Towards Safe Semi-Supervised Learning for Multivariate Performance Measures*

Yu-Feng Li[1,2] **James T. Kwok**[3] **Zhi-Hua Zhou**[1,2]
[1] National Key Laboratory for Novel Software Technology, Nanjing University
[2] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, 210023
[3] Department of Computer Science & Engineering, Hong Kong University of Science and Technology, Hong Kong
liyf@nju.edu.cn jamesk@cse.ust.hk zhouzh@nju.edu.cn

Abstract

Semi-supervised learning (SSL) is an important research problem in machine learning. While it is usually expected that the use of unlabeled data can improve performance, in many cases SSL is outperformed by supervised learning using only labeled data. To this end, the construction of a performance-safe SSL method has become a key issue of SSL study. To alleviate this problem, we propose in this paper the UMVP (safe semi-sUpervised learning for MultiVariate Performance measure) method, because of the need of various performance measures in practical tasks. The proposed method integrates multiple semi-supervised learners, and maximizes the *worst-case* performance gain to derive the final prediction. The overall problem is formulated as a *maximin* optimization. In oder to solve the resultant difficult maximin optimization, this paper shows that when the performance measure is the Top-k Precision, F_β score or AUC, a minimax convex relaxation of the maximin optimization can be solved efficiently. Experimental results show that the proposed method can effectively improve the safeness of SSL under multiple multivariate performance measures.

Introduction

In many real-world tasks, labeled data are expensive to collect and so supervised learning may not be able to attain good performance. In contrast, large amounts of unlabeled data are often readily available. It is now well-known that by using both labeled and unlabeled data, much better performance can be obtained by semi-supervised learning (SSL) (Chapelle, Schölkopf, and Zien 2006; Zhu 2007; Zhou and Li 2010). Over the past decades, SSL has received a lot of attention, and played a significant role in many applications. Examples include image classification, text categorization, bioinformatics, and information retrieval.

In the past, one usually expects SSL to have improved performance whenever labeled data are few. However, recent studies have shown that in many cases, not only is SSL unable to improve performance, it may even be outperformed by supervised learning with the use of labeled data only (Blum and Chawla 2001; Chapelle, Schölkopf, and Zien 2006; Chawla and Karakoulas 2005; Cozman, Cohen, and Cirelo 2002; Li and Zhou 2011a; 2015; Nigam et al. 2000; Zhang and Oles 2000). This phenomenon seriously affects the use of SSL in many applications. The development of safe SSL has thus been considered as an important issue in SSL (Zhou and Li 2010; Li and Zhou 2015). In other words, besides being able to improve performance, SSL should not be worse than simple supervised learning in the worst case.

There are some studies (Cozman, Cohen, and Cirelo 2002; Balcan and Blum 2010; Singh, Nowak, and Zhu 2009; Yang and Priebe 2011; Li and Zhou 2011a; 2015) about the quality of unlabeled data, providing some insights about how to exploit unlabeled data in better ways. However, safe SSL remains challenging. Moreover, in many practical applications, the performance measures are often diverse. For example, in text categorization, the F1-score and precision-recall breakeven point are often used; in information retrieval, precision and recall are more preferred; in ranking applications, the Area Under the ROC Curve (AUC) and Top-k precision are more popular. To this end, one needs to develop a safe SSL method which can work with such a diversity of performance measures.

To alleviate this problem, we propose in this paper the UMVP (safe semi-sUpervised learning for MultiVariate Performance measure) method, which produces the final SSL prediction by integrating multiple semi-supervised learners. Without sufficient knowledge in distinguishing multiple semi-supervised learners, UMVP models the safe SSL problem as a maximin optimization problem. Specifically, the performance gain (with respect to various multivariate performance criteria) is maximized relative to the baseline supervised model in the worst-case scenario. However, the multivariate performance measure is typically discontinuous and non-convex, and thus difficult to optimize. In this paper, a tight minimax convex relaxation technique is adopted, which relaxes the original optimization problem into a convex one with a global solution. We show that when the performance measure is the Top-k Precision, F_β score or AUC, this convex relaxation problem can be solved efficiently by closed-form solutions and small linear programs. Experimental results on a number of data sets show that the proposed method effectively improves the safeness of SSL under multiple multivariate performance measures.

*This research was supported by the NSFC (61333014, 61403186), 863 Program (2015AA015406), JiangsuSF (BK2014 0613) and Hong Kong RGC (614012).

The rest of the paper is organized as follows. We first introduce related works and some commonly used multivariate performance measures. Next, we present the proposed UMVP method. This is then followed by extensive experimental results, and finally we give concluding remarks.

Related Work

Experiments in some literatures (Blum and Chawla 2001; Chapelle, Schölkopf, and Zien 2006; Chawla and Karakoulas 2005; Cozman, Cohen, and Cirelo 2002; Li and Zhou 2011a; 2015; Nigam et al. 2000; Zhang and Oles 2000) have shown that when using unlabeled data, sometimes the performance may be worse than using labeled data only. There are several discussions about the source of the performance degradation. For example, for generative methods, Cozman, Cohen, and Cirelo (2002) conjectured that the performance degradation is caused by the use of incorrect model assumption in the generative methods; however, it is almost impossible to make correct model assumption without adequate domain knowledge. For disagreement-based methods (Zhou and Li 2010), such as co-training (Blum and Mitchell 1998), Li and Zhou (2005) realized that incorrect pseudo-labels used in the SSL process are the source of performance degradation. For graph-based methods, Zhu (2007) conjectured that the graph construction is crucial for the performance, whereas there is no theory to indicate how to construct the correct graph. For semi-supervised SVMs (S3VMs), Li and Zhou (2011b) proved that the existence of multiple low-density separator and the incorrect selection will lead to performance degradation.

Designing safe SSL approaches is very challenging. The first safe SSL approach was developed by (Li and Zhou 2005), which focuses on disagreement-based SSL methods and introduces data editing techniques to filter out potentially incorrect pseudo-labels during the SSL process. This approach works well in many scenarios, however, it is based on heuristics. Li and Zhou (2011b,2015) advocate the importance of safe SSL approaches, and proposed a sound approach to construct safe S3VMs. Their approach, S4VM, tries to optimize the worst-case performance among all potentially selection of low-density separators, and S4VM is proved to be safe given that the ground-truth separator is a low-density separator. In this paper we consider safe SSL under more performance measures besides simply accuracy used in S4VM.

