Enabling Precision/Recall Preferences for Semi-supervised SVM Training

Zeyi Wen Rui Zhang Kotagiri Ramamohanarao
University of Melbourne
{zeyi.wen, rui.zhang, kotagiri}@unimelb.edu.au

ABSTRACT

Semi-supervised learning is an essential approach to classification when the available labeled data is insufficient and we need to also make use of unlabeled data in the learning process. Numerous research efforts have focused on designing algorithms to improve the F_1 score, but have any mechanism to control precision or recall individually. However, many applications have precision/recall preferences. For instance, an email spam classifier requires a precision of 0.9 to mitigate the false dismissal of useful emails. In this paper, we propose a method that allows to specify a precision/recall preference while maximising the F_1 score. Our key idea is that we divide the semi-supervised learning process into multiple rounds of supervised learning, and the classifier learned at each round is calibrated using a subset of the labeled dataset before we use it on the unlabeled dataset for enlarging the training dataset. Our idea is applicable to a number of learning models such as Support Vector Machines (SVMs), Bayesian networks and neural networks. We focus our research and the implementation of our idea on SVMs. We conduct extensive experiments to validate the effectiveness of our method. The experimental results show that our method can train classifiers with a precision/recall preference, while the popular semi-supervised SVM training algorithm (which we use as the baseline) cannot. When we specify the precision preference and the recall preference to be the same, which indicates to maximise the F_1 score only as the baseline does, our method achieves better or similar F_1 scores to the baseline. An additional advantage of our method is that it converges much faster than the baseline.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Miscellaneous

General Terms

Support Vector Machines, Precision, Recall

CIKM'14, November 3–7, 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2661977.

Table 1: S^{3}VM with recall preferences

Preferences	our method			the baseline		
	r	p	F_1	r	p	F_1
$r_g = 0.5$	**0.57**	0.60	0.59	0.59	0.63	0.61
$r_g = 0.6$	**0.68**	0.57	0.62			
$r_g = 0.7$	**0.73**	0.55	0.63			
$r_g = 0.8$	**0.77**	0.54	0.63			

Keywords

Semi-supervised SVM training, Precision, Recall, Preference

1. INTRODUCTION

Semi-supervised learning is an essential approach to classification when the available labeled data is insufficient and we need to also make use of unlabeled data in the learning process. Unlabeled data are shown to be useful for training better classifiers in many applications [1, 2]. Numerous research efforts have focused on designing algorithms to improve the F_1 score in semi-supervised classification, but have any mechanism to control precision or recall individually. However, many applications have high precision/recall preferences. An email server needs a classifier to filter out unwanted or harmful content known as spam [3]. The email spam classifier requires a precision of 0.9 (i.e., a *precision preference* of 0.9) to mitigate the dismissal of useful emails. A classifier for cancer detection has a recall preference of 0.95 trying to avoid missing patients with cancer [4]. In this paper, we propose a semi-supervised learning method that allows to specify a precision/recall preference while maximising the F_1 score. Our key idea is that we divide the semi-supervised learning process into multiple rounds of supervised learning, and the classifier learned at each round is calibrated using a subset of the labeled dataset before we use it on the unlabeled dataset for enlarging the training dataset of the next round. Our idea is applicable to a number of learning models such as Support Vector Machines (SVMs) [5], Bayesian networks [6] and neural networks [7]. We focus our research and the implementation of our idea on SVMs. Table 1 gives an example of what our method can achieve in comparison to the popular semi-supervised SVM (S^{3}VM) training algorithm [8] which we use as the baseline in this paper; r_g represents the recall preference, and all the other settings such as kernel parameters of SVMs are the same; the values highlighted by bold font are the recall of our final classifiers. As can be seen from the table, our method can train classifiers with different recall preferences

while the baseline gives precision and recall from the algorithm that does not have mechanism to control the precision and recall individually and only optimises the F_1 score.

The key steps of our method are as follows. (i) We randomly divide the labeled dataset into two subsets $\mathcal{L}$ and $\mathcal{B}$. Let $\mathcal{T}_i$ denote the training dataset in the i^{th} round; $\mathcal{L}$ is used as the initial training dataset, i.e., $\mathcal{T}_1$. (ii) In the i^{th} round, $\mathcal{T}_i$ is used for training an SVM classifier $\mathbb{H}$ and the dataset $\mathcal{B}$ is used for measuring the precision and recall of $\mathbb{H}$. (iii) If the precision (or recall) preference is satisfied, we adjust the decision value of $\mathbb{H}$ to improve the F_1 score. Otherwise, we increase (or decrease) the decision value of $\mathbb{H}$ to improve the precision (or recall). (iv) Then, the classifier $\mathbb{H}$ with the adjusted decision value is used to classify the unlabeled dataset to form a dataset $\mathcal{S}$ of highly confident instances. The result of the union of the dataset $\mathcal{S}$ and the dataset $\mathcal{L}$ is used as the training dataset, $\mathcal{T}_{i+1}$, in the next round. The steps (ii) to (iv) are repeated until the termination condition is reached.

We conduct extensive experiments to validate the effectiveness of our method. When we specify a preference, the other preference is set to be 0 by default. The experimental results show that our method can train classifiers with a precision/recall preference, while the popular S^3VM training algorithm (i.e., Transductive SVM [8] and TSVM for short) does not have any mechanism to allow for preferences. When we specify the precision preference and the recall preference to be the same for our method, which indicates to maximise the F_1 score only as TSVM does, our method achieves better F_1 scores than or similar F_1 scores to TSVM.

An additional advantage of our method is that it converges faster than TSVM. This is because TSVM trains the SVM classifier in the way similar to a combinatorial optimisation approach which requires more training iterations, while our method trains the SVM classifier by dividing the S^3VM training process into a small number of the supervised SVM training processes. Our experimental results show that our method converges at least several times faster than TSVM and in some cases the improvement factor is more than one order of magnitude.

The remainder of the paper is organised as follows. We discuss related work in Section 2, and present preliminaries in Section 3. Following that, we elaborate our method in Section 4 and provide our experimental results in Section 5. Finally, we conclude the paper in Section 6.

2. RELATED WORK

Zhu [9] gives a nice survey on semi-supervised learning. Our study falls into the category called "self-learning" of the semi-supervised learning research field. We focus on semi-supervised SVM training. In what follows, we review the existing studies on supervised SVM training, S^3VM training and partially supervised SVM training, respectively.

Supervised SVM training: Supervised SVM training only uses labeled instances, which is a foundation for S^3VM training as S^3VM training problems can be solved using adopted supervised SVM training algorithms. Osuna et al. [10] proposed a supervised SVM training algorithm based on decomposition. This approach decomposes the training instances into groups. In each training iteration, one group of the instances is used to update the currently found hyperplane. To improve Osuna et al.'s algorithm, Platt [11] proposed the *Sequential Minimal Optimisation* (SMO) algorithm. SMO also uses decomposition, but in each training iteration, only two training instances are used to update the currently found hyperplane. Other SVM training algorithms, such as Joachims' algorithm [12] and "Pegasos" [13], focus on improving the training efficiency for linear SVMs. As SMO is space efficient and fast, and can train both linear and non-linear SVMs, we use it as the supervised SVM training algorithm in our method.

