New Incremental Learning Algorithm for Semi-Supervised Support Vector Machine

Bin Gu
Department of Electrical & Computer Engineering
University of Pittsburgh
bin.gu@pitt.edu

Xiao-Tong Yuan
School of Information and Control
Nanjing University of Information Science & Technology
xtyuan1980@gmail.com

Songcan Chen
College of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
s.chen@nuaa.edu.cn

Heng Huang*
Department of Electrical & Computer Engineering
University of Pittsburgh
heng.huang@pitt.edu

ABSTRACT

Semi-supervised learning is especially important in data mining applications because it can make use of plentiful unlabeled data to train the high-quality learning models. Semi-Supervised Support Vector Machine (S^3VM) is a powerful semi-supervised learning model. However, the high computational cost and non-convexity severely impede the S^3VM method in large-scale applications. Although several learning algorithms were proposed for S^3VM, scaling up S^3VM is still an open problem. To address this challenging problem, in this paper, we propose a new incremental learning algorithm to scale up S^3VM (IL-S^3VM) based on the path following technique in the framework of Difference of Convex (DC) programming. The traditional DC programming based algorithms need multiple outer loops and are not suitable for incremental learning, and traditional path following algorithms are limited to convex problems. Our new IL-S^3VM algorithm based on the path-following technique can directly update the solution of S^3VM to converge to a local minimum within one outer loop so that the efficient incremental learning can be achieved. More importantly, we provide the finite convergence analysis for our new algorithm. To the best of our knowledge, our new IL-S^3VM algorithm is the first efficient path following algorithm for a non-convex problem (*i.e.*, S^3VM) with local minimum convergence guarantee. Experimental results on a variety of benchmark datasets not only confirm the finite convergence of IL-S^3VM, but also show a huge reduction of computational time compared with existing batch and incremental learning algorithms, while retaining the similar generalization performance.

CCS CONCEPTS

• **Theory of computation → Semi-supervised learning**; • **Computing methodologies → Machine learning**;

KEYWORDS

Semi-Supervised Support Vector Machine, incremental learning, path following algorithm, difference of convex programming

ACM Reference Format:
Bin Gu, Xiao-Tong Yuan, Songcan Chen, and Heng Huang. 2018. New Incremental Learning Algorithm for Semi-Supervised Support Vector Machine. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220092

1 INTRODUCTION

In real-world data mining applications, labeling samples is an expensive task because it usually needs a skilled or professional human agent. For example, medical image analysis [9] usually needs an experienced radiologist to label a large number of medical images which is quite expansive. However, many applications have plentiful unlabeled data, but limited labeled samples such that supervised learning models cannot achieve satisfied performance. To address this issue, many semi-supervised learning methods, including semi-supervised support vector machine (S^3VM) [6], were proposed to utilize both labeled and unlabeled data to train the high-quality learning models.

Among the semi-supervised learning methods, S^3VM is one of the most powerful ones. S^3VM targets to learn a low-density separator by maximizing the margin over the labeled and unlabeled samples. Given the training dataset constituted with a labeled dataset $L = \{(x_1, y_1), \cdots, (x_l, y_l)\}$ and an unlabeled dataset $U = \{x_{l+1}, \cdots, x_{l+u}\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$. Let $f(x_i) = \langle w, \phi(x_i)\rangle + b$ [1] denote the discrimination hyperplane, S^3VM [8] solves the following problem:

$$\min_{w,b} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{l} h_1(y_i f(x_i)) + C^* \sum_{i=l+1}^{l+u} h_1(|f(x_i)|) \qquad (1)$$

where $h_t(\cdot) = \max(0, t - \cdot)$ is the hinge loss, $h_t(|\cdot|)$ is the symmetric hinge loss, and C and C^* are predefined parameters. Especially, for

*To whom all correspondence should be addressed. B. Gu and H. Huang were partially supported by U.S. NSF-IIS 1302675, NSF-IIS 1344152, NSF-DBI 1356628, NSF-IIS 1619308, NSF-IIS 1633753, NSF-IIS 1633753, NIH R01 AG049371. S.C. Chen was partially supported by the Key Program of NSFC under Grant No. 61732006. X.T. Yuan was supported by NSFC-61522308.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220092

[1] $\phi(\cdot)$ is a transformation function from an input space to a high-dimensional reproducing kernel Hilbert space.

$C^* = 0$, Eq. (1) degenerates to the standard SVM which is convex. For $C^* > 0$, Eq. (1) is non-convex because of the symmetric hinge loss penalizing the unlabeled data inside the margin. Essentially, S^{3}VM solves a non-convex optimization problem which is more difficult than solving a convex problem like standard SVM.

As an important semi-supervised learning model, S^{3}VM has attracted a lot of interest from the data mining and machine learning communities. Multiple learning algorithms were proposed to solve S^{3}VM which can be divided into two groups, *i.e.*, the Difference of Convex (DC) programming based and gradient-based algorithms. Specifically, Fung and Mangasarian [12], Collobert et al. [8], and Wang et al. [27] applied the CCCP to solve S^{3}VM based on the framework of DC programming. Note that, traditional DC programming based algorithms only work in batch learning mode, because these methods have to update the solution via multiple outer loops as shown in Fig. 1(a). Chapelle and Zien [7], Chapelle et al. [4], and Chapelle [3] used smooth optimization to approach the original formulation (1), and then applied gradient descent approach or Newton approach to solve the smooth optimization problems. Because directly solving the problem (1), the CCCP algorithm is the most popular one for solving S^{3}VM. However, all above algorithms are with high computational cost due to batch learning algorithms. To sum up, the high computational cost and non-convexity severely impede the S^{3}VM method in large-scale applications. As pointed out in [6], scaling up S^{3}VM is still an open problem.

Incremental (also called on-line) learning is an important large-scale learning approach [20]. Several incremental learning algorithms have been proposed to solve SVM related models, such as [16, 21, 23]. Essentially these algorithms update the solutions based on the path following algorithms [17, 26] by maintaining the KKT conditions [18]. Besides the path following algorithms, Ertekin et al. [11] proposed an on-line algorithm for a non-convex problem. However, there is no theoretical guarantee for the convergence of the solution. Emara et al. [10] proposed an incremental S^{3}VM algorithm based on branch and bound technique [5]. However, the method of [10] needs to train the model multiple times, which is not efficient. In this paper, we focus on the path following technique due to its efficiency and convergence guarantee.

However, as far as we know, all the existing incremental SVM algorithms based on the path following technique [16, 21, 23] are limited to convex optimization. As discussed previously, S^{3}VM essentially solves a non-convex problem. For non-convex problems, the solution satisfying the KKT conditions can only guarantee to be a saddle point, and cannot guarantee to be a local or global minimum [1]. Thus, it is not trivial to extend the existing path following algorithms [16, 21, 23] to a non-convex problem and guarantee the solution to be a local or global minimum.

To address the above challenging problem, we propose a new efficient incremental learning algorithm for S^{3}VM (named as IL-S^{3}VM) based on the path following technique in the framework of DC programming. As mentioned before, traditional DC programming based algorithms need multiple outer loops and is not suitable for incremental learning, and traditional path following algorithms are limited to convex problems. Our new IL-S^{3}VM algorithm based on the path-following technique can directly update the solution of S^{3}VM to converge to a local minimum within one outer loop (as shown in Fig. 1(b)) so that the efficient incremental learning can be achieved. More importantly, we provide the finite convergence analysis for IL-S^{3}VM. Experimental results on a variety of benchmark datasets not only confirm the finite convergence of IL-S^{3}VM, but also show a huge reduction of computational time compared with existing batch and incremental learning algorithms, while retaining the similar generalization performance.