This paper is related to the work in (Joachims 2005) and references therein. However, they typically in the supervised learning setting whereas here we consider SSL. This work also has some connections to semi-supervised ensemble learning approaches (Zhou 2012). However, our primary goal is to make the use of unlabeled data "safe", which has not been considered in previous semi-supervised ensemble learning approaches.

Multivariate Performance Measures

Top-k Precision

Top-k Precision is popularly used in search applications (Joachims 2005), in which one wants to predict whether an instance is relevant. On a given set of u instances, let the ground-truth relevance be stored in $\mathbf{y} = [y_1, \dots, y_u] \in \{0,1\}^u$ (where 1 indicates relevant and 0 otherwise), and the predictions be $\hat{\mathbf{y}} = [\hat{y}_1, \dots, \hat{y}_u] \in \mathbb{R}^u$. Rankings of the instances by $\hat{y}_i$ can be represented by the vector $\pi^{\hat{\mathbf{y}}} \in \{1, \dots, u\}^u$, where $\pi_i^{\hat{\mathbf{y}}} > \pi_j^{\hat{\mathbf{y}}}$ if $\hat{y}_i > \hat{y}_j$ (with ties broken arbitrarily).

Typically, users are only interested in the top-ranked instances. The *top-k precision* is defined as (Manning, Raghavan, and Schtze 2008)

$$Pre@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{k} \sum_{p:1 \le p \le u} \mathbb{I}(y_p = 1)\mathbb{I}(\pi_p^{\hat{\mathbf{y}}} > u - k), \quad (1)$$

where $\mathbb{I}(\cdot)$ is the indicator function that returns 1 when the argument holds, and 0 otherwise.

F_β Score

The F_β score is commonly used for binary classification, particularly when classes are highly imbalanced. Let $\mathbf{y} \in \{0,1\}^u$ be the vector of ground-truth labels (where 1 is for positive instances and 0 for negative ones), and $\hat{\mathbf{y}} \in \{0,1\}^u$ be the predictions. Precision and recall are defined as $Pre(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{\mathbf{y}}'\mathbf{y}}{\hat{\mathbf{y}}'\mathbf{1}}$ and $Rec(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{\mathbf{y}}'\mathbf{y}}{\mathbf{y}'\mathbf{1}}$, respectively, where $\mathbf{1}$ is the vector of all ones, and $'$ denotes the transpose of a vector. The F_β score is a weighted harmonic average of precision and recall (Manning, Raghavan, and Schtze 2008)

$$\begin{aligned} F_\beta(\hat{\mathbf{y}}, \mathbf{y}) &= (1+\beta^2)\frac{Pre(\hat{\mathbf{y}}, \mathbf{y})Rec(\hat{\mathbf{y}}, \mathbf{y})}{\beta^2 Pre(\hat{\mathbf{y}}, \mathbf{y}) + Rec(\hat{\mathbf{y}}, \mathbf{y})} \\ &= \frac{(1+\beta^2)\hat{\mathbf{y}}'\mathbf{y}}{\hat{\mathbf{y}}'\mathbf{1} + \beta^2 \mathbf{y}'\mathbf{1}}, \quad (2) \end{aligned}$$

where β (usually set to 1) trades off precision and recall.

AUC (Area Under the ROC Curve)

The AUC (Manning, Raghavan, and Schtze 2008) characterizes the probability that positive instances are ranked higher than negative ones. Let the ground-truth labels on the unlabeled data be $\mathbf{y} = [y_1, \dots, y_u] \in \{0,1\}^u$, and the predictions be $\hat{\mathbf{y}} = [\hat{y}_1, \dots, \hat{y}_u] \in \mathbb{R}^u$. The AUC is defined as

$$AUC(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\sum_{p,q:1 \le p,q \le u} \mathbb{I}(y_p > y_q)\mathbb{I}(\hat{y}_p > \hat{y}_q)}{P_{\mathbf{y}} N_{\mathbf{y}}}, \quad (3)$$

where $P_{\mathbf{y}}$ and $N_{\mathbf{y}}$ are the numbers of positive and negative labels in $\mathbf{y}$, respectively.

The UMVP Method

The UMVP method uses b semi-supervised learners to construct safe predictions. These learners may be obtained by, e.g., (i) running a SSL algorithm with different parameters, (ii) different SSL algorithms, or (iii) a hybrid of the two.

Let *perf* be the target performance measure (e.g., top-k precision, F$_\beta$, AUC). Without loss of generality, we assume that the larger the *perf* value, the better the performance. Our goal is to find a prediction $\hat{\mathbf{y}}$ that maximally aligns with predictions from the b semi-supervised learners, and also performs better than a given baseline learner (e.g., a supervised learner) $\hat{\mathbf{y}}^0$. This can be formulated as the following optimization problem

$$\max_{\hat{\mathbf{y}}\in\mathcal{Y}} \sum_{i=1}^{b} \alpha_i \big(\mathit{perf}(\hat{\mathbf{y}},\mathbf{y}^i) - \mathit{perf}(\hat{\mathbf{y}}^0,\mathbf{y}^i)\big), \qquad (4)$$

where $\{\mathbf{y}^1,\dots,\mathbf{y}^b\}$ are predictions of the b semi-supervised learners on the unlabeled instances, $\mathbf{y}^i = [y_1^i,\dots,y_u^i] \in \{0,1\}^u$, $\mathcal{Y}$ is the feasible region[1] of $\hat{\mathbf{y}}$; $\big(\mathit{perf}(\hat{\mathbf{y}},\mathbf{y}^i) - \mathit{perf}(\hat{\mathbf{y}}^0,\mathbf{y}^i)\big)$ is the performance improvement of $\hat{\mathbf{y}}$ relative to $\mathbf{y}^0$, if $\mathbf{y}^i$ is the ground-truth label assignment, $\boldsymbol{\alpha} = [\alpha_1,\dots,\alpha_b]$ captures the relative importance of the b semi-supervised learners. Without loss of generality, we assume that $\boldsymbol{\alpha}$ is in the simplex $\mathcal{M} = \{\boldsymbol{\alpha} \mid \sum_{i=1}^{b}\alpha_i = 1, \alpha_i \geq 0\}$. We also assume that $\hat{\mathbf{y}}_0 \in \mathcal{Y}$, and so the objective of Eq.(4) is non-negative.