Semi-supervised SVM training: Bennett et al. [14] first introduced S^3VM and the training algorithm. The S^3VM training algorithm attempts to train SVMs using both labeled and unlabeled data. The goal of S^3VM training is to find a hyperplane that separates the two classes of instances in the labeled dataset with the maximum margin and meanwhile, to minimise the number of unlabeled instances falling between the margin. Finding the exact solution to the S^3VM training problem is NP-hard [9], and hence some existing studies [15, 16] improve the efficiency of the S^3VM training algorithm by approximation. The key idea of the S^3VM training algorithms is to convert an S^3VM training process to multiple supervised SVM training processes. These S^3VM training algorithms focus on maximising the F_1 score but cannot train SVMs with a precision/recall preference. Among these algorithms, Transductive SVM (TSVM) [8] is one of the most popular algorithms for S^3VM training and can train both linear and non-linear SVMs. We use TSVM as our baseline algorithm, and will discuss it in Section 3.

Partially supervised SVM training: Another category of work similar to S^3VM training is partially supervised SVM training. Partially supervised SVM training does not require labeled negative instances. "PEBL" [17] is a partially supervised SVM training algorithm based on positive and unlabeled instances. The algorithm first identifies some negative instances from the unlabeled dataset based on some features of the positive instances. Then those identified negative instances are put together with the positive instances to train an SVM classifier using the supervised SVM training algorithm. The trained SVM classifier is used to identify more negative instances from the unlabeled dataset for enlarging the training dataset of the next round of the supervised SVM training. These steps are repeated until no more negative instances are identified. To achieve faster convergence, Fung et al. [18] proposed an algorithm similar to PEBL. Their algorithm not only identifies more negative instances but also identifies more positive instances for enlarging the training dataset of the next round of the supervised SVM training. Liu et al [19] gave a more comprehensive study on partially supervised learning. These studies have different settings from ours that has positive, negative and unlabeled instances. More importantly, we propose a method for S^3VM training with precision/recall preferences.

3. PRELIMINARIES

In this section, we first present the details of supervised SVM training, S^3VM training and SVM classification. Then, we provide our problem definition.

3.1 Supervised SVM training

A labeled training instance $\boldsymbol{x}_i$ is attached with an integer $y_i \in \{+1, -1\}$ as its label. A positive (negative) instance is a training instance with the label of $+1$ (-1). Given a set $\boldsymbol{\mathcal{X}}$ of n training instances, the goal of training SVMs is to find a hyperplane that separates the positive from the

negative instances in $\boldsymbol{\mathcal{X}}$ with the maximum margin, and meanwhile, with the minimum misclassification error on the training instances. The training is equivalent to solving the following optimisation problem:

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\xi},b}{\text{argmin}} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i - b) \geq 1-\xi_i \\ & \xi_i \geq 0,\ \forall i \in \{1,...,n\} \end{aligned} \tag{1}$$

where $\boldsymbol{w}$ is the normal vector of the hyperplane, C is the penalty parameter, $\boldsymbol{\xi}$ is the slack variable vector to tolerant some training instances falling in the wrong side of the hyperplane, and b is the bias of the hyperplane.

To handle the non-linearly separable data, SVMs uses a mapping function to map the training instances from the original data space to a higher dimensional data space where the data may become linearly separable. The optimisation problem 1 can be rewritten to a dual form [20] where mapping functions can be replaced by kernel functions [21] which make the mapping easier. The optimisation problem in the dual form is shown as follows.

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & F(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\boldsymbol{\alpha}^{\boldsymbol{T}}\boldsymbol{Q}\boldsymbol{\alpha} \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \forall i \in \{1,...,n\} \\ & \sum_{i=1}^{n} y_i\alpha_i = 0 \end{aligned} \tag{2}$$

where $F(\boldsymbol{\alpha})$ is the objective function; $\boldsymbol{\alpha} \in \mathbb{R}^n$ is a weight vector, where α_i denotes the *weight* of the training instance $\boldsymbol{x}_i$; $\boldsymbol{Q}$ is a symmetric matrix, where $\boldsymbol{Q} = [Q_{ij}]$, $Q_{ij} = y_iy_jK(\boldsymbol{x}_i,\boldsymbol{x}_j)$ and $K(\boldsymbol{x}_i,\boldsymbol{x}_j)$ is a kernel value computed from a kernel function (e.g., the Gaussian kernel function, $K(\boldsymbol{x}_i,\boldsymbol{x}_j) = exp\{-\gamma||\boldsymbol{x}_i - \boldsymbol{x}_j||^2\}$).

The goal of the training translates to finding a weight vector $\boldsymbol{\alpha}$ that maximises the value of the objective function $F(\boldsymbol{\alpha})$. The training instances with their weights greater than 0 are called *support vectors*. In our method, we use a popular training algorithm, the Sequential Minimal Optimisation (SMO) algorithm as discussed in Section 2. SMO iteratively improves the weight vector until the optimal conditions (i.e., the Karush-Kuhn-Tucker conditions [22]) are met. For more details of SMO, please consult the original paper of SMO [11].

3.2 Semi-supervised SVM training

A labeled instance $\boldsymbol{x}_i$ is attached with an integer $y_i \in \{+1,-1\}$ as its label, while the label $y_j^* \in \{+1,-1\}$ of an unlabeled instance $\boldsymbol{x}_j^*$ is unknown. Given n labeled instances and k unlabeled instances, the S^{3}VM training problem can be formulated as follows.

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\xi},\boldsymbol{\xi}^*,\boldsymbol{y}^*,b}{\text{argmin}} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n}\xi_i + C^*\sum_{j=1}^{k}\xi_j^* \\ \text{subject to} \quad & y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i - b) \geq 1-\xi_i \\ & y_j^*(\boldsymbol{w}\cdot\boldsymbol{x}_j^* - b) \geq 1-\xi_j^* \\ & \xi_i \geq 0, \quad \forall i \in \{1,...,n\} \\ & \xi_j^* \geq 0, \quad \forall j \in \{1,...,k\} \end{aligned} \tag{3}$$

where C and C^* are the penalty parameters of the labeled instances and the unlabeled instances, respectively; $\boldsymbol{\xi}$ and $\boldsymbol{\xi}^*$ are the slack variable vectors to tolerant misclassification for the labeled and unlabeled instances, respectively.

There are many S^{3}VM training algorithms. Here we focus on one of the most popular S^{3}VM training algorithms, i.e., TSVM [8]. Initially, the TSVM algorithm trains an SVM using only the labeled instances. Then the trained SVM classifier is used to classify the unlabeled instances, and to assign each unlabeled instance with a label that we call "soft label" in this paper. Following that, TSVM switches two of the soft-labeled instances and retrains the SVM classifier with all the training instances such that the value of the objective function is reduced. The label switching process and the training process are repeated until the value of the object function is minimised.

Instead of solving the S^{3}VM training problem in the way similar to a combinatorial optimisation approach, we break the S^{3}VM training process into a small number of supervised SVM training processes. The training dataset (except which of the initial training) in TSVM is always all the labeled instances and all the unlabeled instances. In comparison, the training dataset in our method is a subset of the labeled instances and a subset of the unlabeled instances, which makes each training iteration faster. We also design our algorithm to train an SVM classifier with a precision/recall preference. We will detail our method in Section 4.