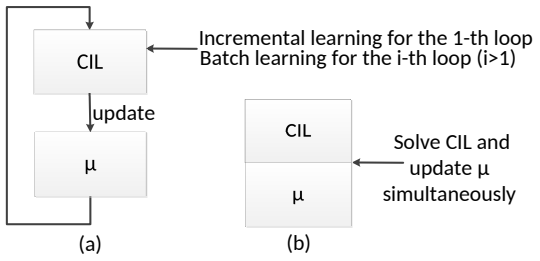


Figure 1: (a) The traditional DC programming based algorithms for S^{3}VM solve the CIL problem and update μ separately. (b) Our new incremental S^{3}VM algorithm solves the CIL problem and updates μ simultaneously.

Contributions. The main contributions of this paper are summarized as follows:

(1) To the best of our knowledge, our IL-S^{3}VM is the first path following algorithm for a non-convex problem (*i.e.*, S^{3}VM) with local minimum convergence guarantee.
(2) Our theoretical results show that the energy function of IL-S^{3}VM monotonically decreases even increases in different conditions, which breaks through the theoretical results of traditional path following algorithms on convex problems, *i.e.*, the energy function only monotonically decreases [14–16].

Organization. The rest of the paper is organized as follows. Section 2 presents the DC formulation of S^{3}VM. In Section 3, we propose our IL-S^{3}VM algorithm. The analysis for IL-S^{3}VM is carried out in Section 4. In Section 5, we show the experimental results. Finally, in Section 6, we conclude the paper.

2 DC FORMULATION OF S^{3}VM

We first briefly review DC programming and its algorithm for S^{3}VM. After that, we provide the KKT conditions for the dual formulation of a convex problem derived from S^{3}VM.

2.1 DC Programming Revisit

As mentioned in [19], the general formulation of DC programming is:

$$\begin{aligned} \min_{w} \quad & o(w) - v(w) \\ s.t. \quad & c_i(w) \le 0, i \in \{1, \cdots, m\}; \quad d_j(w) = 0, j \in \{1, \cdots, p\} \end{aligned} \tag{2}$$

where o, v, and c_j are real-valued convex functions, d_j is an affine function. For the DC programming problem (2), the CCCP algorithm [28] is an extensively used algorithm for finding a local minimal turning point.

Suppose the gradient of $v(w)$ is denoted as $\nabla v(w^{(\ell)})$. The idea of CCCP is to linearize the concave part of (2) around a solution obtained in the current iteration so that $o(w) - w^T \nabla v(w^{(l)})$ is convex

in w. The CCCP algorithm iteratively solves a sequence of the convex program (3) defined by linearizing the concave part, until w converges. In this paper, we call the formulation (3) as convex inner loop (CIL) problem:

$$\begin{aligned} w^{(\ell+1)} \in \arg\min_{w} \quad & o(w) - w^T \nabla v(w^{(\ell)}) \\ s.t. \quad & c_i(w) \le 0, \ \ i \in \{1, \cdots, m\}; \\ & d_j(w) = 0, \ \ j \in \{1, \cdots, p\} \end{aligned} \tag{3}$$

2.2 DC Programming of S^{3}VM

To reformulate the S^{3}VM (1) as the DC programming (2), we double the unlabeled dataset U, and create an artificial labeled dataset $\widetilde{U} = \{(x_{l+1}, +1), \cdots, (x_{l+u}, +1), (x_{l+u+1}, -1), \cdots, (x_{l+2\times u}, -1)\}$. Thus, the original S^{3}VM formulation (1) can be rewritten as (4) according to the DC formulation in (2). It must be pointed out that the problem (4) is definitely equivalent to the original problem (1) as proved in [8].

$$\begin{aligned} \min_{w,b} \ & \underbrace{\frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{l} h_1(y_i f(x_i)) + C^* \sum_{i=l+1}^{l+2\times u} h_1(y_i f(x_i))}_{o(w,b)} \\ & \underbrace{- C^* \sum_{i=l+1}^{l+2\times u} h_0(y_i f(x_i))}_{v(w,b)} \end{aligned} \tag{4}$$

In order to derive the CIL problem for (4), we compute $[w, b]\nabla v(w, b)$ as follows:

$$[w, b]\nabla v(w, b) = -\sum_{i=l+1}^{l+2\times u} \mu_i y_i f(x_i) \tag{5}$$

$$\text{where} \quad \mu_i = \begin{cases} C^* & \text{if } \ y_i f(x_i) < 0, \ \ i \ge l+1 \\ 0 & otherwise \end{cases}$$

Thus, we can obtain the primal CIL problem for (4) based on the formulation (3) which is skipped here. We directly show the corresponding dual CIL problem as follows [8]:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T H\alpha - y^T\alpha \\ s.t. \quad & \sum_{i=1}^{l+2\times u} \alpha_i = 0; \quad \underline{C}_i \le \alpha_i \le \overline{C}_i, \ i = 1, \cdots, l+2\times u \end{aligned} \tag{6}$$

where H is a positive semidefinite matrix with $H_{ij} = K(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle$ for all $1 \le i, j \le l + 2 \times u$, $K(x_i, x_j)$ is the kernel function, and $\underline{C}_i$ and $\overline{C}_i$ are defined as follows:

$$\underline{C}_i = \begin{cases} -\mu_i & \text{if } \ y_i = +1 \\ \mu_i - C & \text{if } \ y_i = -1, i = 1, \cdots, l \\ \mu_i - C^* & \text{if } \ y_i = -1, i \ge l+1 \end{cases}; \tag{7}$$

$$\overline{C}_i = \begin{cases} C - \mu_i & \text{if } \ y_i = +1, i = 1, \cdots, l \\ -\mu_i & \text{if } \ y_i = +1, i \ge l+1 \\ \mu_i & \text{if } \ y_i = -1 \end{cases} \tag{8}$$

Remark 1. *For an unlabeled sample x_{l+i} with the corresponding artificial labeled samples $(x_{l+i}, +1)$ and $(x_{l+u+i}, -1)$. According to (5) and (7)-(8), we have that $\left[\underline{C}_{l+i}, \overline{C}_{l+i}\right] = [-C^*, 0]$ and $\left[\underline{C}_{l+u+i}, \overline{C}_{l+u+i}\right] = [-C^*, 0]$, if $f(x_i) < 0$. Correspondingly, if $f(x_i) > 0$, we have that $\left[\underline{C}_{l+i}, \overline{C}_{l+i}\right] = [0, C^*]$ and $\left[\underline{C}_{l+u+i}, \overline{C}_{l+u+i}\right] = [0, C^*]$.*

2.3 KKT Conditions for Dual CIL Problem of S^{3}VM

According to the convex optimization theory [1], the solution of the dual CIL problem (6) can be obtained by the following min-max problem:

$$\min_{\underline{C} \le \alpha \le \overline{C}} \max_{b' \in \mathbb{R}} \quad \mathcal{W} = \frac{1}{2}\alpha^T H\alpha - y^T\alpha + b'\left(\sum_{i=1}^{l+2\times u} \alpha_i\right) \tag{9}$$

where b' is the Lagrangian multiplier. Further, from the KKT theorem [18], the first-order derivative of $\mathcal{W}$ leads to the following KKT conditions:

$$\frac{\partial \mathcal{W}}{\partial b'} = \sum_{i=1}^{l+2\times u} \alpha_i = 0 \tag{10}$$

$$g_i \stackrel{\text{def}}{=} \frac{\partial \mathcal{W}}{\partial \alpha_i} = \sum_{j=1}^{l+2\times u} \alpha_j H_{ij} + b' - y_i \tag{11}$$

$$\begin{cases} > 0 & \text{then} \quad \alpha_i = \underline{C}_i \\ = 0 & \text{then} \quad \underline{C}_i \le \alpha_i \le \overline{C}_i \\ < 0 & \text{then} \quad \alpha_i = \overline{C}_i \end{cases}$$

Thus, according to the value of g_i in Eq. (11), the extended training sample set $L \cup \widetilde{U}$ can be partitioned as $\pi = (\mathcal{M}, \mathcal{E}, \mathcal{O})$, where

(1) $\mathcal{M} = \{i \in L \cup \widetilde{U} : \ g_i = 0, \ \underline{C}_i \le \alpha_i \le \overline{C}_i\}$,
(2) $\mathcal{E} = \{i \in L \cup \widetilde{U} : \ g_i < 0, \ \alpha_i = \overline{C}_i\}$,
(3) and $\mathcal{O} = \{i \in L \cup \widetilde{U} : \ g_i > 0, \ \alpha_i = \underline{C}_i\}$.