In real-world situations, the relative importance of semi-supervised learners are typically unknown. Simply trusting any one of these may risk performance degradation. To address this problem, we consider the *worst-case*, adversarial setting of $\boldsymbol{\alpha}$. The goal is to achieve the best possible performance gain relative to the baseline $\hat{\mathbf{y}}_0$ even in this scenario. The UMVP method formulates this as the following maximin optimization problem:

$$\max_{\hat{\mathbf{y}}\in\mathcal{Y}} \min_{\boldsymbol{\alpha}\in\mathcal{M}} \sum_{i=1}^{b} \alpha_i \big(\mathit{perf}(\hat{\mathbf{y}},\mathbf{y}^i) - \mathit{perf}(\hat{\mathbf{y}}_0,\mathbf{y}^i)\big). \qquad (5)$$

It is easy to see from Eq.(5) that when one of the semi-supervised leaners realizes the ground-truth label assignment, the UMVP solution obtained is safe. Recall that the success of SSL often relies on some assumptions (e.g., cluster assumption, low-density assumption, manifold assumption) (Chapelle, Schölkopf, and Zien 2006). It remains challenging to see which one is more suitable for a particular data set. Eq.(5) shows that as long as one of these assumptions realizes a perfect solution, a safe SSL method can be obtained. Moreover, note that this is a sufficient condition, but not necessary condition for the safeness of UMVP.

Optimizing Eq.(5)

As many performance measures are discontinuous and non-convex, continuous optimization techniques (such as gradient-based approaches) are not applicable. In this paper, we propose the use of convex relaxation and cutting plane algorithm (Kelley 1960), which have been very useful in solving nonsmooth problems (Sra, Nowozin, and Wright 2012).

Exchanging the $\max_{\hat{\mathbf{y}}\in\mathcal{Y}}$ and $\min_{\boldsymbol{\alpha}\in\mathcal{M}}$ operators in Eq.(5), we have

$$\min_{\boldsymbol{\alpha}\in\mathcal{M}} \max_{\hat{\mathbf{y}}\in\mathcal{Y}} \sum_{i=1}^{b} \alpha_i \big(\mathit{perf}(\hat{\mathbf{y}},\mathbf{y}^i) - \mathit{perf}(\hat{\mathbf{y}}_0,\mathbf{y}^i)\big). \qquad (6)$$

From the minimax theorem (Kim and Boyd 2008), the objective of Eq.(6) upper-bounds that of Eq.(5). Eq.(6) can also be rewritten as

$$\min_{\boldsymbol{\alpha}\in\mathcal{M},\theta} \quad \theta \qquad (7)$$

$$\text{s.t.} \quad \theta \geq \sum_{i=1}^{b} \alpha_i \big(\mathit{perf}(\hat{\mathbf{y}},\mathbf{y}^i) - \mathit{perf}(\hat{\mathbf{y}}_0,\mathbf{y}^i)\big), \forall \hat{\mathbf{y}} \in \mathcal{Y}.$$

Both the objective and constraints in problem (7) are linear in $\boldsymbol{\alpha}$ and θ. Hence, problem (7) is convex. In other words, problem (7), or equivalently problem (6), is a convex relaxation of (5). Such convex relaxations have been shown to be tight for a number of non-convex problems (Li et al. 2013).

Because of the large number of possible label assignments/ranks for the unlabeled instances, problem (7) involves an exponential number of constraints. Usually, not all of them are active, and using a small subset of these can lead to a good approximate solution. This motivates the use of the cutting-plane algorithm, which iteratively adds a cutting plane to shrink the feasible region. Since the constraints in problem (7) are linear in $\boldsymbol{\alpha}$, the cutting plane corresponds to finding the most violated constraint for the current $\boldsymbol{\alpha}$ (Nesterov 2003), i.e., solving the following optimization problem

$$\arg\max_{\hat{\mathbf{y}}\in\mathcal{Y}} \sum_{i=1}^{b} \alpha_i \big(\mathit{perf}(\hat{\mathbf{y}},\mathbf{y}^i) - \mathit{perf}(\hat{\mathbf{y}}_0,\mathbf{y}^i)\big). \qquad (8)$$

While this step can only be suboptimally solved in some machine learning problems (Tsochantaridis et al. 2005; Li et al. 2013), here, we show when the performance measure is either the Top-k Precision, F$_\beta$ score or AUC, Eq.(8) has a closed-form optimal solution.

Closed-Form Solutions of Eq.(8) When $\boldsymbol{\alpha}$ is known, Eq.(8) reduces to the simpler problem:

$$\max_{\hat{\mathbf{y}}\in\mathcal{Y}} \sum_{i=1}^{b} \alpha_i \, \mathit{perf}(\hat{\mathbf{y}},\mathbf{y}^i). \qquad (9)$$

In the following, we show that closed-form solution exists when either the Top-k Precision, F$_\beta$ score or AUC is used as performance measure.