3.3 SVM classification

After the supervised SVM training or the S^{3}VM training process, the trained SVM classifier can be used to classify an instance $\boldsymbol{x}_l$ using the following equations.

$$\begin{aligned} v &= \sum_{i=1}^{m} y_i\alpha_i K(\boldsymbol{x}_{sv_i},\boldsymbol{x}_l) + b \\ y_l &= \begin{cases} +1 & \text{if } v > \delta, \\ -1 & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

where $\boldsymbol{x}_{sv_i}$ is the i^{th} support vector of the SVM classifier, m is the number of the support vectors, b is the bias of the hyperplane, y_l is the predicted label of $\boldsymbol{x}_l$, and δ is the decision value which equals to 0 by default.

3.4 Problem definition

Given a labeled dataset $\mathcal{M}$, an unlabeled dataset $\mathcal{U}$ and a testing dataset $\mathcal{G}$, S^{3}VM training with a precision (or recall) preference p_g (or r_g) is to train a classifier $\mathbb{H}$ using $\mathcal{M}$ and $\mathcal{U}$ such that $\mathbb{H}$ classifies $\mathcal{G}$ with precision p (or recall r) meeting one of the following conditions:

- if $p < p_g$ (or $r < r_g$), p (or r) is the highest value the S^{3}VM training process can obtain.
- if $p \geq p_g$ (or $r \geq r_g$), F_1 is the highest value the S^{3}VM training process can obtain.

4. S^{3}VM TRAINING WITH PRECISION OR RECALL PREFERENCES

In the section, we elaborate our S^{3}VM training method. As our method can train SVM classifiers with precision/recall preferences, we call our method *P*reference enabled semi-supervised *SVM* training (**PSVM** for short). Our key idea is that we divide the semi-supervised learning process into

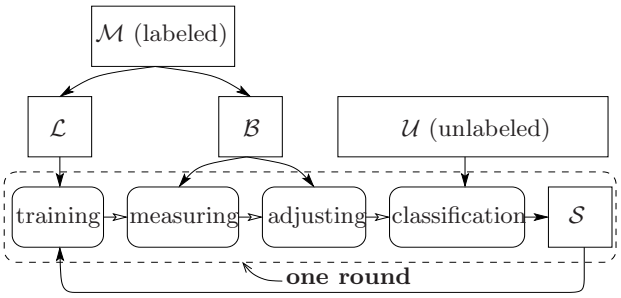


Figure 1: The PSVM method

multiple rounds of supervised learning, and the classifier trained at each round is calibrated using a subset of the labeled dataset before we use it on the unlabeled dataset for enlarging the training dataset of the next round.

The key steps of our method are as follows. (i) We randomly divide the labeled dataset $\mathcal{M}$ ("M" for **m**anually labeled dataset) into two datasets, denoted by $\mathcal{L}$ and $\mathcal{B}$ ("L" for **l**abeled training dataset and "B" for cali*B*ration dataset), respectively. Let $\mathcal{T}_i$ ("T" for **t**raining) denote the training dataset in the i^{th} round; $\mathcal{L}$ is used as the initial training dataset, i.e., $\mathcal{T}_1$. (ii) In the i^{th} round, the dataset $\mathcal{T}_i$ is used for training an SVM classifier $\mathbb{H}$ using the supervised SVM training algorithm. The calibration dataset $\mathcal{B}$ is used for measuring the precision and recall of the classifier $\mathbb{H}$. (iii) If the precision (or recall) preference is satisfied, we adjust the decision value (i.e., δ in Equation 4) of $\mathbb{H}$ to improve the F_1 score. Otherwise, we increase (or decrease) the decision value to improve the precision (or recall). (iv) The classifier $\mathbb{H}$ with the adjusted decision value is used for classifying the instances in the unlabeled dataset $\mathcal{U}$ ("U" for **u**nlabeled). We refer to the unlabeled instances classified by $\mathbb{H}$ as the "soft-labeled" instances. Those soft-labeled instances form a dataset denoted by $\mathcal{S}$ ("S" for **s**oft-labeled). The result of the union of the dataset $\mathcal{S}$ and the labeled training dataset $\mathcal{L}$ forms the training dataset, $\mathcal{T}_{i+1}$, for the next round, and the steps (ii) to (iv) are repeated until the termination condition is achieved. We discuss the termination condition in detail in Section 4.5.1.

A round in PSVM can be summarised in four phases: training, measuring, adjusting and classification. Figure 1 gives an overview of PSVM. Adopting our idea to other models (e.g., Bayesian networks and neural networks) is straightforward except that the adjusting phase needs some special cares. In what follows, we first describe the necessity of a separated calibration dataset. Then, we give the details of the training, measuring, adjusting and classification phases. Finally, we give our overall S^{3}VM training algorithm.

4.1 Necessity of a separated calibration dataset

As we can see from Figure 1, there are three purposes of using the labeled dataset $\mathcal{M}$ at each round:

- forming the labeled training dataset (i.e., dataset $\mathcal{L}$ in Figure 1) for supervised SVM training;
- measuring the precision and recall of the trained SVM classifier (i.e., dataset $\mathcal{B}$ in Figure 1);
- adjusting the decision value for classifying the unlabeled instances (i.e., dataset $\mathcal{B}$ in Figure 1).

In our method, we randomly divide the labeled dataset $\mathcal{M}$ into two subsets (i.e., the labeled training dataset $\mathcal{L}$ and calibration dataset $\mathcal{B}$). We call this approach "**PART**", because $\mathcal{L}$ and $\mathcal{B}$ are part of $\mathcal{M}$. Alternatively, we may use the whole labeled dataset $\mathcal{M}$ for training, measuring and adjusting without division (i.e., $\mathcal{L} = \mathcal{B} = \mathcal{M}$). We call this approach "**FULL**", since both $\mathcal{L}$ and $\mathcal{B}$ contain the whole dataset $\mathcal{M}$. The advantage of the PART approach is that the calibration dataset $\mathcal{B}$ is independent of the training dataset for training the SVM classifier. Hence, the measuring is a better estimation of the effectiveness of the SVM classifier on the unseen data, and the adjusting is more effective than that using the FULL approach. In comparison, using the training dataset to measure the classifier itself (i.e., $\mathcal{L} = \mathcal{B} = \mathcal{M}$) can lead to the over-fitting problem. We conduct experiments to validate the effectiveness of PART in comparison to FULL in Section 5.

4.2 The training phase of PSVM

The training phase in Figure 1 is a standard supervised SVM training process. The training dataset $\mathcal{T}$ for the supervised SVM training is the union of the labeled training dataset $\mathcal{L}$ and the soft-labeled dataset $\mathcal{S}$ (cf. Figure 1). Note that the training dataset for training the first SVM classifier is $\mathcal{L}$, since the dataset $\mathcal{S}$ is empty initially.

We use SMO as the supervised SVM training algorithm. We implement the algorithm using the GPU ("GPU" for the Graphics Processing Unit [23]) based on Catanzaro et al.'s GPU SVM algorithm [24]. The implementation details of the GPU-based algorithm are out of the scope of this paper and hence we will not discuss it any further.