3 NEW INCREMENTAL LEARNING ALGORITHM

In this section, we first present the principle of our incremental S^{3}VM algorithm (*i.e.*, IL-S^{3}VM), then give the details of IL-S^{3}VM algorithm. Finally, we discuss the difference between our IL-S^{3}VM and the existing incremental SVM algorithms.

3.1 Principle of our IL-S^{3}VM Algorithm

Multiple DC programming based algorithms have been proposed to solve S^{3}VM, but these methods can only do batch learning. Specifically, when these algorithms are applied to solve S^{3}VM, according to the CCCP algorithm discussed before, the algorithms generally solve the CIL problem and update μ individually, *i.e.*, the algorithms need multiple outer loops until μ converges (see Fig. 1(a)). We can use incremental learning algorithms for the 1-th loop. However, for the i-th loop with $i > 1$, only batch solvers can be used, which is known to be inefficient for on-line scenario. Therefore, it is highly desirable to design an incremental learning algorithm with solving the CIL problem and updating μ simultaneously (see Fig. 1(b)). Note that, although the method of [11] was designed to handle on-line learning for a non-convex problem, they fixed μ during the whole adjustments. Thus, there is no theoretical guarantee for the convergence of the solution to a local minimum.

In this paper, we will solve the CIL problem and update μ simultaneously while guaranteeing the convergence of the solution to a local minimum. During the incremental learning for S^{3}VM, the values of μ_i may change from 0 to C^* or from C^* to 0 as discussed in Section 2.2, which leads the change of the dual CIL problem (6) and makes the corresponding samples violate the KKT conditions. To handle this complexity, we define a KKT-violating set $\mathcal{A}$ in Definition 1.

Definition 1 (KKT-violating set $\mathcal{A}$). *The KKT-violating set $\mathcal{A}$ is defined as a subset of an union of $\widetilde{U}$ and an added sample (x_c, y_c), such that all the samples violating the KKT conditions are included in $\mathcal{A}$.*

Our fundamental principle for IL-S^{3}VM is to *detect the new samples violating the KKT conditions and add these samples into the KKT-violating set $\mathcal{A}$, while pushing the samples in $\mathcal{A}$ to satisfy the KKT conditions* (see Fig. 2). Note that, during the adjustments, the sample in $O \cup \mathcal{E}$ moving into $\mathcal{A}$ (drawn by blue arrow lines in Fig. 2) corresponds to the change of the value of μ_i in (5), which is unique in IL-S^{3}VM and totally different to the existing incremental SVM algorithms.

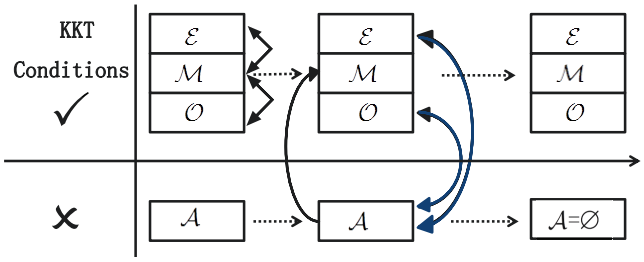


Figure 2: The fundamental principle of IL-S^{3}VM. The movements drawn by blue arrow lines are different to the existing incremental SVM algorithms.

3.2 IL-S^{3}VM Algorithm

When a labeled sample (x_{new}, y_{new}) or an unlabeled sample x_{new} is added into the labeled dataset L or the unlabeled dataset $\widetilde{U}$, there correspondingly exist additions in L or $\widetilde{U}$. We define the additions in L and $\widetilde{U}$ as L_{new} and $\widetilde{U}_{new}$ respectively. We use $\Delta\alpha_{\mathcal{A}}$ to represent the changes of the weights of $\mathcal{A}$, and set the direction to $\Delta\alpha_{\mathcal{A}}$ as $d_{\mathcal{A}} = \widetilde{C}_{\mathcal{A}} - \alpha_{\mathcal{A}}$, where $\widetilde{C}_i = \overline{C}_i$ if $y_i = +1$, otherwise $\widetilde{C}_i = \underline{C}_i$. Thus, we have $\Delta\alpha_{\mathcal{A}} = \eta \cdot d_{\mathcal{A}}$, where η is a parameter with $0 \le \eta \le 1$ to control the adjustment qualities of $\alpha_{\mathcal{A}}$. Correspondingly, we use $\Delta\alpha$ to represent the changes of the weights of samples in $L \cup \widetilde{U} - \mathcal{A}$. Thus, the following two issues need to be addressed for designing the incremental S^{3}VM algorithm:

i What is the direction of $\Delta\alpha$ with respect to η, with keeping all the samples in $\mathcal{M}$, $\mathcal{E}$ and O satisfying the KKT conditions during the adjustment?

ii What is the maximum adjustment quantity η^{max} of η, such that if η exceeds $\Delta\eta^{max}$: **1**) a sample migrates among the sets $\mathcal{M}$, $\mathcal{E}$ and O; **2**) the KKT conditions for one sample in $\mathcal{A}$ will be satisfied; **3**) or the samples in $O \cup \mathcal{E}$ violate the KKT conditions, *i.e.*, we need to update the values of μ_i in Eq. (5)?

After finding the maximum adjustment quantity η^{max}, we can update the α_c, α, b', $\underline{C}$, $\overline{C}$, $\mathcal{A}$, $\mathcal{M}$, $\mathcal{E}$, and O correspondingly. Repeating the above procedures until the set $\mathcal{A}$ becomes an empty set derives our IL-S^{3}VM algorithm, which is summarized in Alg. 1. In the following, we provide the detailed descriptions for addressing the above two issues. The detailed procedure for updating $\alpha_{\mathcal{A}}$, α, b', g, $\underline{C}$, $\overline{C}$, $\mathcal{A}$, $\mathcal{M}$, $\mathcal{E}$ and O can be found in our Appendix at http://www.pitt.edu/%7eheh45/appendix.pdf.

Algorithm 1 IL-S^{3}VM

Input: α, b', $\underline{C}$, $\overline{C}$, $\mathcal{M}$, $\mathcal{E}$, O and $L_{new} \cup \widetilde{U}_{new}$.
Output: α, b', $\underline{C}$, $\overline{C}$, $\mathcal{M}$, $\mathcal{E}$ and O.

```
while L_new ∪ Ũ_new ≠ ∅ do
    Read a new sample (x_c, y_c) from L_new ∪ Ũ_new.
    Remove (x_c, y_c) from L_new ∪ Ũ_new.
    Initialize α_c = 0 and compute C_c, C̄_c and g_c.
    Add (x_c, y_c) into M, E, O or A according to g_c.
    while A ≠ ∅ do
        Compute β_b, and β_M, o and γ (see Section 3.2.1).
        Compute the maximal quantity η^max (see Section 3.2.2).
        Update α_A, α, b', g, C, C̄, A, M, E and O.
    end while
end while
```

3.2.1 Compute the Direction of $\Delta\alpha$. As mentioned before, one solution of (6) corresponds a partition π. Conversely, given a partition π and a KKT-violating set $\mathcal{A}$, we can define a set of parameter η which shares the given partition π and the given $\mathcal{A}$. We give a formal definition of this set of parameter η in Definition 2.

