with the annealing loop, and one set of parameters) takes 15.0 seconds on average, while the conv-∝SVM algorithm takes only 4.3 seconds. But as shown in the experimental results, for many datasets, the performance of conv-∝SVM is marginally worse than that of alter-∝SVM. This can be explained by the multiple relaxations used in conv-∝SVM, and also the 10 time initializations of alter-∝SVM. As a heuristic solution for speeding up the computation, one can use conv-∝SVM (or InvCal) to initialize alter-∝SVM. For large-scale problems, in which linear SVM is used, alter-∝SVM is preferred, because its computational complexity is $\mathcal{O}(N)$.

The speed of both alter-∝SVM and conv-∝SVM can be improved further by solving the SVM in their inner loops incrementally. For example, one can use warm start and partial active-set methods (Shilton et al., 2005). Finally, one can linearize kernels using explicit feature maps (Rahimi & Recht, 2007; Vedaldi & Zisserman, 2012), so that alter-∝SVM has linear complexity even for certain nonlinear kernels.

8. Conclusion and Future Work

We have proposed the ∝SVM framework for learning with label proportions, and introduced two algorithms to efficiently solve the optimization problem. Experiments on several standard and one large-scale dataset show the advantage of the proposed approach over the existing methods. The simple, yet flexible form of ∝SVM framework naturally spans supervised, unsupervised and semi-supervised learning. Due to the usage of latent labels, ∝SVM can also be potentially used in learning with label errors. In the future, we will design algorithms to handle bags with overlapping data. Also, we plan to investigate the theoretical conditions under which the label proportions can be preserved with the convex relaxations.

Acknowledgment. We thank Novi Quadrianto and Yu-Feng Li for their help. We thank Jun Wang, Yadong Mu and anonymous reviewers for the insightful suggestions.

References

Bach, F.R., Lanckriet, G.R.G., and Jordan, M.I. Multiple kernel learning, conic duality, and the smo algorithm. In *Proceedings of the 21th International Conference on Machine learning*, pp. 6, 2004.

Bellare, K., Druck, G., and McCallum, A. Alternating projections for learning with expectation constraints. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pp. 43–50, 2009.

Chapelle, O., Sindhwani, V., and Keerthi, S.S. Optimization techniques for semi-supervised support vector machines. *The Journal of Machine Learning Research*, 9:203–233, 2008.

Chen, B.C., Chen, L., Ramakrishnan, R., and Musicant, D.R. Learning from aggregate views. In *Proceedings of the 22nd International Conference on Data Engineering*, pp. 3, 2006.

Fan, K. Minimax theorems. *Proceedings of the National Academy of Sciences of the United States of America*, 39(1):42, 1953.

Gillenwater, J., Ganchev, K., Graça, J., Pereira, F., and Taskar, B. Posterior sparsity in unsupervised dependency parsing. *The Journal of Machine Learning Research*, 12:455–490, 2011.

Joachims, T. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning*, pp. 200–209, 1999.

Joachims, T. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 217–226, 2006.

Keerthi, S.S., Sundararajan, S., and Shevade, S.K. Extension of TSVM to multi-class and hierarchical text classification problems with general losses. In *Proceeding of the 24th International Conference on Computational Linguistics*, pp. 1091–1100, 2012.

Kelley Jr, J.E. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial & Applied Mathematics*, 8(4):703–712, 1960.

Kuck, H. and de Freitas, N. Learning about individuals from group statistics. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pp. 332–339, 2005.

Li, Y.F., Kwok, J.T., and Zhou, Z.H. Semi-supervised learning using label mean. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 633–640, 2009a.

Li, Y.F., Tsang, I.W., Kwok, J.T., and Zhou, Z.H. Tighter and convex maximum margin clustering. In *Proceeding of the 12th International Conference on Artificial Intelligence and Statistics*, pp. 344–351, 2009b.

Mann, G.S. and McCallum, A. Simple, robust, scalable semi-supervised learning via expectation regularization. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 593–600, 2007.

Musicant, D.R., Christensen, J.M., and Olson, J.F. Supervised learning by training on aggregate outputs. In *Proceedings of the 7th International Conference on Data Mining*, pp. 252–261, 2007.

Quadrianto, N., Smola, A.J., Caetano, T.S., and Le, Q.V. Estimating labels from label proportions. *The Journal of Machine Learning Research*, 10:2349–2374, 2009.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20:1177–1184, 2007.

Rakotomamonjy, A., Bach, F., Canu, S., and Grandvalet, Y. SimpleMKL. *The Journal of Machine Learning Research*, 9:2491–2521, 2008.

Rüeping, S. SVM classifier estimation from group probabilities. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 911–918, 2010.

Shilton, A., Palaniswami, M., Ralph, D., and Tsoi, A.C. Incremental training of support vector machines. *Neural Networks, IEEE Transactions on*, 16 (1):114–131, 2005.

Stolpe, M. and Morik, K. Learning from label proportions by optimizing cluster model selection. In *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases-Volume Part III*, pp. 349–364, 2011.

Vedaldi, A. and Zisserman, A. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3): 480–492, 2012.

Xu, L., Neufeld, J., Larson, B., and Schuurmans, D. Maximum margin clustering. *Advances in Neural Information Processing Systems*, 17:1537–1544, 2004.