4.3 The measuring phase of PSVM

As shown in Figure 1, after the training phase, we measure the precision, recall and F_1 score of the trained SVM classifier using the calibration dataset $\mathcal{B}$. The measuring phase is a standard classification process on the dataset $\mathcal{B}$. If the computed value v of an instance using Equation 4 is larger than 0 (i.e., $\delta = 0$), then the instance is assigned a label of $+1$. Otherwise the instance is assigned a label of -1. After the classification on all the instances on the dataset $\mathcal{B}$, we compare those assigned labels with the true labels of the instances. Then, we can compute the precision p, recall r and F_1 score using the following equations.

$$\text{precision} \quad p = \frac{tp}{tp + fp} \tag{5}$$

$$\text{recall} \quad r = \frac{tp}{tp + fn} \tag{6}$$

$$F_1 \text{ score} \quad F_1 = \frac{2pr}{p + r} \tag{7}$$

where tp is the number of true positive instances, fp is the number of false positive instances and fn is the number of false negative instances.

The measuring phase has two purposes. The first purpose is to check if the SVM classifier satisfies the precision (or recall) preference. Specifically, we compare the precision p (or recall r) with the precision (or recall) preference p_g (or r_g), and check if the precision p (or recall r) satisfies the precision (or recall) preference, i.e., $p \geq p_g$ (or $r \geq r_g$).

The second purpose of the measuring phase is to identify the best SVM classifier $\mathbb{H}_{best}$ for the final result of the S^{3}VM

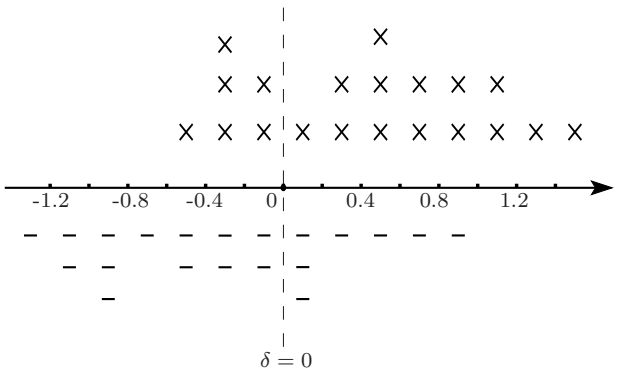


Figure 2: Adjusting δ to satisfy the preference

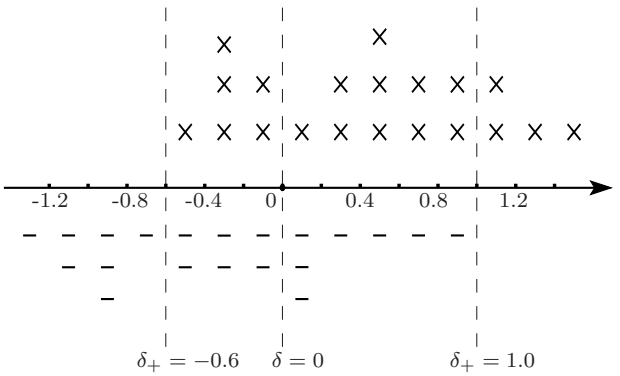


Figure 3: Adjusting δ to improve F_1

training process. There are two cases when we identify the best SVM classifier $\mathbb{H}_{best}$.

- **Case 1:** the precision (or recall) preference is not satisfied. We choose the SVM classifier with the largest precision (or recall) among all the trained SVM classifiers during the S^{3}VM training process.
- **Case 2:** the precision (or recall) preference is satisfied. We choose the SVM classifier with the largest F_1 score among all the trained classifiers that satisfy the precision (or recall) preference.

4.4 The adjusting phase of PSVM

After the measuring phase, we can decide to improve the precision (or recall) or to improve the F_1 score (i.e., the first purpose of the measuring phase). When we adjust the decision value δ in Equation 4, we have two cases as follows:

- The trained SVM classifier at the current round does not satisfy the precision (or recall) preference. We increase (or decrease) the decision value δ such that the classification result of the classifier on the calibration dataset $\mathcal{B}$ satisfies the precision (or recall) preference.
- The trained SVM classifier at the current round satisfies the precision (or recall) preference. We adjust the decision value δ to improve the F_1 score.

Next, we first describe the technique to adjust the decision value for improving precision (or recall) to satisfy the precision (or recall) preference. Then, we explain the approach to adjusting the decision value for improving the F_1 score. Finally, we discuss adjusting the decision value for identifying the negative instances from the unlabeled dataset $\mathcal{U}$ more confidently.

4.4.1 Satisfying the precision/recall preference

If the trained SVM classifier at the current round does not satisfy the precision (or recall) preference, we can increase (or decrease) the decision value δ of the SVM classifier to improve its precision (or recall). Note that when we have a precision (or recall) preference, the recall (or precision) preference r_g (or p_g) equals to 0.

Without loss of generality, we analyse the approach for increasing the precision p. Let us first rewrite Equation 5 into the following equation: $p = \frac{1}{1 + \frac{fp}{tp}}$. To increase precision p, we need to reduce $\frac{fp}{tp}$ (i.e., the ratio of the number of false positive and that of true positive). We can increase the decision value δ of Equation 4 to reduce $\frac{fp}{tp}$. This is because a positive instance is likely to have a larger v value while a negative instance is likely to have a smaller v value (i.e., the property of SVMs). As a result, by increasing δ, the number of false positive instances decreases more significantly than that of true positive instances. Hence, $\frac{fp}{tp}$ decreases and precision p increases. Similarly, we can decrease δ to increase recall r.

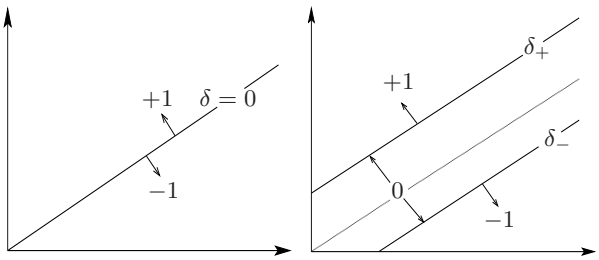
In what follows, we describe the approach for computing the new decision value δ_+. Figure 2 shows an example of the classification results on the calibration dataset $\mathcal{B}$, where a "x" represents a positive instance and a "-" represents a negative instance in the calibration dataset $\mathcal{B}$. The dashed line indicates the decision value δ which equals to 0. The position of an instance indicates the v value of the instance. As can be seen from the figure, 14 positive instances have v values greater than δ (i.e., true positive), and 7 negative instances have v values greater than δ (i.e., false positive). So, the precision is $p = \frac{14}{14+7} = 0.67$. Similarly, the recall is $r = \frac{14}{14+6} = 0.7$ on the calibration dataset $\mathcal{B}$. Suppose we would like to train an SVM classifier with a precision preference of 0.7 (i.e., $p_g = 0.7$). In this example, we increase the decision value from 0 to 0.2 (i.e., $\delta_+ = 0.2$) to get the precision of $\frac{13}{13+4} = 0.76$ at the cost of decreasing recall to $\frac{13}{13+7} = 0.65$.