Definition 2. *Given a partition π and a KKT-violating set $\mathcal{A}$, we define $\mathcal{I}(\pi, \mathcal{A}) = \{\eta \mid \pi(\eta) = \pi(0), \mathcal{A}(\eta) = \mathcal{A}(0)\}$[2], such that $\mathcal{I}(\pi, \mathcal{A})$ shares the given partition π and the given KKT-violating set $\mathcal{A}$.*

Theorem 1 shows that $\mathcal{I}(\pi, \mathcal{A})$ is a closed interval such that $\mathcal{I}(\pi, \mathcal{A}) = [0, \eta^{max}]$. Further, α and b' are linear w.r.t. η in the interval $\mathcal{I}(\pi, \mathcal{A})$. The proof to Theorem 1 can be found in our Appendix at http://www.pitt.edu/%7eheh45/appendix.pdf.

Theorem 1. *Given a partition π and a KKT-violating set $\mathcal{A}$. We have that $\mathcal{I}(\pi, \mathcal{A})$ is a closed interval as $\mathcal{I}(\pi, \mathcal{A}) = [0, \eta^{max}]$. $\begin{bmatrix} b' \\ \alpha \end{bmatrix}$ is linear w.r.t. η in the interval $\mathcal{I}(\pi, \mathcal{A})$ with the directions $\beta = \begin{bmatrix} \beta_{b'} \\ \beta_{\mathcal{M}} \\ \beta_{O\cup\mathcal{E}} \end{bmatrix}$, where $\beta_{O\cup\mathcal{E}} = \mathbf{0}$, and $\begin{bmatrix} \beta_{b'} \\ \beta_{\mathcal{M}} \end{bmatrix}$ is obtained by solving the following linear system*

$$\underbrace{\begin{bmatrix} 0 & \mathbf{1}_{\mathcal{M}}^T \\ \mathbf{1}_{\mathcal{M}} & H_{\mathcal{M}\mathcal{M}} \end{bmatrix}}_{\widetilde{H}} \begin{bmatrix} \beta_{b'} \\ \beta_{\mathcal{M}} \end{bmatrix} = - \begin{bmatrix} \mathbf{1}_{\mathcal{A}}^T \\ H_{\mathcal{M}\mathcal{A}} \end{bmatrix} d_{\mathcal{A}} \tag{12}$$

[2]In this paper, $\pi(0)$ and $\mathcal{A}(0)$ defaultly denote the given partition π and the given KKT-violating set $\mathcal{A}$, respectively.

Remark 2. *According to Theorem 1, we know the direction of $\Delta\alpha$ for all $\eta \in \mathcal{I}(\pi, \mathcal{A})$, which can be obtained by solving Eq. (12). The Traditional way for solving Eq. (12) is the direct matrix inverse of $\widetilde{H}$ [17, 26]. As mentioned in [15, 22], the training samples may be linearly dependent in the reproducing kernel Hilbert space, which make the key matrix $\widetilde{H}$ singular. Thus, we cannot solve Eq. (12) by directly computing the inverse of the matrix $\widetilde{H}$. To make IL-S^{3}VM run in a more robust way, we compute $\beta_{b'}$ and $\beta_{\mathcal{M}}$ in Eq. (12) based on the QR decomposition with column pivoting [24], without directly computing the inverse of $\widetilde{H}$.*

After obtaining the direction of $\Delta\alpha$, we can further obtain the linear relationship between Δg_i and η as shown in Corollary 1.

Corollary 1. *Given a direction of $\Delta\alpha$ as β, the linear relationship between Δg_i ($\forall i \in L \cup \widetilde{U} \cup \mathcal{A}$) and η can be obtained as:*

$$\gamma_i \stackrel{\text{def}}{=} \sum_{j \in \mathcal{A}} H_{ij} d_j + \sum_{j \in \mathcal{M}} H_{ij} \beta_j + \beta_{b'} \tag{13}$$

Remark 3. *According to Eq. (11), we have $\gamma_i = 0$ for all $i \in \mathcal{M}$. For an unlabeled sample x_{l+i} with the corresponding artificial labeled samples $(x_{l+i}, +1)$ and $(x_{l+u+i}, -1)$, we have $\gamma_{l+i} = \gamma_{l+u+i}$ according to Eq. (13). In addition, d_{l+i} and d_{l+u+i} have the same sign.*

3.2.2 Compute the Maximal Quantity η^{max}. As mentioned in Theorem 1, given a partition π and a KKT-violating set $\mathcal{A}$, we have that $\mathcal{I}(\pi, \mathcal{A})$ is a closed interval as $\mathcal{I}(\pi, \mathcal{A}) = [0, \eta^{max}]$, where η^{max} is the maximal quantity of η. Corollary 2 answers the value of the maximum adjustment η^{max}. Corollary 2 provides a theoretical result to compute η^{max}. We provide a detail procedure to compute η^{max} in our Appendix at http://www.pitt.edu/%7eheh45/appendix.pdf.

Corollary 2. *The maximum adjustment quantity η^{max} can be obtained by solving the following system of linear inequalities.*

$$\begin{aligned}
\underline{C}_i \le \alpha_i + \beta_i \eta &\le \overline{C}_i, \ \forall i \in \mathcal{M} && (14)\\
g_i + \gamma_i \eta &> 0, \ \forall i \in \mathcal{O} && (15)\\
g_i + \gamma_i \eta &< 0, \ \forall i \in \mathcal{E} && (16)\\
y_i g_i + 1 + y_i \gamma_i \eta &\ge 0, \ \forall i \in \widetilde{U} \& \mu_i = 0 && (17)\\
y_i g_i + 1 + y_i \gamma_i \eta &< 0, \ \forall i \in \widetilde{U} \& \mu_i = C^* && (18)\\
\alpha_i + \beta_i \eta &\le \overline{C}_i, \ \forall i \in \mathcal{A} \& y_i = +1 && (19)\\
\alpha_i + \beta_i \eta &\ge \underline{C}_i, \ \forall i \in \mathcal{A} \& y_i = -1 && (20)\\
g_i + \gamma_i \eta &\ge 0, \ \forall i \in \mathcal{A} \& y_i = +1 && (21)\\
g_i + \gamma_i \eta &\le 0, \ \forall i \in \mathcal{A} \& y_i = -1 && (22)
\end{aligned}$$

Remark 4. *The system of linear inequalities (14)-(22) gives an interval of η. The right endpoint of this interval is the value of η^{max}. Specifically, there are three cases for computing the maximum adjustment quantity η^{max}: **1)** Eqs. (14)-(16) correspond that a sample migrates among the sets $\mathcal{M}$, $\mathcal{E}$ and $\mathcal{O}$ (shown in the 2nd column of Fig. 2). **2)** Eqs. (18)-(19) correspond that the KKT conditions for one sample in $\mathcal{A}$ will be satisfied (shown by the black arrow lines in the 3rd column of Fig. 2). **3)** Eqs. (20)-(22) correspond that the samples in $\mathcal{O} \cup \mathcal{E}$ violate the KKT conditions,* i.e., *we need to update the values of μ_i in Eq. (5) (drawn by blue arrow lines in the 3rd column of Fig. 2).*

3.3 Differences Compared with existing algorithms

The main differences between our IL-S^{3}VM and the existing incremental SVM algorithms [16, 21, 23] are summarized as follows.