Proposition 1. *When the Top-k precision is used in Eq.(9), any $\hat{\mathbf{y}}$ that ranks the unlabeled instances as $\mathbf{s} = \sum_{i=1}^{b}\alpha_i\mathbf{y}^i$ is optimal (ties are broken arbitrarily).*

Proof. From Eq.(1), the optimal $\hat{\mathbf{y}}^*$ of Eq.(9) only relates to its ranking vector $\pi^{\hat{\mathbf{y}}^*}$. The key is to show $\pi^{\hat{\mathbf{y}}^*} = \pi^{\mathbf{s}}$. Assume, to the contrary, that $\pi^{\hat{\mathbf{y}}^*} \neq \pi^{\mathbf{s}}$. Then there exist two unlabeled instances $\mathbf{x}_{l+p}$, $\mathbf{x}_{l+q}$ such that $\pi_p^{\hat{\mathbf{y}}^*} > \pi_q^{\hat{\mathbf{y}}^*}$ but $s_p < s_q$. Define a new $\mathbf{z}$ such that $\pi^{\mathbf{z}} = [\pi_j^{\mathbf{z}}]$ with

$$\pi_j^{\mathbf{z}} = \begin{cases} \pi_j^{\mathbf{y}^*} & \text{if } j \neq p \text{ and } j \neq q \\ \pi_p^{\hat{\mathbf{y}}^*} & \text{if } j = q \\ \pi_q^{\hat{\mathbf{y}}^*} & \text{if } j = p \end{cases} .$$

From Eq.(1),

$$\begin{aligned} &\sum\nolimits_{i=1}^{b} \alpha_i Pre@k(\hat{\mathbf{y}}^*,\mathbf{y}^i) - \sum\nolimits_{i=1}^{b} \alpha_i Pre@k(\mathbf{z},\mathbf{y}^i) \\ = \;& \frac{1}{k}\big(\sum\nolimits_{i=1}^{b}\alpha_i y_p^i - \sum\nolimits_{i=1}^{b}\alpha_i y_q^i\big) < 0, \end{aligned}$$

[1] If the target $\hat{\mathbf{y}}$ are values output by the learner, $\mathcal{Y} = \mathbb{R}^u$. If $\hat{\mathbf{y}}$ corresponds to the binary class labels, $\mathcal{Y} = \{0,1\}^u$.

which contradicts the optimality of $\hat{\mathbf{y}}^*$. Thus $\pi^{\hat{\mathbf{y}}^*} = \pi^{\mathbf{s}}$. □

Proposition 2. *Assume that $\hat{\mathbf{y}}'\mathbf{1} = c$, a constant. Let $\mathbf{s} = \sum_{i=1}^{b} \frac{\alpha_i(1+\beta^2)}{c+\beta^2 P_{\mathbf{y}^i}}\mathbf{y}^i$, where $P_{\mathbf{y}^i} = \mathbf{y}^{i\prime}\mathbf{1}$ is the number of positive labels in $\mathbf{y}^i$. When the F_β score is used in Eq.(9), the optimal $\hat{\mathbf{y}}^* = [\hat{y}_j^*]$ is given by*

$$\hat{y}_j^* = \begin{cases} 1 & \pi_j^{\mathbf{s}} > u - c \\ 0 & \textit{otherwise} \end{cases}.$$

Proof. Expand Eq.(9) with Eq.(2), we have

$$\sum_{i=1}^{b} \alpha_i \frac{(1+\beta^2)\hat{\mathbf{y}}'\mathbf{y}^i}{\hat{\mathbf{y}}'\mathbf{1}+\beta^2\mathbf{y}^{i\prime}\mathbf{1}} = \sum_{i=1}^{b} \frac{\alpha_i(1+\beta^2)\hat{\mathbf{y}}'\mathbf{y}^i}{c+\beta^2 P_{\mathbf{y}^i}} = \hat{\mathbf{y}}'\mathbf{s}.$$

Eq.(9) is equivalent to

$$\max_{\hat{\mathbf{y}}} \hat{\mathbf{y}}'\mathbf{s} \ \text{ s.t. } \ \hat{\mathbf{y}}'\mathbf{1} = c, \hat{\mathbf{y}} \in \{1, 0\}^u.$$

Obviously, selecting the largest c elements of $\mathbf{s}$ and setting the corresponding $\hat{y}_i$ entries to one is optimal. □

Before showing the closed-form solution of Eq.(9) for AUC, we first introduce the following Lemma.

Lemma 1. *Eq.(3) can be rewritten as*

$$AUC(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\sum_{p:1\le p\le u} \mathbb{I}(y_p = 1)\pi_p^{\hat{\mathbf{y}}} - \frac{1}{2}(P_{\mathbf{y}}+1)P_{\mathbf{y}}}{P_{\mathbf{y}}N_{\mathbf{y}}}. \tag{10}$$

Proof. Note that the AUC is proportional to the number of positive-negative instance pairs such that the positive ones are ranked higher than the negative ones. Denote these as the correct pairs. The AUC can be rewritten as

$$AUC(\hat{\mathbf{y}}, \mathbf{y}) = \text{number of correct pairs}/P_{\mathbf{y}}N_{\mathbf{y}}.$$

Let $[\pi_1^{\hat{\mathbf{y}}}, \ldots, \pi_{P_{\mathbf{y}}}^{\hat{\mathbf{y}}}]$ be the ranks of positive instances, with $\pi_1^{\hat{\mathbf{y}}} > \pi_2^{\hat{\mathbf{y}}} > \cdots > \pi_{P_{\mathbf{y}}}^{\hat{\mathbf{y}}}$. The total number of correct pairs is equal to $\sum_{i=1}^{P_{\mathbf{y}}} \big(\pi_i^{\hat{\mathbf{y}}} - (P_{\mathbf{y}} - i + 1)\big) = \sum_{i=1}^{P_{\mathbf{y}}} \pi_i^{\hat{\mathbf{y}}} - \sum_{i=1}^{P_{\mathbf{y}}} (P_{\mathbf{y}} - i + 1) = \sum_{1\le p\le u} \mathbb{I}(y_p = 1)\pi_p^{\hat{\mathbf{y}}} - \frac{1}{2}(P_{\mathbf{y}}+1)P_{\mathbf{y}}$, and hence Eq.(10). □

Proposition 3. *When the AUC is used in Eq.(9), any $\hat{\mathbf{y}}$ that ranks the unlabeled instances as $\mathbf{s} = \sum_{i=1}^{b} \frac{\alpha_i}{P_{\mathbf{y}^i}N_{\mathbf{y}^i}}\mathbf{y}^i$ is optimal.*