Instead of having a precision preference, suppose we would like to train an SVM classifier with a recall preference of 0.9 (i.e., $r_g = 0.9$). We decrease the decision value from 0 to -0.4 (i.e., $\delta_+ = -0.4$) to get the recall of $\frac{15}{15+1} = 0.94$ at the cost of decreasing precision to $\frac{19}{19+11} = 0.63$.

After we obtain the new decision value δ_+, we expect that the classification result of the SVM classifier using the decision value δ_+ on the calibration dataset $\mathcal{B}$ satisfies the precision (or recall) preference.

4.4.2 Improving the F_1 score

There are two scenarios that we need to improve the F_1 score. First, the trained classifier satisfies the precision (or recall) preference, but we want to improve its F_1 score. Sec-



(a) Original boundary (b) Adjusted boundaries

Figure 4: Adjusting decision boundary

ond, we would like to train a classifier with as high F_1 score as possible without having a specific precision (or recall) preference. These two scenarios are essentially the same, i.e., we have the precision preference p_g and the recall preference r_g to be the same (e.g., both to be 1) in the adjusting phase. In Figure 3, the decision value for satisfying the precision preference p_g is the rightmost dashed line where δ_+ equals to 1.0, while the decision value for satisfying the recall preference r_g is the leftmost dashed line where δ_+ equals to -0.6. As we cannot find a decision value which satisfies both of the preferences, we adjust the decision value δ to a value that improves the F_1 score. The approach that we use to improve the F_1 score is to reduce the gap between precision p and recall r, i.e., the value of $|p - r|$.

THEOREM 1. *Given p, r and $p + r = z$ where z is a constant, the F_1 score is maximised when p equals to r.*

The proof is straightforward and omitted. Although z may not be exactly a constant in S^{3}VM training, the theorem gives a direction of improving the F_1 score. Intuitively, while improving the F_1 score, our method treats the precision and recall equally and minimises both the number of false positive instances and the number of false negative instances. In the example in Figure 3, we improve the precision since the precision is smaller than the recall (0.67 v.s. 0.7). To handle this case, our method temporarily specify the precision preference to be 0.7 (i.e., the recall of the trained classifier) and compute the decision value δ_+ using the same idea we discussed in Section 4.4.1.

4.4.3 Identifying negative instances more confidently

So far, we only consider the decision value δ_+ for deciding the positive instances. In other words, v values of the positive instances should be larger than δ_+. All the instances with v values no larger than δ_+ are identified as negative instances. It is also important to identify the negative instances more confidently. We use a decision value δ_- to identify negative instances. In our method, δ_- equals to the average v value of all the negative instances in the calibration dataset $\mathcal{B}$. Please note that the decision value δ_- for negative instances has no effect on adjusting the precision and recall of the SVM classifiers.

Figure 4 shows an SVM classifier before and after the decision value adjusting. The SVM with the decision values δ_+ and δ_- is applied to classify the instances in the unlabeled dataset $\mathcal{U}$. In what follows, we discuss the classification phase of PSVM on the unlabeled dataset.

4.5 The classification phase of PSVM

We use the SVM classifier with the adjusted decision values to classify an unlabeled instance $\boldsymbol{x}_j$ to be positive, negative or uncertain. The label y_j of the instance $\boldsymbol{x}_j$ can be predicted by the following equation.

$$y_j = \begin{cases} +1 & \text{if } v > \delta_+, \\ -1 & \text{if } v < \delta_-, \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

We refer to these labeled instances by the SVM classifiers as soft-labeled instances and they form a dataset denoted by $\mathcal{S}$. The dataset $\mathcal{S}$ is used as part of the training dataset in the next round of the S^{3}VM training process. Next, we describe two approaches to forming the $\mathcal{S}$ dataset.

4.5.1 Forming the soft-labeled dataset $\mathcal{S}$

As we can see from Figure 1, each round of the S^{3}VM training process outputs soft-labeled instances to $\mathcal{S}$. We provide the following two approaches for forming the soft-labeled dataset $\mathcal{S}$. (i) The soft-labeled dataset $\mathcal{S}$ is emptied after the training phase at each round and accepts the soft-labeled instances from the classification phase. (ii) The soft-labeled dataset $\mathcal{S}$ unions itself with the soft-labeled instances from the classification phase at each round, and the instances in $\mathcal{S}$ are removed from the unlabeled dataset $\mathcal{U}$ forming a new set of unlabeled data. We call the first approach "Reconstruction" denoted by **REC**, since each round $\mathcal{S}$ is reconstructed. We refer to the second approach "Increment" denoted by **INC**, because each round $\mathcal{S}$ is increased based on the soft-labeled dataset $\mathcal{S}$ of the previous round.

Termination: A termination condition for both INC and REC is that the trained SVM classifier (denoted by $\mathbb{H}_c$) at the current round, is identical to any one of the SVM classifiers trained in the previous rounds, denoted by $\mathbb{H}_p$. Under this condition, for the INC approach, the classification phase cannot identify any more instances as positive or negative from the unlabeled dataset, since all the possible positive or negative instances have been identified by $\mathbb{H}_p$ and already stored in $\mathcal{S}$. For the REC approach, the classification phase using $\mathbb{H}_c$ outputs the same soft-labeled dataset as that output by $\mathbb{H}_p$.

Compared with the REC approach, the INC approach has the following advantages. Firstly, the INC approach has an additional termination condition that the classification phase cannot identify any new soft-labeled instances. This leads to much faster convergence of the S^{3}VM training process. Secondly, those soft-labeled instances are of highly confident and tend to be labeled with the same labels as the previous rounds. Hence, we do not need to classify them again and can reduce the computation cost. One drawback of the INC approach is the unrecoverable mistake. For instance, an unlabeled instance is wrongly labeled as a positive instance by an SVM classifier. In the INC approach, that instance is always used as a positive instance in the rest of the S^{3}VM training process, and there is no chance to correct it. In contrast, that wrongly labeled instance may be corrected by other SVMs classifier if we use the REC approach. Due to the SVM's good property of error-tolerance, some wrongly labeled instances actually are allowed. Furthermore, the instances in the soft-labeled dataset $\mathcal{S}$ are of highly confident and tend to be labeled with the same labels in different rounds.

In summary, we recommend the INC approach which makes the S^{3}VM training converge faster. We conduct experiments to validate our claim in our experimental study.