(1) The key difference is that our IL-S^{3}VM can handle a non-convex problem with a convergence guarantee, however, the existing incremental SVM algorithms [16, 21, 23] cannot. Our IL-S^{3}VM is a big step forward compared with the existing incremental SVM algorithms.
(2) Technically, during the adjustments of the existing incremental SVM algorithms, the samples can only migrate from $\mathcal{A}$ into $\mathcal{M} \cup \mathcal{E} \cup \mathcal{O}$ (see the black arrow lines in the 3rd column of Fig. 2). However, we define a new set $\mathcal{A}$ for all samples violating the KKT conditions, which is the key to handle the non-convexity in S^{3}VM. For IL-S^{3}VM, the samples can migrate bidirectionally between $\mathcal{A}$ and $\mathcal{M} \cup \mathcal{E} \cup \mathcal{O}$ (see the 3rd column in Fig. 2).
(3) Theoretically, the energy function of existing incremental SVM algorithms is monotonically decreasing [14–16]. However, the energy function of our IL-S^{3}VM is monotonically decreasing even increasing in different conditions (cf. Theorem 3), which will be discussed in detail below.

4 ANALYSIS AND DISUCSSION

In this section, we first prove the finite convergence of IL-S^{3}VM, and then provide the time complexity analysis of IL-S^{3}VM.

4.1 Finite Convergence of IL-S^{3}VM

In this section, we prove the finite convergence of IL-S^{3}VM, which guarantees that IL-S^{3}VM converges to a local minimum in a finite number of iterations. All the proofs can be found in our Appendix at http://www.pitt.edu/%7eheh45/appendix.pdf. Existing theoretical results related to the path following algorithms focus on convex problems [14–16]. To the best of our knowledge, this is the first theoretical result for the path following algorithm on a non-convex problem (*i.e.*, S^{3}VM) with local minimum convergence guarantee.

Before proving the finite convergence of IL-S^{3}VM, we first prove that a sample cannot migrate back and forth in successive adjustment steps among the sets $\mathcal{A}$, $\mathcal{M}$, $\mathcal{E}$, and $\mathcal{O}$ in Theorem 2.

Theorem 2. *During the adjustments of IL-S^{3}VM, a sample cannot migrate back and forth in successive adjustment steps among the sets $\mathcal{A}$, $\mathcal{M}$, $\mathcal{E}$, and $\mathcal{O}$.*

To prove Theorem 2, we prove the following four sub-conclusions: 1) if a sample (x_t, y_t) is added into the set $\mathcal{M}$, (x_t, y_t) will not be removed from $\mathcal{M}$ in the immediate next adjustment. 2) If (x_t, y_t) is removed from the set $\mathcal{M}$, then (x_t, y_t) will not be added into $\mathcal{M}$ in the immediate next adjustment. 3) If (x_t, y_t) is removed from the set $\mathcal{E}$ or $\mathcal{O}$ and added into the set $\mathcal{A}$, then (x_t, y_t) will not be removed from $\mathcal{A}$ in the immediate next adjustment. 4) If (x_t, y_t) is removed from the set $\mathcal{A}$ and added into the set $\mathcal{E}$, $\mathcal{M}$ or $\mathcal{O}$, then (x_t, y_t) will not be removed from $\mathcal{E}$, $\mathcal{M}$ or $\mathcal{O}$, in the immediate next adjustment. Based on Theorem 2, we have that $\eta^{max} > 0$ for each adjustment (Corollary 3).

Corollary 3. *For each adjustment of IL-S^{3}VM, the maximum adjustment η^{max} is greater than zero.*

Based on Corollary 3, we prove that the function $\mathcal{W}$ is strictly monotonically decreasing and increasing under different conditions (Theorem 3).

Theorem 3. *During the adjustments of IL-S^{3}VM, the the function $\mathcal{W}$ has the following properties:*

(1) If $\mathcal{A}$ only includes the added sample, i.e., *$\mathcal{A} = \{(x_c, y_c)\}$, $\mathcal{W}$ is strictly monotonically decreasing.*

(2) If $\mathcal{A}$ dos not include the added sample (x_c, y_c) and $\mathcal{A} \neq \emptyset$, $\mathcal{W}$ is strictly monotonically increasing.

Finally, based on Theorem 3 and Corollary 3, we can prove the finite convergence of IL-S^{3}VM in Theorem 4.

Theorem 4. *IL-S^{3}VM converges to a local minimal of (4) in a finite number of steps.*

4.2 Time Complexity Analysis

As mentioned in Alg. 1, for each iteration, we need to solve the system of equations (12) of size $|\mathcal{M}| + 1$ to compute $\beta_{b'}$ and $\beta_{\mathcal{M}}$, which mainly involves an QR decomposition. Because the changed in $\widetilde{H}$ is slight (either adding or deleting a row and a column from $\widetilde{H}$), the QR decomposition can be updated without recomputing the QR decomposition from scratch [13]. The computational complexity of the updating is $O\left(|\mathcal{M}|^2\right)$. According to Eq. (13), the computation of γ_i requires $O\left(|\mathcal{M}|(l + 2 \times u)\right)$ computations. According to (14)-(22), the computation of η^{max} requires $O\left(l + 2 \times u\right)$ computations. The updates of $\alpha_{\mathcal{A}}$, α, b', g, $\underline{C}$, $\overline{C}$, $\mathcal{A}$, $\mathcal{M}$, $\mathcal{E}$ and $\boldsymbol{O}$ require $O\left(l + 2 \times u\right)$ computations. Thus, the time complexity of each iteration is $O\left(|\mathcal{M}|(l + 2 \times u) + |\mathcal{M}|^2\right)$.

As shown in Theorem 4, IL-S^{3}VM converges to a local minimal of (4) in a finite number of steps. Our experience shows that the number of iterations of IL-S^{3}VM is some small number of the extended sample size $l + 2 \times u$. Thus, the computational complexity of IL-S^{3}VM is $O\left(|\mathcal{M}|(l + 2 \times u)^2 + |\mathcal{M}|^2(l + 2 \times u)\right)$ which is much cheap compared with the one of batch S^{3}VM algorithm [8].

5 EXPERIMENTS

In this section, we first present the experimental setup, and then provide the experimental results and discussions.

5.1 Experimental Setup

Design of Experiments: In the experiments, we first verify the effectiveness of our IL-S^{3}VM, and then demonstrate the advantage of our IL-S^{3}VM for large-scale semi-supervised learning.

To show the effectiveness of our IL-S^{3}VM, we investigate the finite convergence of IL-S^{3}VM. Specifically, we count the numbers of the iterations (denoted as *Iterations*), and the changing of the values of μ_i (denoted as *Changing*$-\mu$) in IL-S^{3}VM. By counting the *Changing*$-\mu$, we want to verify that the changing of the values of μ_i can be handled by our IL-S^{3}VM. By counting the *Iterations*, we show that IL-S^{3}VM can converge to a local minimum in a finite number of iterations.

To show the advantage of IL-S^{3}VM for large-scale semi-supervised learning, we compare the running time of our IL-S^{3}VM with other batch and incremental S^{3}VM algorithms. Specifically, the compared algorithms are summarized as follows.

(1) BL-S^{3}VM (also called UniverSVM [25]): the state-of-the art batch S^{3}VM algorithm based on the CCCP algorithm [8] and SMO algorithm [2].
(2) NN-BB: an incremental S^{3}VM algorithm based on branch and bound technique [10].
(3) IL-S^{3}VM: Our proposed incremental S^{3}VM algorithm.

In addition, we compare the unlabeled errors (*i.e.*, training errors on the unlabeled samples) of different algorithms.

Implementation: We implement our IL-S^{3}VM in MATLAB. Sinz and Roffilli [25] implemented BL-S^{3}VM based on CCCP algorithm in C/C++. To compare the run-time at the same platform, we implement BL-S^{3}VM in MATLAB. We implement NN-BB in MATLAB. For kernels, the linear kernel, polynomial kernel $K(x_1, x_2) = (x_1 \cdot x_2 + 1)^d$ with $d = 2$ and Gaussian kernel $K(x_1, x_2) = \exp(-\kappa\|x_1 - x_2\|^2)$ with $\kappa = 0.1$ are used in all the experiments. The parameters C and C^* are fixed at 10 and 5 respectively.