Proof. From (3), the optimal $\hat{\mathbf{y}}^*$ of Eq.(9) only relates to its ranking vector $\pi^{\hat{\mathbf{y}}^*}$. From Lemma 1, on substituting Eq.(10) into Eq.(9), we have

$$\sum_{i=1}^{b} \alpha_i AUC(\hat{\mathbf{y}}, \mathbf{y}^i) = \sum_{i=1}^{b} \alpha_i \frac{\sum_{1\le p\le u} \mathbb{I}(y_p^i = 1)\pi_p^{\hat{\mathbf{y}}}}{P_{\mathbf{y}^i}N_{\mathbf{y}^i}} - h,$$

where $h = \sum_{i=1}^{b} \alpha_i \frac{(P_{\mathbf{y}^i}+1)P_{\mathbf{y}^i}}{2P_{\mathbf{y}^i}N_{\mathbf{y}^i}}$ is a constant (not related to $\hat{\mathbf{y}}$). Now,

$$\sum_{i=1}^{b} \alpha_i \frac{\sum\limits_{1\le p\le u} \mathbb{I}(y_p^i = 1)\pi_p^{\hat{\mathbf{y}}}}{P_{\mathbf{y}^i}N_{\mathbf{y}^i}} = \sum_{i=1}^{b} \alpha_i \frac{\sum\limits_{1\le p\le u} y_p^i \pi_p^{\hat{\mathbf{y}}}}{P_{\mathbf{y}^i}N_{\mathbf{y}^i}} = \pi^{\hat{\mathbf{y}}\prime}\mathbf{s}.$$

Hence, maximizing $\pi^{\hat{\mathbf{y}}\prime}\mathbf{s}$ leads to $\pi^{\hat{\mathbf{y}}}$ having the same ranking of elements in $\mathbf{s}$. □

Algorithm 1 Cutting-plane algorithm for Eq.(7).

```
Input: $\{\mathbf{y}^i\}_{i=1}^{b}$.
1: Initialize $\hat{\mathbf{y}}^0 = \sum_{i=1}^{b} \frac{1}{b}\mathbf{y}^i$ and working set $\mathcal{C} = \{\hat{\mathbf{y}}^0\}$.
2: Solve the subproblem
     $\min_{\boldsymbol{\alpha}\in\mathcal{M},\theta} \ \theta$     (11)
     s.t. $\theta \ge \sum_{i=1}^{b} \alpha_i\big(perf(\hat{\mathbf{y}}, \mathbf{y}^i) - perf(\hat{\mathbf{y}}_0, \mathbf{y}^i)\big), \forall\hat{\mathbf{y}} \in \mathcal{C}$
   as a linear program, and obtain the solution $\boldsymbol{\alpha}$. Denote
   the obtained objective value as $\hat{o}$;
3: Solve Eq.(8) via the closed-form solutions, and obtain
   the solution $\hat{\mathbf{y}}_{new}$. Denote the obtained objective value
   as $o$.
4: Update $\mathcal{C} = \mathcal{C} \bigcup \{\hat{\mathbf{y}}_{new}\}$;
5: Repeat steps 2-4 until $o - \hat{o} \le \epsilon$;
6: Output $\hat{\mathbf{y}} = \hat{\mathbf{y}}_{new}$.
```

Algorithm 1 shows the optimization procedure. It iteratively solves $\boldsymbol{\alpha}$ via a small set of constraints (i.e., Eq.(11)) and finds a violated constraint according to the updated $\boldsymbol{\alpha}$ (i.e, Eq.(8)). Solving Eq.(11) leads to a small linear program and solving Eq.(8) has closed-form solution. These allow the proposed algorithm to be very efficient. After obtaining the optimal $\boldsymbol{\alpha}$, the optimal solution of Eq.(8) is employed as an approximate solution of Eq.(5).

Experiments

We evaluate the proposed UMVP method on a number of binary classification data sets[2] that cover a wide range of properties (Table 1). The sample size ranges from 1,500 to more than 70,000. The feature dimensionality ranges from 19 to more than 20,000. The proportion of classes (i.e., ratio of the number of positive samples to that of negative samples) ranges from 0.03 to around 1. For each data set, 1% of the samples are labeled and the rest are unlabeled. The same class imbalance ratio is maintained on both sets. Each experiment is repeated 10 times, and the average performance on the unlabeled data is reported.

The semi-supervised learners used in UMVP are the semi-supervised SVMs (S3VM's) (Joachims 1999). By using 20 random initial label assignments on the unlabeled data, 20 S3VM's are obtained. Predictions on the unlabeled data from these 20 S3VMs are then grouped into 5 clusters by using the k-means clustering algorithm. The S3VM with the smallest S3VM objective in each cluster is selected as the semi-supervised learner of UMVP. The linear program in Eq.(11) is solved by the Matlab function *linprog*, and ϵ in Algorithm 1 is set to 10^{-6}. When F$_\beta$ is used as the performance measure, the c in Proposition 2 is set to the average

[2] Downloaded from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`, `http://www.kyb.tuebingen.mpg.de/ssl-book/`, and from (Mallapragada et al. 2009) (for the ethn data set). For the MNIST data set, we focus on its four most difficult binary classification tasks (Zhang, Tsang, and Kwok 2007).

Table 1: Results on Pre@k, F1 and AUC (all in percentages). "*Imbalance rate*" is the ratio of the numbers of positive instances to that of negative instances. △ (resp., ♠) denotes that the result is significantly worse (resp., better) than that of SVM^{perf} (the paired t-tests at 95% significance level).