Algorithm 1: S^{3}VM training in PSVM

Input: $\mathcal{L}$ and $\mathcal{B}$: sets of labeled instances;
$\mathcal{U}$: a set of unlabeled instances;
C: a penalty parameter; γ: a kernel parameter;
p_g and r_g: precision and recall preferences.
Output: an SVM classifier $\mathbb{H}_{best}$

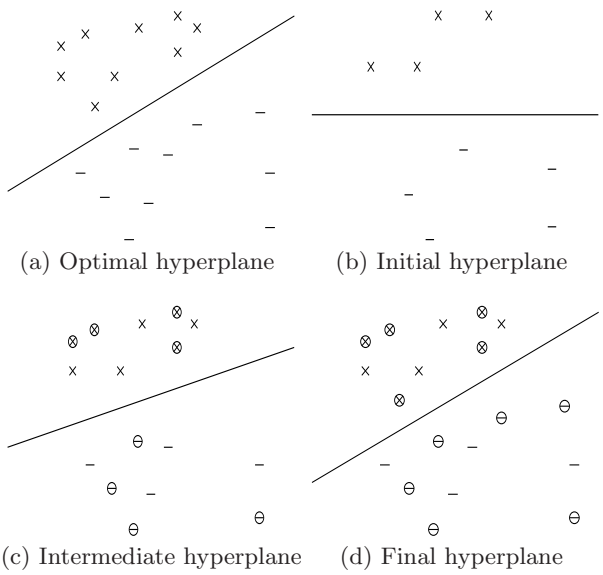
```
1  S'+ := ∅ , S'- := ∅        /* soft-labeled instance sets */
2  repeat
3      S+ := S'+ , S- := S'-, S := S+ ∪ S-
4      H ← train(L, S, C, γ)        /* supervised training */
5      H_best ← measureClassifier(H, B, H_best)
6      adjustDecisionValue(H, B, δ+, δ-, p_g, r_g)
7      foreach x ∈ U do          /* for each unlabeled x */
8          if x ∉ S then              /* not been labeled */
9              v ← classify(H, x)
10             if v > δ+ then
11                 S'+ := S+ ∪ {x}
12             else if v < δ- then
13                 S'- := S- ∪ {x}
14 until S'+ ⊆ S+ ∧ S'- ⊆ S-          /* no x is added */
```

4.6 The overall algorithm of PSVM

In what follows, we describe the overall S^{3}VM training algorithm of PSVM. Without loss of generality, we assume the Gaussian kernel function is used in the training algorithm and using the INC approach for forming the soft-labeled dataset. The pseudo-code of our S^{3}VM training algorithm is summarised in Algorithm 1. Initially, the soft-labeled dataset $\mathcal{S}$ (line 3) is empty. In the supervised SVM training, the training dataset includes the soft-labeled dataset $\mathcal{S}$ and the labeled training dataset $\mathcal{L}$ (line 4). After an SVM $\mathbb{H}$ is trained, we measure the precision, recall and F_1 score of the trained SVM $\mathbb{H}$, compare it with the identified best SVM $\mathbb{H}_{best}$, and keep the better SVM to $\mathbb{H}_{best}$ (line 5). Based on the precision (or recall) preference p_g (or r_g), we compute two decision values δ_+ and δ_- (i.e., adjusting the decision value) of the trained SVM using the calibration dataset $\mathcal{B}$ (line 6). For each unlabeled instance that is not in the soft-labeled dataset $\mathcal{S}$, we compute the value v using Equation 4, and use Equation 8 to classify the instance (lines 7 to 13). The above steps are repeated until no instance in the unlabeled dataset is identified as a positive or negative instance by the SVM classifier with the adjusted decision value.

The number of the supervised SVM training processes is determined by the classification phase (line 9). In the worse case, there is only one instance added to the soft-labeled dataset $\mathcal{S}$ (lines 7 to 13) at each round of the S^{3}VM training. Hence, the number of rounds at the worse case equals to the number of unlabeled instances. In the average case, many of the unlabeled instances are added to the soft-labeled dataset at each round, and the number of rounds is much smaller than the number of unlabeled instances.

As discussed in Section 4.5.1, the REC approach can be used to form the soft-labeled dataset $\mathcal{S}$. The implementation for this method is straightforward. We just need to add a line right after line 4 to empty $\mathcal{S}_+$, $\mathcal{S}_-$, $\mathcal{S}'_+$, $\mathcal{S}'_-$ and $\mathcal{S}$. The termination condition is changed to that the currently trained SVM $\mathbb{H}$ is identical to any one of the previous SVM classifiers.



(a) Optimal hyperplane (b) Initial hyperplane

(c) Intermediate hyperplane (d) Final hyperplane

Figure 5: Hyperplane changes in PSVM

4.6.1 A running example of PSVM

To show the intuition of our method, we give an example. The optimal hyperplane for the given dataset is shown in Figure 5a. Figure 5b shows the hyperplane found only using the labeled training dataset $\mathcal{L}$. After adding some soft-labeled instances (denoted by circles containing "x" or "-"), the hyperplane is improved to the one shown in Figure 5c. As the S^{3}VM training proceeds, the hyperplane shown in Figure 5d approaches close to the optimal hyperplane shown in Figure 5a.

5. EXPERIMENTAL STUDY

In this section, we empirically evaluate our PSVM method. A supervised SVM traing method and an S^{3}VM training method (i.e., Transductive SVM [8]), denoted by SSVM and TSVM respectively, are served as our baselines. We implemented PSVM and SSVM using GPUs in CUDA-C [23], based on Catanzaro et al.'s GPU SVM implementation [24]. TSVM is written in C and downloaded from the SVM^{light} site[1]. Four datasets from the libSVM site[2] are used in our experiments. The cardinality of the datasets varies from 17,766 to 60,000 and the dimensionality of them varies from 22 to 780. All the SVM implementations use the Guassian kernel function and the same parameters which are selected using the grid search [25]. The information of the datasets and the parameters are listed in Table 2. We randomly sample 10% of each dataset to form the testing dataset to validate the effectiveness of the finally obtained classifier. The size of the labeled dataset $\mathcal{M}$ is 10% of the whole dataset by default, and the remaining instances form the unlabeled dataset $\mathcal{U}$. In PSVM, a half of the labeled dataset $\mathcal{M}$ is used as the labeled training dataset $\mathcal{L}$, and the other half is used as the calibration dataset $\mathcal{B}$. SSVM uses only the labeled

[1] http://svmlight.joachims.org/
[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Table 2: The datasets and parameters

Dataset	Card.	Dim.	C	γ
Adult	32,542	123	100	0.5
IJCNN1	49,990	22	10	0.125
MNIST	60,000	780	10	0.125
Protein	17,766	357	10	0.25

Table 3: Precision preferences (Adult)

precision preference p_g	0.55	0.6	0.65	0.7
precision	**0.549**	**0.633**	**0.666**	**0.672**
recall	0.736	0.503	0.391	0.370
F_1	0.628	0.558	0.491	0.475

Table 4: Recall preferences (Adult)

recall preference r_g	0.5	0.6	0.7	0.8
recall	**0.572**	**0.675**	**0.734**	**0.770**
precision	0.603	0.573	0.552	0.535
F_1	0.587	0.619	0.630	0.630

Table 5: F_1 score comparison

Dataset	PSVM	TSVM	SSVM
Adult	0.63±0.015	0.61	0.51
IJCNN1	0.75±0.01	0.76	0.65
MNIST	0.94±0.01	0.93	0.84
Protein	0.51±0.02	0.51	0.24

dataset $\mathcal{M}$ as its training dataset, and TSVM uses $\mathcal{M}$ as the labeled training dataset and $\mathcal{U}$ as the unlabeled training dataset. All the experiments are conducted on a desktop computer running Linux with a Xeon E5-2643 CPU, 32GB main memory and a Tesla C2075 GPU.

In what follows, we first show that PSVM can train classifiers with precision/recall preferences. Then, we investigate the effectiveness of PSVM on maximising the F_1 score and the efficiency of PSVM. Finally, we study the effect of each proposed technique.