For the experiments of verifying the effectiveness of IL-S^{3}VM, we add a label or unlabeled sample into the original training dataset. For the experiments of comparing the running time of different algorithms, we add 20 label or unlabeled samples into the original training dataset. BL-S^{3}VM need recompute a solution from scratch. NN-BB and our IL-S^{3}VM can update the current solution to incorporate the 20 new (labeled or unlabeled) samples.

Datasets: Table 1 summarizes the eight benchmark datasets (*i.e.*, CodRNA, Madelon, IJCNN1, Text, Usps, W6a, A9a and Mushrooms datasets) used in the experiments. They are from LIBSVM [3] and Olivier [4] sources. Originally, the Usps dataset has ten classes from 0 to 9. We created a binary version of Usps dataset by classifying digits 0 to 4 versus 5 to 9. Originally, these datasets are used for supervised learning. To conduct the experiments of semi-supervised learning, we transfer these fully labeled datasets to the partially labeled datasets, by randomly dropping the labels of a part of samples out. The numbers of unlabeled samples are listed in the column "Unlabeled" of Table 1.

Table 1: The real-world dasetsets used in the experiments.

Dataset	Dimensionality	Samples	Unlabeled	Source
CodRNA	8	59,535	59,235	LIBSVM
Madelon	500	2,000	1,800	LIBSVM
IJCNN1	22	49,990	49,790	LIBSVM
Text	7,511	1,946	1,800	Olivier
Usps	256	2,007	1,800	LIBSVM
W6a	300	17,188	17,000	LIBSVM
A9a	123	32,561	32,200	LIBSVM
Mushrooms	112	8,124	7,900	LIBSVM

5.2 Results and Discussions

Table 2 presents the average and standard deviation of the numbers of **Iterations** for running IL-S^{3}VM by adding a labeled sample over 20 trials. Table 3 presents the average and standard deviation of the numbers of **Iterations** for running IL-S^{3}VM by an unlabeled sample over 20 trials. The results show that **IL-S^{3}VM converges to**

[3] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html.
[4] http://olivier.chapelle.cc/lds/.

Table 2: Average results with the standard deviation of IL-S^{3}VM (adding a labeled sample) over 20 trials, where linear, polynomial and Gaussian kernels were used.

Dataset	Size	*Iterations*			*Changing*$-\mu$		
		Linear	Polynomial	Gaussian	Linear	Polynomial	Gaussian
CodRNA	10,000	12.8±10.3	27.1± 17.9	24.4±12.5	12.6±10.0	24.9±17.9	42.6±35.6
	20,000	21.6±13.3	36.8±26.0	21.8±15.3	21.3±13.0	36.5±25.8	21.5±15.0
	30,000	15.6±12.2	36.0±26.6	27.6±18.3	15.6±12.0	36.8±26.4	27.4±18.1
	40,000	24.9±17.3	27±15.9	30.3±16.3	24.8±17.3	24.8±15.5	30.0±16.0
Madelon	400	2.4±5.2	0.6±0.7	0.6±0.9	2.1±4.8	0.1±0.3	0.3±0.5
	800	5.6±12.7	0.7±0.6	0.6±1.0	5.2±12.4	0.1±0.3	0.3±0.5
	1,200	42.5±68.2	0.6±0.8	0.3±0.7	42±67.9	0.2±0.4	0.2±0.4
	1,600	71.3±177.3	1.4±1.7	0.2±0.7	71.1±176.9	0.7±1.5	0.1±0.3
IJCNN1	10,000	54.8±51.6	16.6±25.1	4.8±11.5	54.3±51.1	16.2±24.8	4.5±11.2
	20,000	53.6±67.4	63.0±97.8	6.8±12.8	53.2±66.9	62.6±97.8	6.4±12.5
	30,000	74.8±69.9	51.8±119.1	13.9±15.1	74.2±69.4	51.5±118.9	13.3±14.8
	40,000	85.16±75.5	81.6±89.6	17.1±35.8	81.6±75.0	80.2±89.4	16.6±35.6
Text	400	1.0±1.0	0.8±1.0	0.5±0.9	0.4±0.7	0.4±0.5	0.3±0.4
	800	2.6±3.9	0.8±1.0	0.6±0.9	2.2±3.5	0.4±0.5	0.3±0.5
	1,200	4.6±5.6	0.9±1.5	0.8±1.0	4±5.2	0.5±1.1	0.4±0.5
	1,600	25.1±38.4	1.3±1.8	0.8±1.0	24.5±38.1	0.8±1.4	0.4±0.5
Usps	400	4.2±7.3	1.2±2.9	0.7±0.6	3.8±6.9	1.0±2.5	0.1±0.3
	800	22.4±35.6	7.5±16.4	0.6±0.6	22.2±35.5	7.2±16.1	0.0±0.2
	1,200	39.4±56.0	8.8±14.7	0.4±0.5	40.5±54.0	8.4±14.3	0±0
	1,600	30.0±55.6	11.4±16.1	0.6±0.9	30.0±55.5	11.0±15.8	0.1±0.6
W6a	4,000	20±24	28±12	11±14	18±24	26±28	7±11
	8,000	45±40.2	24±12.8	18±3.7	84±40.2	38±32.4	15±23
	12,000	56±48	48±30	14±22	65±58	45±38	10±19
	16,000	57±43	72±51	34±23	37±64	40±21	22±18
A9a	6,000	33.2±29.5	9.3±14.2	3.3±6.5	26.3±25.3	11.1±16.2	3.2±5.2
	12,000	28.1±34	32.0±41.3	13.2±11.3	31.1±36.1	28.4±42.3	5.1±7.5
	18,000	34.3±28.3	21.3±39.3	7.8±8.2	34.4±31.1	26.5±49	11.3±13.8
	24,000	45.6±35.7	45.2±43.4	9.3±17.2	41.2±45.0	40.1±41.4	9.6±21.3
Mushrooms	1,500	3.1±4.7	4.7±1.2	2.4±1.6	2.3±1.8	1.3±1.6	1.3±1.5
	3,000	5.2±7.2	3.2±3.2	1.9±2.3	6.2±7.1	3.4±2.5	2.1±1.7
	4,500	13.4±15.6	6.9±5.5	4.1±5.0	13±15.2	5.2±4.1	3.6±4.3
	6,000	55.1±68.2	17.2±16.3	5.2±4.0	44.6±57.2	7.2±4.4	4.7±3.5

a local minimal in a finite number of iterations. Tables 2 and 3 also present the average and standard deviation of the numbers of *Changing*$-\mu$ for running IL-S^{3}VM over 20 trials. The results verify that the changing of the values of μ_i indeed happens in IL-S^{3}VM with a high probability. **Our IL-S^{3}VM can effectively handle *Changing*$-\mu$, and guarantee to converge to a local minimal in a finite number of iterations**.

Fig. 3 presents the running time (in seconds) of BL-S^{3}VM, NN-BB and IL-S^{3}VM. In the notation $(\cdot)$, the abbreviations L, P and G stand for the linear, polynomial and Gaussian kernels respectively. The results clearly demonstrate that **our IL-S^{3}VM is much faster than NN-BB and BL-S^{3}VM**. This is because, BL-S^{3}VM need rebuild the solution of S^{3}VM from scratch by solving multiple DC programming problems when new samples are added. NN-BB also need solve multiple SVMs even incremental learning strategy is used. However, our IL-S^{3}VM can directly and efficiently update the solution to converge to a local minimal.