		Pre@k	F1	AUC
COIL2	SVM^{perf}	58.5 ± 3.9	62.2 ± 5.8	61.4 ± 4.7
# samples: 1,500	self-SVM^{perf}	57.3 ± 3.8	57.7 ± 4.9	58.0 ± 4.6 △
# features: 241	S4VM	60.6 ± 4.8	61.1 ± 4.5	62.1 ± 4.1
Imbalance rate: 1.0	UMVP$^-$	58.4 ± 4.7	59.8 ± 5.4	59.3 ± 3.4 △
1% labeled instances	UMVP	58.7 ± 4.4	60.0 ± 5.6	62.2 ± 4.6
digit1	SVM^{perf}	83.2 ± 3.8	79.5 ± 3.6	92.0 ± 2.8
# samples: 1,500	self-SVM^{perf}	84.5 ± 3.3 ♠	85.5 ± 3.5 ♠	86.8 ± 3.7 △
# features: 241	S4VM	83.4 ± 3.3	83.2 ± 3.7 ♠	89.6 ± 2.8 △
Imbalance rate: 0.96	UMVP$^-$	87.0 ± 3.0 ♠	86.6 ± 3.4 ♠	94.7 ± 2.5 ♠
1% labeled instances	UMVP	86.7 ± 3.0 ♠	86.4 ± 3.3 ♠	94.0 ± 2.4 ♠
ethn	SVM^{perf}	80.8 ± 4.0	78.0 ± 2.4	88.8 ± 3.3
# samples: 2,630	self-SVM^{perf}	86.4 ± 3.9 ♠	84.5 ± 5.7 ♠	81.5 ± 4.4 △
# features: 30	S4VM	82.3 ± 5.1	81.7 ± 5.4 ♠	79.7 ± 5.9 △
Imbalance rate: 0.99	UMVP$^-$	83.2 ± 4.9 ♠	83.4 ± 5.0 ♠	91.7 ± 3.7 ♠
1% labeled instances	UMVP	84.5 ± 5.4 ♠	82.2 ± 5.5 ♠	91.7 ± 3.0 ♠
mnist4vs9	SVM^{perf}	91.4 ± 0.9	92.2 ± 0.9	97.2 ± 0.6
# samples: 13,782	self-SVM^{perf}	86.6 ± 1.8 △	93.8 ± 0.8 ♠	96.4 ± 1.6
# features: 629	S4VM	93.3 ± 0.6 ♠	93.5 ± 0.9 ♠	98.8 ± 0.5 ♠
Imbalance rate: 0.98	UMVP$^-$	94.1 ± 0.5 ♠	94.3 ± 1.4 ♠	98.5 ± 0.3 ♠
1% labeled instances	UMVP	94.0 ± 0.5 ♠	94.1 ± 1.3 ♠	97.9 ± 0.4 ♠
mnist3vs8	SVM^{perf}	92.8 ± 1.1	93.3 ± 0.8	97.8 ± 0.6
# samples: 13,966	self-SVM^{perf}	90.3 ± 1.4 △	94.3 ± 1.2 ♠	97.7 ± 0.7
# features: 631	S4VM	93.8 ± 0.9 ♠	94.3 ± 0.9 ♠	99.1 ± 0.2 ♠
Imbalance rate: 1.05	UMVP$^-$	94.4 ± 0.8 ♠	95.0 ± 0.8 ♠	98.3 ± 0.4 ♠
1% labeled instances	UMVP	94.3 ± 0.8 ♠	95.0 ± 0.8 ♠	98.1 ± 0.6 ♠
mnist7vs9	SVM^{perf}	91.7 ± 1.2	92.2 ± 1.0	97.0 ± 0.7
# samples: 14,251	self-SVM^{perf}	88.3 ± 1.4 △	94.2 ± 0.7 ♠	96.0 ± 1.2 △
# features: 600	S4VM	93.1 ± 1.1 ♠	93.6 ± 1.0 ♠	98.8 ± 0.4 ♠
Imbalance rate: 1.05	UMVP$^-$	93.6 ± 0.7 ♠	94.0 ± 0.7 ♠	97.7 ± 0.4 ♠
1% labeled instances	UMVP	93.4 ± 0.8 ♠	93.8 ± 0.8 ♠	97.5 ± 0.5 ♠
mnist1vs7	SVM^{perf}	98.0 ± 0.2	98.1 ± 0.4	99.9 ± 0.0
# samples: 15,170	self-SVM^{perf}	93.8 ± 1.6 △	98.8 ± 0.4 ♠	97.5 ± 1.0 △
# features: 652	S4VM	98.4 ± 0.1 ♠	99.0 ± 0.1 ♠	99.9 ± 0.1 ♠
Imbalance rate: 1.08	UMVP$^-$	98.6 ± 0.1 ♠	99.1 ± 0.1 ♠	99.9 ± 0.0 ♠
1% labeled instances	UMVP	98.6 ± 0.1 ♠	99.1 ± 0.1 ♠	99.9 ± 0.0 ♠
adult-a	SVM^{perf}	62.2 ± 1.2	63.1 ± 1.1	86.8 ± 0.8
# samples: 32,561	self-SVM^{perf}	59.4 ± 1.6 △	58.4 ± 1.6 △	82.0 ± 2.8 △
# features: 123	S4VM	59.9 ± 2.0 △	59.9 ± 2.1 △	76.4 ± 1.2 △
Imbalance rate: 0.32	UMVP$^-$	61.2 ± 1.9 △	62.1 ± 1.0 △	86.5 ± 1.0 △
1% labeled instances	UMVP	61.8 ± 1.4	62.2 ± 1.3 △	86.8 ± 0.8
w8a	SVM^{perf}	36.1 ± 4.1	24.8 ± 3.0	82.7 ± 3.0
# samples: 49,749	self-SVM^{perf}	32.2 ± 5.6	31.9 ± 5.7 ♠	97.8 ± 0.2 ♠
# features: 300	S4VM	35.9 ± 2.7	35.6 ± 1.9 ♠	92.9 ± 1.3 ♠
Imbalance rate: 0.03	UMVP$^-$	37.2 ± 4.4	34.5 ± 6.1 ♠	79.1 ± 2.7 △
1% labeled instances	UMVP	37.5 ± 2.9	35.1 ± 5.8 ♠	82.9 ± 2.9 ♠
real-sim	SVM^{perf}	85.3 ± 0.6	54.6 ± 0.4	97.3 ± 0.2
# samples: 72,309	self-SVM^{perf}	76.5 ± 0.2 △	76.0 ± 0.2 ♠	98.1 ± 0.1 ♠
# features: 20,958	S4VM	89.1 ± 0.5 ♠	89.2 ± 0.6 ♠	98.8 ± 0.1 ♠
Imbalance rate: 0.44	UMVP$^-$	88.9 ± 0.8 ♠	89.3 ± 0.8 ♠	98.0 ± 0.2 ♠
1% labeled instances	UMVP	88.7 ± 1.2 ♠	89.5 ± 1.1 ♠	97.7 ± 0.2 ♠

number of positive samples for the multiple learners., i.e., $\frac{1}{b}\sum_{i=1}^{b}(\mathbf{y}^i)'\mathbf{1}$.