5.1 Enabling precision/recall preferences

We use the Adult dataset as the representative to demonstrate the training with precision/recall preferences of PSVM. The experiments on the other datasets showed similar results which are omitted.

To validate the effectiveness of training classifiers with precision preferences, we specify the precision preference p_g to be 0.55, 0.6, 0.65 and 0.7, respectively, and specify r_g to be 0. As can be seen from Table 3, the precision preference can be satisfied when the precision preference is between 0.55 and 0.65. While the precision preference is too high (i.e., at 0.7), PSVM cannot satisfy the preference but tries its best to satisfy the preference. In fact, when the precision preference is specified to be 1, the preference is unlikely to be satisfied for the chosen hyper-parameters C and γ. However, specifying precision preference to be 1 indicates to train a classifier with the maximum precision. This is useful when we would like to train a classifier with as high precision as possible.

Next, we show the results of training classifiers with recall preferences. Table 4 shows the results on specifying the recall preference r_g to be 0.5, 0.6, 0.7 and 0.8, respectively, and on specifying p_g to be 0. As can be seen from the table, when the recall preference is between 0.5 and 0.7, PSVM can train classifiers satisfying the preferences. When the recall preference is specified to be 0.8, PSVM cannot satisfy the preference but it tries its best to train a classifier with recall of 0.77.

Note that SSVM and TSVM do not have any mechanism to control the precision/recall individually, and give only the precision and recall of 0.66 and 0.41, and 0.63 and 0.59, respectively.

5.2 Maximising the F_1 score

To validate the effectiveness of PSVM on maximising F_1 only, we specify both the precision and recall preferences to be 1 (i.e., both of the preferences are the same). We compare the F_1 scores of the three methods using the four datasets. The results in Table 5 are the average values of the results obtained by repeating the experiments 20 times. The variance (e.g., ±0.01) is because of the random sampling for constructing the datasets in the experiments. As can be seen from Table 5, PSVM significantly outperforms SSVM by over 10% of improvement on the F_1 score. This demonstrates the usefulness of using the unlabeled data for training SVM classifiers. Compared with TSVM, PSVM achieves better or similar F_1 scores. This indicates that PSVM is a competitive method for S^3VM training.

5.3 Efficiency of PSVM

We also conducted a set of experiments to show the efficiency of the two S^3VM training methods, i.e., PSVM and TSVM. Figure 6 gives the result on the efficiency comparison. As can be seen from the figure, PSVM constantly outperforms TSVM by three orders of magnitude. In the experiment, we observed that TSVM took more than 20 hours for the MNIST dataset while PSVM only took around 8 minutes to complete the whole semi-supervised training process and both methods trained a similar classifier (F_1 score: 0.93 v.s. 0.94). This demonstrates that PSVM is highly efficient and is a promising semi-supervised training method.

As the speedup factor is achieved by both the computation power of the GPU and the fast convergence of PSVM, we show the experimental results to inference the speedup contributed from the fast convergence. Table 6 shows the total number of the training iterations in the whole semi-supervised training. In all the datasets tested, the total number of the training iterations of PSVM is several times smaller than that of TSVM. On the IJCNN1 dataset, the total number of training iterations of PSVM is 10 times less than that of TSVM. Note that a training iteration of

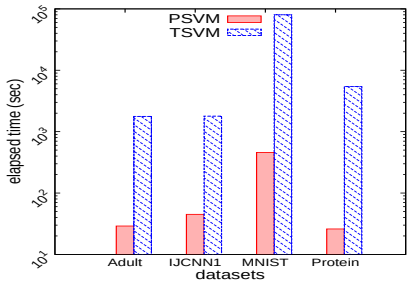


Figure 6: Efficiency comparison

Table 6: The total number of training iterations

Dataset	PSVM	TSVM
Adult	37,976	151,018
IJCNN1	20,913	278,098
MNIST	224,368	652,377
Protein	83,262	190,096

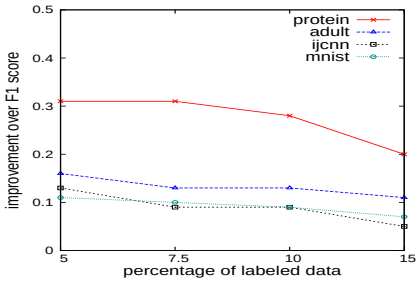


Figure 7: Improvement over the F_1 score

PSVM is faster than a training iteration in TSVM because of the smaller training dataset, as we discussed in Section 3.2. Catanzaro et al. [24] reported that the GPU implementation of SVMs is one to two orders of magnitude faster than the CPU implementation of SVMs. Therefore, we can infer that the reduction of the total number of the training iterations (i.e., faster convergence) contributes to around one order of magnitude speedup.

5.4 Effects of the proposed techniques

In what follows, we first investigate the effects of the size of the labeled dataset $\mathcal{M}$. Then we study the effect of using a calibration dataset $\mathcal{B}$ which is independent of the training dataset. Following that, we compare the results of two techniques for forming the soft-labeled dataset $\mathcal{S}$. Finally, we show the experimental results on the effect of adjusting the decision values.

5.4.1 Effect of the size of the labeled dataset $\mathcal{M}$

This set of experiments is to demonstrate the effect of the size of the labeled dataset $\mathcal{M}$. We vary the size of the labeled dataset $\mathcal{M}$ by varying the percentage of the whole dataset that is used to form $\mathcal{M}$. The percentage varies from 5% to 15%. The experimental results on TSVM are similar to the results on PSVM and hence are omitted. We use SSVM as the base and compute the improvement indicator using the following equation:

$$imp = F_1^{PSVM} - F_1^{SSVM}$$

In the experiments on the four datasets, we noticed that the F_1 score of SSVM increases by around 10% while the F_1 score of PSVM increases by less than 5%. This indicates that the size of the labeled dataset $\mathcal{M}$ has a more significant impact on the F_1 score of SSVM while PSVM is relatively insensitive to the size of the labeled dataset, thanks to the usage of the unlabeled dataset. Hence, the improvement indicator *imp* is expected to decrease as the size of $\mathcal{M}$ increases. The phenomenon is shown in Figure 7.

5.4.2 Effect of using the calibration dataset $\mathcal{B}$

As discussed in Section 4.1, we can use the whole labeled dataset $\mathcal{M}$ for the training, measuring and adjusting (i.e., the FULL approach), instead of taking a half of the labeled dataset to form the calibration dataset $\mathcal{B}$ (i.e., the PART approach). We denote our method that uses the PART approach by PSVM-part and the method that uses the FULL approach by PSVM-full. Table 7 shows the results on F_1. As we can see from the table, PSVM-part significantly outperforms PSVM-full on the MNIST and Protein datasets, and PSVM-part has similar F_1 scores to PSVM-full on the Adult and IJCNN1 datasets. PSVM-part demonstrates the robustness among the datasets tested, while PSVM-full suffers from the over-fitting problem on the MNIST and Protein datasets. Although PSVM-full has a larger labeled training dataset, our experiment on the effect of the dataset size of $\mathcal{M}$ show that PSVM is insensitive to the size of the labeled dataset. Therefore, we recommend the PSVM-part method which is more robust.