Figure 4 shows the unlabeled errors over 10 trials with notched box plot, when adding 20 labeled and unlabeled samples into the original dataset and using the Gaussian kernel. The results show that NN-BB, BL-S^{3}VM and IL-S^{3}VM have the similar accuracies on the unlabeled dataset. Especially, IL-S^{3}VM and BL-S^{3}VM achieve the same accuracy at most cases. These results confirm the superiority of our IL-S^{3}VM, because our IL-S^{3}VM is much faster than existing batch and incremental learning algorithms, while retaining the similar generalization performance as discussed above.

6 CONCLUSION

In this paper, we propose a new incremental learning algorithm to scale up S^{3}VM (IL-S^{3}VM) based on the path following algorithm in the framework of DC programming. The traditional DC programming based algorithms need multiple outer loops and is not suitable for incremental learning, and traditional path following algorithms only work for convex problems. Our new IL-S^{3}VM algorithm based

Table 3: Average results with the standard deviation of IL-S^{3}VM (adding an unlabeled sample) over 20 trials, where linear, polynomial and Gaussian kernels were used.

Dataset	Size	*Iterations*			*Changing–μ*		
		Linear	Polynomial	Gaussian	Linear	Polynomial	Gaussian
CodRNA	10,000	15.8±10.3	24.3±15.4	15.5±10.8	15.6±10.0	24.1±15.2	15.2±10.6
	20,000	15.2±13.0	6.3±6.1	15.3±9.4	15.1±13.0	6.2±5.8	12.9±9.1
	30,000	6.5±8.6	18.6±14.3	9.6±6.7	6.4±8.3	18.5±14.3	9.3±6.3
	40,000	6.2±8.2	6.0±11.3	18.7±11.4	6.1±7.9	9.0±11.3	16.4±11.2
Madelon	400	4.8±4.9	1.0±0	0.6±0.5	4.2±4.5	0±0	0±0
	800	3.5±6.4	1.2±0.7	0.3±0.5	3.2±6.0	0.2±0.7	0±0
	1,200	1.7±6.0	1.1±0.5	0.6±0.5	1.6±5.7	0.2±0.5	0±0
	1,600	5±12.3	1.4±0.9	0.3±0.5	4.8±12.0	1.4±0.9	0±0
IJCNN1	10,000	43.6±48.3	12.5±17.1	2.6±4.3	43.2±47.9	12.1±16.7	2.1±4.0
	20,000	46±55.0	27.7±33.4	3.5±5.8	45.6±54.5	27.2±33.1	3.0±5.5
	30,000	73.1±79.7	39.6±81.4	4.6±6.8	72.6±79.2	39.4±81.1	4.0±6.5
	40,000	58.6±72.8	45.4±84.0	5.8±5.1	58.2±72.3	45.1±83.6	7.4±4.8
Text	400	1.4±0.8	1.0±0	1.0±0.4	0.4±0.8	0±0	0.1±0.2
	800	3.5±4.1	1.0±0	0.8±0.4	2.5±4.1	0±0	0±0
	1,200	5.5±5.9	1.0±0	0.9±0.3	4.6±5.8	0±0	0±0
	1,600	4.1±4.2	1.1±0.3	0.9±0.3	3.3±4.0	0.1±0.3	0±0
Usps	400	3±5.3	2.1±4.3	1.1±1.3	2.7±4.9	1.8±3.9	0.2±1.2
	800	2.4±6.3	1.6±4.7	1.2±1.7	2.2±6.0	1.4±4.5	0.3±1.6
	1,200	7.7±13.8	4.5 ±6.5	1.0±0.2	7.4±13.5	3.9±6.2	0.0±0.2
	1,600	8.4±21.0	3.9±11.5	1.0±0.9	8.3±20.9	3.6±11.2	0.2±0.8
W6a	4,000	8±23	18±16	15±18	7±12	14±23	11±14
	8,000	26±15	17±19	24±33	15±21	15±25	19±23
	12,000	42±41	53±35	24±31	21±23	35±39	12±23
	16,000	19±26	14±28	25±32	18±22	23±25	22±28
A9a	6,000	21.3±23.3	7.6±9.2	2.1±4.1	23.1±25.2	6.5±8.2	1.3±2.1
	12,000	24±28.0	15.2±17.3	2.8±3.9	28.3±28.3	17.1±16.3	2.4±4.1
	18,000	36.2±39.3	21.2±41.1	3.5±5.7	28.6±33.1	18.4±31.2	4.1±5.1
	24,000	31.3±35.8	25.1±34.0	4.8±4.3	28.2±42	21.2±34.5	4.3±3.4
Mushrooms	1,500	5.2±3.3	3.2±1.4	3.1±1.4	1.2±1.5	1.0±0.8	0.4±0.6
	3,000	7.5±6.3	3.1±1.4	2.2±1.2	6.1±4.1	1.4±1.0	0.8±1.0
	4,500	15.1±12.9	5.1±2.1	3.4±2.3	12.4±15.3	3.6±2.6	2.3±1.2
	6,000	18.2±14.2	5.2±6.2	3.1±1.3	12.3±19.0	8.2±4.3	5.3±4.2

on the path-following algorithm can directly update the solution to converge to a local minimal within one outer loop so that the incremental learning can be achieved. More importantly, we provide the finite convergence analysis for our new algorithm. To the best of our knowledge, our new IL-S^{3}VM algorithm is the first efficient path following algorithm for a non-convex problem (*i.e.*, S^{3}VM) with local minimum convergence guarantee. Experimental results on benchmark datasets not only confirm the finite convergence of IL-S^{3}VM, but also show a huge reduction of computational time compared with existing batch and incremental learning algorithms, while retaining the similar generalization performance.

REFERENCES

[1] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization.* Cambridge university press.

[2] Feng Cai and Vladimir Cherkassky. 2012. Generalized SMO algorithm for SVM-based multitask learning. *IEEE transactions on neural networks and learning systems* 23, 6 (2012), 997–1003.

[3] Olivier Chapelle. 2007. Training a support vector machine in the primal. *Neural computation* 19, 5 (2007), 1155–1178.

[4] Olivier Chapelle, Mingmin Chi, and Alexander Zien. 2006. A continuation method for semi-supervised SVMs. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 185–192.

[5] Olivier Chapelle, Vikas Sindhwani, and S Sathiya Keerthi. 2007. Branch and bound for semi-supervised support vector machines. In *Advances in neural information processing systems*. 217–224.

[6] Olivier Chapelle, Vikas Sindhwani, and Sathiya S Keerthi. 2008. Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research* 9, Feb (2008), 203–233.