UMVP is compared with the following methods.

1. SVM^{perf}(Joachims 2005)[3]: This is our baseline supervised model, which uses the SVM to optimize multivariate performance measures with only labeled data.

2. Self-SVM^{perf}: This is a semi-supervised extension of SVM^{perf} based on self-training (Yarowsky 1995). It first trains a standard SVM^{perf} (with only labeled data). By adding the predicted labels on the unlabeled data as "ground-truth", another SVM^{perf} is trained. This process is repeated until predictions on the unlabeled data no longer change or a maximum number of iterations (set to 25 in the experiments) is reached.

Table 2: A summary of the experimental results. For each performance measure, the "average performance improvement" is the average of (performance value of the semi-supervised learner − performance value of SVM^{perf}) over all data sets. "Win/tie/loss" counts the data sets for which the semi-supervised learner is statistically significantly better/comparable/significantly worse than SVM^{perf}. "(H, p-value)" of the Wilcoxon sign test is calculated from Win/Tie/Loss. H= $-1/0/1$ denotes that the semi-supervised learner is statistically significantly worse/comparable/significantly better than SVM^{perf}, and the corresponding p-value.

		Pre@k	F1	AUC
average performance improvement	Self-SVM^{perf}	-2.5	3.7	-0.9
	S4VM	1.0	5.3	-0.5
	UMVP$^-$	1.7	6.0	0.3
	UMVP	1.8	5.9	0.8
win/tie/loss	Self-SVM^{perf}	2/2/6	8/1/1	2/2/6
	S4VM	4/5/1	8/1/1	6/1/3
	UMVP$^-$	6/3/1	8/1/1	7/0/3
	UMVP	6/4/0	8/1/1	8/2/0
sign test (H, p-value)	Self-SVM^{perf}	(0, 0.29)	(1, 0.04)	(0, 0.29)
	S4VM	(0, 0.38)	(1, 0.04)	(0, 0.51)
	UMVP$^-$	(0, 0.13)	(1, 0.04)	(0, 0.34)
	UMVP	(1, 0.03)	(1, 0.04)	(1, 0.01)

3. S4VM (Safe Semi-Supervised SVM) (Li and Zhou 2015)[4]: S4VM performs quite well in many scenarios when using accuracy as performance measurement.

4. UMVP$^-$, a variant of UMVP which assigns uniform weights to the semi-supervised leaners.

For all methods, the C parameter in SVM is set to 1 and the linear kernel is used. Parameters of S4VM are set as recommended in the package. Performance evaluation is based on Pre@k (with k being the number of positive instances in the test set), F1 and AUC. The experiments are used with MATLAB 8.0.1 and LIBLINEAR 1.91. Experiments are run on a PC with a 3.2GHz Core2 Duo CPU and 4GB memory.

We compare the safeness of SSL methods with the baseline supervised model by the following:

1. Average performance improvement.

2. Win/Tie/Loss: This characterizes the degree of performance degradation of SSL methods.

3. Sign test: This characterizes the dependence between the performance of SSL methods and data sets.

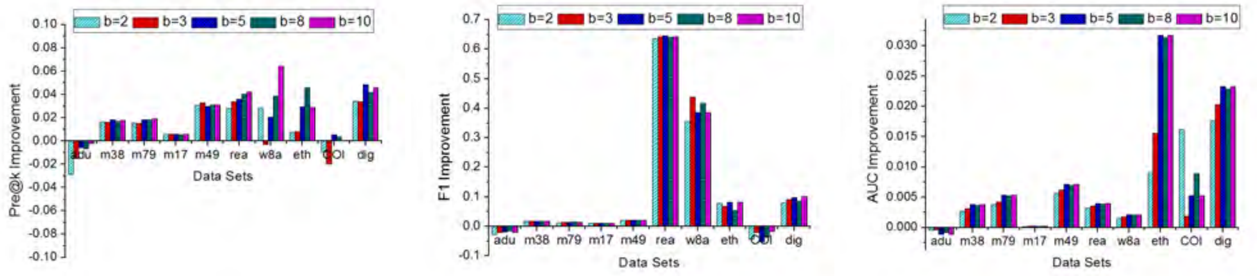
Table 1 shows the experimental results of the various methods on all data sets. Table 2 summarizes the results.

[3] http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html

[4] http://lamda.nju.edu.cn/code_S4VM.ashx

Table 3: Average CPU time (in seconds) with different performance measures for the various methods. For UMVP, the first number inside brackets is the time for generating the SSL learners, and the second number is the time for optimization.

	SVM^{perf}	Self-SVM^{perf}	S4VM	UMVP
adult-a	0.844	145.516	22.403	34.811 (32.936 + 1.875)
mnist3vs8	3.622	621.665	148.980	87.891 (87.435 + 0.456)
mnist7vs9	3.093	638.300	116.440	72.622 (72.155 + 0.467)
mnist1vs7	2.791	465.190	101.235	57.697 (57.220 + 0.477)
mnist4vs9	3.411	597.095	121.038	87.179 (86.765 + 0.414)
real-sim	7.975	1073.755	93.880	129.196 (119.552 + 9.644)
w8a	1.486	888.995	35.172	38.985 (35.091 + 3.894)
ethn	0.247	9.737	2.074	3.521 (3.458 + 0.063)
COIL2	0.698	16.593	20.114	11.506 (11.466 + 0.04)
digit1	0.699	22.700	20.342	11.472 (11.430 + 0.042)



Pre@k F1 AUC

Figure 1: Average performance improvement of UMVP (i.e., (performance value of UMVP - performance value of SVM^{perf})/(performance value of SVM^{perf})) versus number of semi-supervised learners.