Table 7: Effect of F_1 using PART and FULL

dataset	PSVM-part	PSVM-full
Adult	0.63	0.61
IJCNN1	0.75	0.79
MNIST	0.94	0.84
Protein	0.51	0.24

Table 8: Effect of F_1 using INC and REC

dataset	PSVM-inc	PSVM-rec
Adult	0.63	0.63
IJCNN1	0.75	0.74
MNIST	0.94	0.94
Protein	0.51	0.50

5.4.3 Effect of forming the soft-labeled dataset $\mathcal{S}$

As discussed in Section 4.5.1, we can incrementally enlarge the soft-labeled dataset $\mathcal{S}$ (i.e., the INC approach). We can also empty $\mathcal{S}$ after the training phase and reconstruct the dataset at each round (i.e., the REC approach). We denote our method using the INC approach by PSVM-inc, and denote that using the REC approach by PSVM-rec. As can be seen from Table 8, the F_1 scores of PSVM-inc and PSVM-rec are almost the same. In the experiment, we noticed that the PSVM-inc is several times faster than PSVM-rec. This is because the convergence is slower for PSVM-rec than PSVM-inc which has one more termination condition as discussed in Section 4.5.1. We recommend using the PSVM-inc method which is more efficient but retains similar effectiveness of the classifiers.

5.4.4 Effect of adjusting decision values

Here, we show experimental results to validate the effect of adjusting the decision values. We compare our method with decision value adjusting with two methods that do not adjust the decision value δ. Specifically,

- PSVM is the method that uses a separated calibration dataset (i.e., the PART approach) and adjusts the decision value.
- PSVM-part-NA is the method that uses the PART approach but does *not* adjust the decision value.
- PSVM-full-NA is the method that uses the FULL approach and does *not* adjust the decision value.

Table 9: Effect of adjusting decision values

dataset	PSVM	PSVM-part-NA	PSVM-full-NA
Adult	0.63	0.51	0.51
IJCNN1	0.75	0.68	0.66
MNIST	0.94	0.78	0.83
Protein	0.51	0.25	0.24

All these methods use the INC approach to forming the soft-labeled dataset $\mathcal{S}$ as INC is efficient. PSVM-part-NA and PSVM-full-NA do not adjust the decision value and hence the decision value δ equals to 0.

As can be seen from Table 9, PSVM significantly outperforms the methods without adjusting the decision value on all the datasets tested. This demonstrates the importance of adjusting the decision value in the S^3VM training, since by adjusting the decision value we only add highly confident instances to the soft-labeled dataset $\mathcal{S}$.

6. CONCLUSION

Semi-supervised learning is an essential approach to classification when the available labeled data is insufficient and we need to also make use of unlabeled data in the learning process. Numerous research efforts have focused on designing algorithms to improve the F_1 score, but have any mechanism to control precision or recall individually. In this paper, we proposed a method called PSVM that allows to specify a precision/recall preference while maximising the F_1 score. Our key idea is that we divide the semi-supervised learning process into multiple rounds of supervised learning, and the classifier learned at each round is calibrated using a subset of the labeled dataset before we use it on the unlabeled dataset for enlarging the training dataset. Our idea is applicable to a number of learning models such as SVMs, Bayesian networks and neural networks. We focused our research and the implementation of our idea on SVMs. We conducted extensive experiments to validate the performance of our method. The experimental results showed that our method can train classifiers with a precision/recall preference, while the S^3VM training baseline algorithm cannot. When we specified both the precision preference and the recall preference to be the same, which indicates to maximise the F_1 score only as the baseline does, our method achieved better or similar F_1 scores to the baseline. An additional advantage of our method is one order of magnitude faster than TSVM.

Acknowledgements

This work is supported in part by the Australian Research Council (ARC) Discovery Project DP130104587. Dr. Rui Zhang is supported by the ARC *Future Fellowships Project* FT120100832.

7. REFERENCES

[1] Tong Zhang and F Oles. The value of unlabeled data for classification problems. In *ICML*, pages 1191–1198. Citeseer, 2000.

[2] Matthias Seeger et al. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

[3] D Sculley and Gabriel M Wachman. Relaxed online svms for spam filtering. In *SIGIR conference on Research and development in information retrieval*, pages 415–422, 2007.

[4] Susan C Harvey, Berta Geller, Robert G Oppenheimer, Melanie Pinet, Leslie Riddell, and Brian Garra. Increase in cancer detection and recall rates with independent double interpretation of screening mammography. *American Journal of Roentgenology*, 180(5):1461–1467, 2003.

[5] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[6] Ira Cohen, Nicu Sebe, FG Gozman, Marcelo Cesar Cirelo, and Thomas S Huang. Learning bayesian network classifiers for facial expression recognition both labeled and unlabeled data. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–595. IEEE, 2003.

[7] Judith E Dayhoff. *Neural network architectures: an introduction*. Van Nostrand Reinhold Co., 1990.

[8] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

[9] Xiaojin Zhu. Semi-supervised learning literature survey. *University of Wisconsin-Madison*, 2:3, 2006.

[10] Edgar Osuna, Robert Freund, and Federico Girosi. An improved training algorithm for support vector machines. In *Proceedings of the 1997 IEEE Workshop, Neural Networks for Signal Processing VII.*, pages 276–285, 1997.

[11] John C. Platt. Fast training of SVMs using sequential minimal optimization. In *Advances in kernel methods*, pages 185–208. MIT Press, 1999.

[12] Thorsten Joachims. Training linear SVMs in linear time. In *KDD*, pages 217–226, 2006.

[13] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

[14] Kristin Bennett, Ayhan Demiriz, et al. Semi-supervised support vector machines. *Advances in Neural Information processing systems*, pages 368–374, 1999.

[15] Ayhan Demiriz and Kristin P Bennett. Optimization approaches to semi-supervised learning. In *Complementarity: Applications, Algorithms and Extensions*, pages 121–141. Springer, 2001.

[16] Tijl De Bie and Nello Cristianini. Semi-supervised learning using semi-definite programming. *Semi-supervised learning. MIT Press, Cambridge-Massachussets*, 32, 2006.

[17] Hwanjo Yu, Jiawei Han, and Kevin Chen-Chuan Chang. PEBL: positive example based learning for web page classification using SVM. In *KDD*, pages 239–248, 2002.

[18] Gabriel Pui Cheong Fung, Jeffrey Xu Yu, Hongjun Lu, and Philip S Yu. Text classification without negative examples revisit. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):6–20, 2006.

[19] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Building text classifiers using positive and unlabeled examples. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 179–186. IEEE, 2003.

[20] Kristin P Bennett and Erin J Bredensteiner. Duality and geometry in svm classifiers. In *ICML*, pages 57–64, 2000.

[21] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Workshop on Computational Learning Theory*, pages 144–152, 1992.

[22] Geoff Gordon and Ryan Tibshirani. Karush-kuhn-tucker conditions. *Optimization*, 10(725/36):725.

[23] CUDA NVIDIA. NVIDIA CUDA programming guide, 2011.

[24] Bryan Catanzaro, Narayanan Sundaram, and Kurt Keutzer. Fast SVM training and classification on graphics processors. In *ICML*, pages 104–111, 2008.

[25] Steven M LaValle, Michael S Branicky, and Stephen R Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8):673–692, 2004.