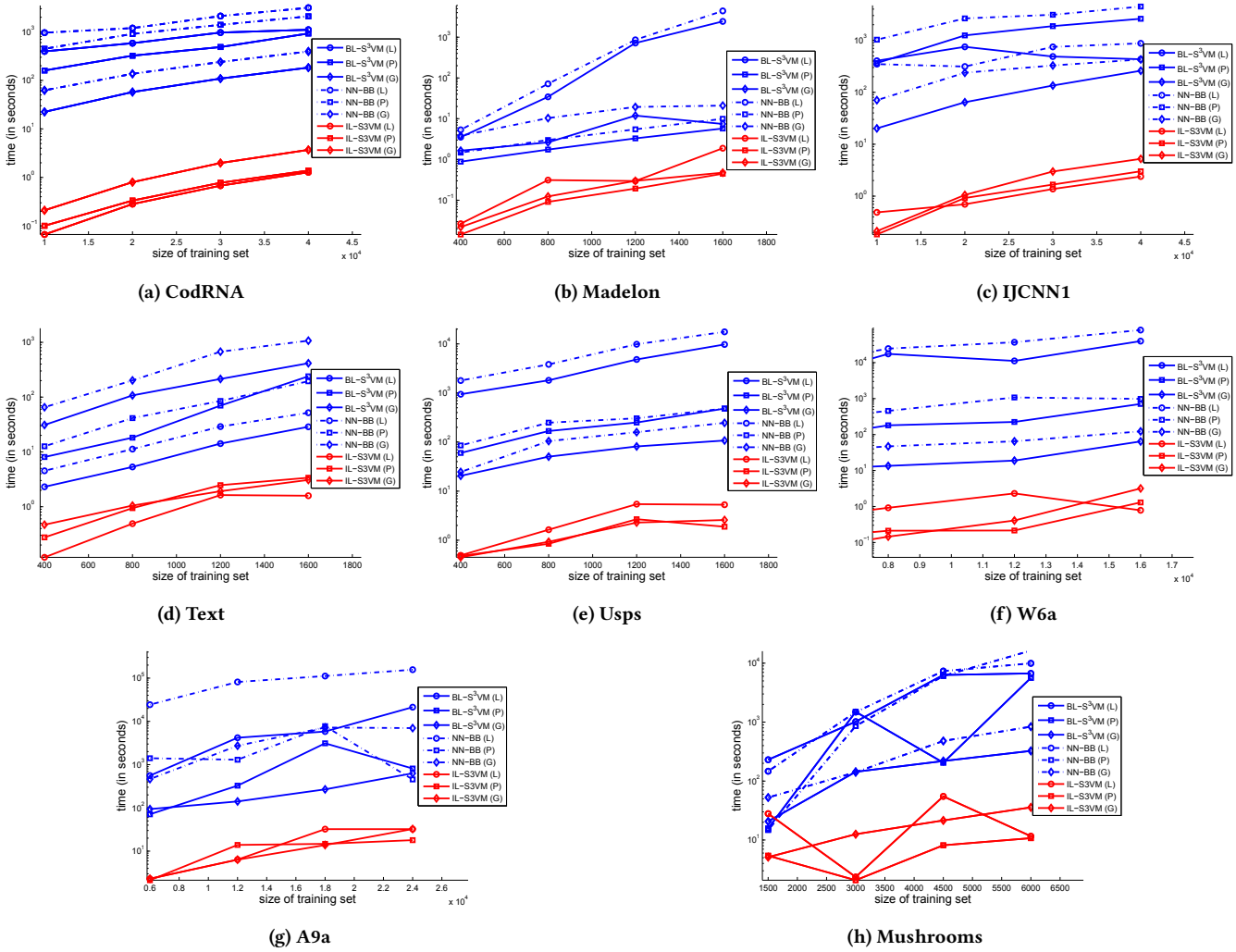
[7] Olivier Chapelle and Alexander Zien. 2005. Semi-Supervised Classification by Low Density Separation.. In *AISTATS*. 57–64.

[8] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. 2006. Large scale transductive SVMs. *Journal of Machine Learning Research* 7, Aug (2006), 1687–1712.

[9] Atam P Dhawan. 2011. *Medical image analysis*. Vol. 31. John Wiley & Sons.

[10] Wael Emara, Mehmed Kantardzic Marcel Karnstedt, Kai-Uwe Sattler, Dirk Habich, and Wolfgang Lehner. 2007. An approach for incremental semi-supervised svm. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*. IEEE, 539–544.

[11] Seyda Ertekin, Leon Bottou, and C. Lee Giles. 2011. Nonconvex Online Support Vector Machines. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 2 (Feb. 2011), 368–381. https://doi.org/10.1109/TPAMI.2010.109



(a) CodRNA (b) Madelon (c) IJCNN1

(d) Text (e) Usps (f) W6a

(g) A9a (h) Mushrooms

Figure 3: Average running time (in seconds) of BL-S^{3}VM, NN-BB and IL-S^{3}VM over 20 trials.

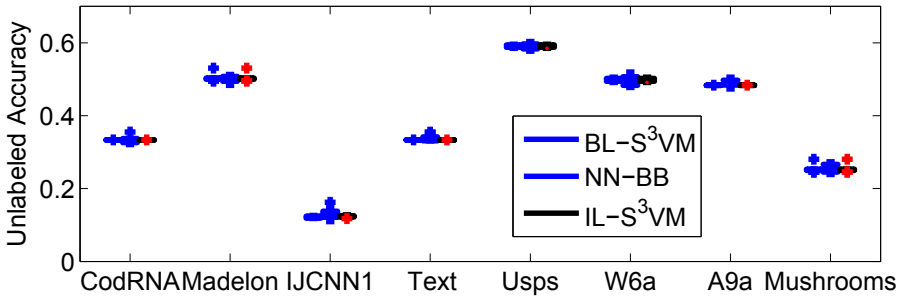


Figure 4: Unlabeled errors of BL-S^{3}VM, NN-BB and IL-S^{3}VM.

[12] Glenn Fung and Olvi L Mangasarian. 2001. Semi-superyised support vector machines for unlabeled data classification. *Optimization methods and software* 15, 1 (2001), 29–44.

[13] Gene H Golub and Charles F Van Loan. 2012. *Matrix computations*. Vol. 3. JHU Press.

[14] Bin Gu and Victor S Sheng. 2013. Feasibility and finite convergence analysis for accurate on-line-support vector machine. *IEEE Transactions on Neural Networks and Learning Systems* 24, 8 (2013), 1304–1315.

[15] B. Gu and V. S. Sheng. 2016. A Robust Regularization Path Algorithm for ν-Support Vector Classification. *IEEE Transactions on Neural Networks and Learning*

Systems PP, 99 (2016), 1–8. https://doi.org/10.1109/TNNLS.2016.2527796

[16] Bin Gu, Victor S Sheng, Keng Yeow Tay, Walter Romano, and Shuo Li. 2015. Incremental support vector learning for ordinal regression. *IEEE Transactions on Neural networks and learning systems* 26, 7 (2015), 1403–1416.

[17] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. 2004. The entire regularization path for the support vector machine. *Journal of Machine Learning Research* 5, Oct (2004), 1391–1415.

[18] William Karush. 1939. *Minima of functions of several variables with inequalities as side constraints*. Ph.D. Dissertation. Master¡s thesis, Dept. of Mathematics, Univ. of Chicago.

[19] Gert R Lanckriet and Bharath K Sriperumbudur. 2009. On the convergence of the concave-convex procedure. In *Advances in neural information processing systems*. 1759–1767.

[20] John Langford, Lihong Li, and Tong Zhang. 2009. Sparse online learning via truncated gradient. *Journal of Machine Learning Research* 10, Mar (2009), 777–801.

[21] Mario Martin. 2002. On-line support vector machine regression. In *European Conference on Machine Learning*. Springer, 282–294.

[22] Chong-Jin Ong, Shiyun Shao, and Jianbo Yang. 2010. An improved algorithm for the solution of the regularization path of support vector machine. *IEEE Transactions on Neural Networks* 21, 3 (2010), 451–462.

[23] T Poggio and G Cauwenberghs. 2001. Incremental and decremental support vector machine learning. *Advances in neural information processing systems* 13 (2001), 409.

[24] David Poole. 2014. *Linear algebra: A modern introduction*. Cengage Learning.

[25] Fabian Sinz and Matteo Roffilli. 2012. UniverSVM. (2012). http://mloss.org/software/view/19/.

[26] Gang Wang, Dit-Yan Yeung, and Frederick H Lochovsky. 2008. A new solution path algorithm in support vector regression. *IEEE Transactions on Neural Networks* 19, 10 (2008), 1753–1767.

[27] Junhui Wang, Xiaotong Shen, and Wei Pan. 2007. On transductive support vector machines. *Contemp. Math.* 443 (2007), 7–20.

[28] Alan L Yuille and Anand Rangarajan. 2003. The concave-convex procedure. *Neural computation* 15, 4 (2003), 915–936.