On average performance improvement, UMVP achieves performance improvement on all three performance measures. S4VM does not achieve performance improvement on AUC. Self-SVM^{perf} does not achieve performance improvement on Pre@k and AUC. Although UMVP^{-} achieves performance improvement on all three measures, its overall improvement is not as large as UMVP. In terms of win/tie/loss, the UMVP method only has significant performance drop on one data set, while the others all have significant drops in at least 5 cases. In addition, the UMVP method achieves significant improvements in 22 cases, which is the most among all methods. In terms of the statistical significance test (using the Wilcoxon sign test at 95% significance level) of 10 data sets, the UMVP method is superior to the baseline supervised model on all three performance measures, while the other methods do not. In summary, the UMVP method effectively improves the safeness of SSL methods under multiple multivariate performance measures.

Table 3 shows the average training time of all methods on all data sets and performance measures. SVM^{perf} is the fastest, because it is a supervised model using a small labeled data set. UMVP spends most of the time on generating the semi-supervised learners, but its optimization procedure is fast (the number of iterations in Algorithm 1 is usually fewer than 100). Overall, the training time of UMVP is comparable with S4VM, and is more efficient than Self-SVM^{perf}.

Figure 1 shows the effect on the performance of UMVP with different numbers of semi-supervised learners. By setting different numbers of clusters (2, 3, 5, 8, 10) in the k-means clustering algorithm, UMVP obtains different numbers of learners. As can be seen, when the number of learners is relatively large (e.g., $b \geq 5$), UMVP obtains robust performance.

Conclusion and Future Work

Designing safe SSL approaches is challenging. We proposed in this paper the UMVP method that rarely deteriorates its performance in terms of multiple multivariate performance measures when using unlabeled data. The proposed method integrates a number of SSL results, and optimizes its *worst-case* performance gain against a supervised model. This is thus modeled as a maximin optimization. When the Top-k Precision, F_{β} score or AUC is used as the performance measure, a minimax convex relaxation is obtained and this can be efficiently solved. Experimental results demonstrate the ability of the proposed method in improving the safeness of SSL under multiple multivariate performance measures.

In the future, it is worth understanding the sufficient and necessary conditions of safe SSL. Extending safe SSL to more performance measures like MAP (Yue et al. 2007) is also a worthy problem.

References

Balcan, M. F., and Blum, A. 2010. A discriminative model for semi-supervised learning. *Journal of the ACM* 57(3).

Blum, A., and Chawla, S. 2001. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 8th International Conference on Machine Learning*, 19–26.

Blum, A., and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 7th Annual Conference on Computational Learning Theory*.

Chapelle, O.; Schölkopf, B.; and Zien, A., eds. 2006. *Semi-Supervised Learning*. MIT Press.

Chawla, N. V., and Karakoulas, G. 2005. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research* 23:331–366.

Cozman, F. G.; Cohen, I.; and Cirelo, M. 2002. Unlabeled data can degrade classification performance of generative classifiers. In *Proceedings of the 15th International Florida Artificial Intelligence Research Society Conference*, 327–331.

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

Joachims, T. 2005. A support vector method for multivariate performance measures. In *Proceedings of the 22th International Conference on Machine Learning*, 377–384.

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

Kim, S.-J., and Boyd, S. 2008. A minimax theorem with applications to machine learning, signal processing, and finance. *SIAM Journal on Optimization* 19(3).

Li, M., and Zhou, Z.-H. 2005. SETRED: Self-training with editing. In *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 611–621.

Li, Y.-F., and Zhou, Z.-H. 2011a. Improving semi-supervised support vector machines through unlabeled instances selection. In *Proceedings of 25th AAAI Conference on Artificial Intelligence*, 386–391.

Li, Y.-F., and Zhou, Z.-H. 2011b. Towards making unlabeled data never hurt. In *Proceedings of the 28th International Conference on Machine learning*, 1081–1088.

Li, Y.-F., and Zhou, Z.-H. 2015. Towards making unlabeled data never hurt. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37(1):175–188.

Li, Y.-F.; Kwok, J.; Tsang, I.; and Zhou, Z.-H. 2013. Convex and scalable weakly label SVMs. *Journal of Machine Learning Research* 14:2151–2188.

Mallapragada, P.; Jin, R.; Jain, A.; and Liu, Y. 2009. SemiBoost: Boosting for semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(11):2000–2014.

Manning, C.; Raghavan, P.; and Schtze, H. 2008. *Introduction to Information Retrieval*. New York, NY: Cambridge University Press.

Nesterov, Y. 2003. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer.

Nigam, K.; McCallum, A. K.; Thrun, S.; and Mitchell, T. 2000. Text classification from labeled and unlabeled documents using EM. *Machine learning* 39(2):103–134.

Singh, A.; Nowak, R.; and Zhu, X. 2009. Unlabeled data: Now it helps, now it doesn't. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 21*. MIT Press. 1513–1520.

Sra, S.; Nowozin, S.; and Wright, S. J. 2012. *Optimization for Machine Learning*. MIT Press.

Tsochantaridis, I.; Joachims, T.; Hofmann, T.; and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6:1453–1484.

Yang, T., and Priebe, C. 2011. The effect of model misspecification on semi-supervised classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33(10):2093–2103.

Yarowsky, D. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, 189–196.

Yue, Y.; Finley, T.; Radlinski, F.; and Joachims, T. 2007. A support vector method for optimizing average precision. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 271–278.

Zhang, T., and Oles, F. 2000. The value of unlabeled data for classification problems. In *Proceedings of the 17th International Conference on Machine Learning*, 1191–1198.

Zhang, K.; Tsang, I. W.; and Kwok, J. T. 2007. Maximum margin clustering made practical. In *Proceedings of the 24th International Conference on Machine Learning*, 1119–1126.

Zhou, Z.-H., and Li, M. 2010. Semi-supervised learning by disagreement. *Knowledge and Information Systems* 24(3):415–439.

Zhou, Z.-H. 2012. *Ensemble Methods: Foundations and Algorithms*. Boca Raton: FL: Chapman & Hall.

Zhu, X. 2007. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison.