The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Efficiently Monitoring Small Data Modification Effect for Large-Scale Learning in Changing Environment

Hiroyuki Hanada,[1] Atsushi Shibagaki,[1] Jun Sakuma,[2 3 4] Ichiro Takeuchi[1 4 5]
[1]Department of Computer Science, Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, 466-8555, Japan
[2]Department of Computer Science, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, 305-8577, Japan
[3]Core Research for Evolutional Science and Technology (CREST), Japan Science and Technology Agency,
4-1-8 Honcho, Kawaguchi, 332-0012, Japan
[4]Center for Advanced Intelligence Project, RIKEN, 1-4-1 Nihonbashi, Chuo-ku, Tokyo, 103-0027, Japan
[5]Center for Materials Research by Information Integration, National Institute for Materials Science,
1-2-1 Sengen, Tsukuba, 305-0047, Japan
hanada.hiroyuki@nitech.ac.jp, shibagaki.a.mllab.nit@gmail.com, jun@cs.tsukuba.ac.jp, takeuchi.ichiro@nitech.ac.jp

Abstract

We study large-scale machine learning problems in changing environments where a small part of the dataset is modified, and the effect of the data modification must be monitored in order to know how much the modification changes the optimal model. When the entire dataset is large, even if the amount of the data modification is fairly small, the computational cost for re-training the model would be prohibitively large. In this paper, we propose a novel method, called the *optimal solution bounding (OSB)*, for monitoring such a data modification effect on the optimal model by efficiently evaluating (without actually re-training) it. The proposed method provides bounds on the unknown optimal model with the cost proportional only to the size of the data modification.

1 Introduction

1.1 Problem Overview

In this paper we study the problem of training a classifier such as support vector machine (SVM) with a large-scale dataset. When a trained classifier is used in a changing environment where small part of the dataset is constantly modified, we need to update the classifier accordingly in order to incorporate the effect of the data modification. However, when the entire dataset is large, even if the amount of the data modification is fairly small, the computational cost of re-training the classifier, e.g. by an incremental learning method, would be prohibitively large.

In this paper, we propose a novel method for efficiently evaluating the effect of the data modification on the classifier without actually re-training it. In this paper, for a class of problems training binary classifiers such as SVM, we propose a novel method for efficiently evaluating the effect of the data modification on the optimal model (e.g. classifier) without actually re-training it. The proposed method provides bounds on the new classifier with the cost proportional only to the size of the data modification. This computational advantage is particularly beneficial when the size of the data modification is much smaller than the size of the entire dataset. The bounds obtained by the proposed method can be used for various tasks in classification problems.

Concretely, consider a linear binary classification problem with n instances and d features, and write the data matrix as $X \in \mathbb{R}^{n \times d}$. Denoting the set of modified elements in X as $\mathcal{M}$, we are particularly interested in the situation in which only a small part of X is modified, i.e., $|\mathcal{M}|$ is much smaller than nd. Let $\boldsymbol{w}^{*\mathrm{old}} \in \mathbb{R}^d$ and $\boldsymbol{w}^{*\mathrm{new}} \in \mathbb{R}^d$ be the linear model parameters of the old and the new classifiers, respectively. The main contribution in this paper is to develop a method called *optimal solution bounding (OSB) method*, which can efficiently compute a lower bound $L[w_j^{*\mathrm{new}}]$ and an upper bound $U[w_j^{*\mathrm{new}}]$ of unknown $w_j^{*\mathrm{new}}$ such that

$$L[w_j^{*\mathrm{new}}] \leq w_j^{*\mathrm{new}} \leq U[w_j^{*\mathrm{new}}] \text{ for each } j \in [d] \qquad (1)$$

rather than computing exact $\boldsymbol{w}^{*\mathrm{new}}$, in the cost proportional only to the number of modified elements $|\mathcal{M}|$. This is much smaller than the cost of computing exact $w_j^{*\mathrm{new}}$, which requires at least $O(nd)$ time since we need to go through the entire data matrix X at least once.

Figure 1 illustrates the motivation of this study as well as three scenarios of data modifications considered in this paper. Let us consider a database whose values are frequently updated by users, for example, a database for evaluating movies by users (e.g., MovieLens dataset (Harper and Konstan 2015)). Let us regard users as instances, movies as features, and the values in X as evaluations for movies by users. In the figure, scenario (a) illustrates a situation in which users newly evaluated movies, i.e., the values in X are modified (spot modifications). Scenario (b) illustrates a situation in which a user joined the service, i.e., the values in the row in X representing the user are modified (instance modification). Scenario (c) illustrates a situation in which a new movie is released, i.e., the values in the column representing the movie are modified (feature modification).

In these scenarios, it is impractical to update the classifier every time there is a modification in X. Therefore, it is important to identify the degree to which the data modification can change the classifier without actually re-training it. Figure 2 illustrates the problem setup and the difference between conventional incremental learning approach and the

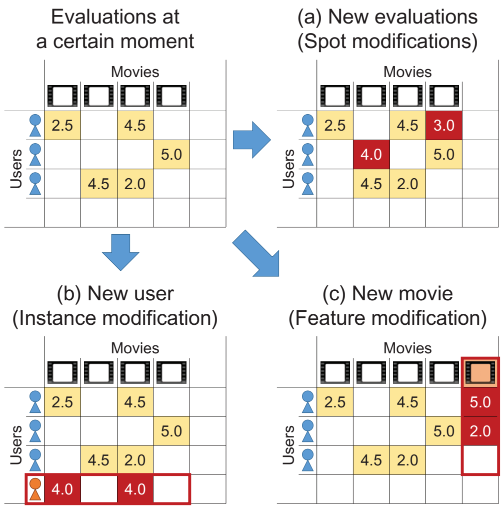

Evaluations at a certain moment
(a) New evaluations (Spot modifications)
(b) New user (Instance modification)
(c) New movie (Feature modification)
Movies
Users
2.5
4.5
5.0
2.0
3.0
4.0

Figure 1: Schematic illustration of the three types of data modifications considered here. When a small part of the data matrix X is modified, we need to quickly evaluate the effect of the data modification on the classifier. The proposed method can compute a lower bound and an upper bound of the new optimal solution parameter $w_j^{*\text{new}}$ with the cost proportional only to the number of modified elements.

proposed OSB approach.

We also discuss an approach for tightening the bounds in (1) with a slight increase in computational cost (§4). For example, when $\mathcal{M} = \{\{(i, j')\}_{i=1}^n\}$, i.e., the j'th column of X is modified (scenario (c) in Figure 1), we conjecture that the j'th parameter of the optimization problem ((2) in §2.1) would change significantly. For such a case, we consider solving the optimization problem only for the j'th variable, i.e., we optimize it *partially*, to obtain a tighter bound.

1.2 Related Works and Our Contributions

In many practical machine learning tasks, we often need to solve multiple related optimization problems. *Incremental learning* methods have been developed for such cases (Laskov et al. 2006; Tsai, Lin, and Lin 2014). A general popular approach in incremental learning is the *warm-start*, where the optimal solution of a related problem is used as an initial starting point of the optimization problem (DeCoste and Wagstaff 2000; Tsai, Lin, and Lin 2014). In the data modification scenarios considered herein, we can use the old parameters $\boldsymbol{w}^{*\text{old}}$ as an initial starting point of the optimization problem in order to obtain the new parameters $\boldsymbol{w}^{*\text{new}}$. Unfortunately, however, even when one uses the warm-start approach, the computational cost of re-training SVM-like classifier is at least $\mathcal{O}(nd)$ because one needs to go through the entire data matrix X at least once for checking the optimality of the new solution. For SVM or several variants of SVMs, specific incremental learning algorithms are proposed (Cauwenberghs and Poggio 2001; Karasuyama and Takeuchi 2010; Zhu et al. 2012; Chitrakar and Huang 2014; Gu et al. 2015a; 2015b; Gâlmeanu, Sasu,

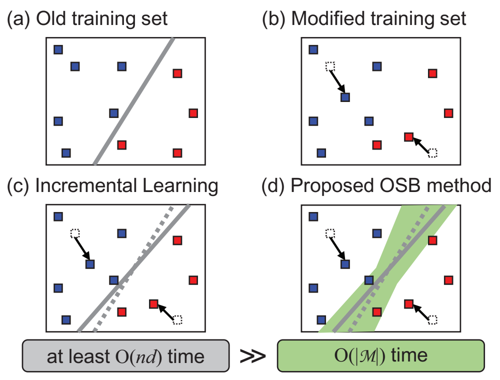


Figure 2: Schematic illustration of the problem setup and the difference between the conventional incremental learning method and the proposed OSB method for a simple linear binary classification example. (a) Assume that we know the optimal classification boundary trained from the old training set. (b) Then, suppose that a part of the training set is modified. (c) In the conventional incremental learning method, the classifier is completely re-trained using the new training set, which requires a computational cost of at least $O(nd)$ (i.e., the total size of the training data). (d) The proposed OSB method computes the bounds of the classification boundary with a much smaller computational cost that depends only on the number of modified elements $\mathcal{M}$.

and Andonie 2016), but their computational cost is still depends at least on $\mathcal{O}(nd)$.

The efficiency of the proposed method can be achieved by computing only intervals of the optimal solutions as in (1), rather than computing the exact optimal solution, as described in Figure 2. The main technical contribution in this paper is to develop a novel algorithm for computing bounds of the new optimal solution after data modification by employing a technique in convex analysis (El Ghaoui, Viallon, and Rabbani 2012; Ogawa, Suzuki, and Takeuchi 2013; Ndiaye et al. 2015; Shibagaki et al. 2015; 2016; Takada et al. 2016; Nakagawa et al. 2016).

The proposed method is especially useful for monitoring the model in changing environment. As in the movie evaluation example in Figure 1, when a small part of the database is gradually changing, it is impractical to update the classifier each time. In such a case, it is reasonable to update the classifier only when the current one is fairly away from the optimality and not useful anymore. The proposed OSB method is useful for judging when we should update the classifier. By using the method, we compute the bounds of the optimal solution for checking the goodness of the current classifier, and update it only when needed. In experiment section §5.1, we illustrate the usefulness of the proposed OSB method. In some specific tasks, efficient monitoring methods have been studied (Gabel, Keren, and Schuster 2015; Okumura, Suzuki, and Takeuchi 2015), but none of these existing methods are not as general as ours.

Although we mainly discuss binary classification problems in this paper, the proposed method can be easily extended to regression problems since all properties are derived without depending on the fact that the outcome is binary. Detailed formulation is shown in Appendix E.

2 Problem Setup and Background

We denote a vector by bold italic such as $\boldsymbol{v}$, and its ith element by a non-bold italic with a subscript such as v_i. For any natural number n, we define $[n] := \{1, \dots, n\}$. The L_2 norm of a vector $\boldsymbol{v} \in \mathbb{R}^m$ is written as $\|\boldsymbol{v}\|_2 := (\sum_{k\in[m]} |v_k|^2)^{1/2}$. A lower bound and an upper bound of a scalar quantity are respectively represented by the notations $L[\cdot]$ and $U[\cdot]$, e.g., $L[w_j] \leq w_j \leq U[w_j]$.

We write the subdifferential operator as ∂. For a function f, its domain is denoted as $\mathrm{dom} f$. Moreover, for a convex function $f(\boldsymbol{v}) : \mathbb{R}^k \to \mathbb{R}$, we denote its *convex conjugate* by $f^*(\boldsymbol{v}) := \sup_{\boldsymbol{u}\in\mathbb{R}^k}\{\boldsymbol{v}^\top\boldsymbol{u} - f(\boldsymbol{u})\}$. Note that $f^{**}(\boldsymbol{v}) = f(\boldsymbol{v})$ if f is a lower-semi-continuous convex function.

A function $f(\boldsymbol{v}) : \mathbb{R}^k \to \mathbb{R}$ is called λ-*strongly convex* if the following holds for any $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^k$: $f(\boldsymbol{b}) - f(\boldsymbol{a}) \geq \partial f(\boldsymbol{a})^\top(\boldsymbol{b} - \boldsymbol{a}) + \frac{\lambda}{2}\|\boldsymbol{b} - \boldsymbol{a}\|_2^2$. A function $f(\boldsymbol{v}) : \mathbb{R}^k \to \mathbb{R}$ is called ν-*smooth* if the following holds for any $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^k$: $\|\nabla f(\boldsymbol{b}) - \nabla f(\boldsymbol{a})\|_2 \leq \nu\|\boldsymbol{b} - \boldsymbol{a}\|_2$ (i.e., the gradient $\nabla f(\boldsymbol{v})$ is ν-Lipschitz). It is known that a convex function $f(\boldsymbol{v})$ is ν-smooth if and only if $f^*(\boldsymbol{v})$ is $(1/\nu)$-strongly convex (see, for example, Theorem 4.2.2 in (Hiriart-Urruty and Lemarechal 1993) for the proof).

2.1 Convex Regularized Learning Problems

In this paper, we study binary classification problems with n training instances and d features. Throughout the paper we use the following notations about the dataset:

Definition 2.1. *Assume binary classification problems with n training instances and d features. The entire dataset is denoted as $(X, \boldsymbol{y})$, where $X \in \mathbb{R}^{n\times d}$ is the input matrix and $\boldsymbol{y} \in \{\pm 1\}^n$ is the label vector. The i^{th} row vector, the j^{th} column vector, and the $(i,j)^{\text{th}}$ element of X are respectively written as $\boldsymbol{x}_{i\cdot} \in \mathbb{R}^d$, $\boldsymbol{x}_{\cdot j} \in \mathbb{R}^n$, and $x_{ij} \in \mathbb{R}$.*

Furthermore, we define $Z := \mathrm{diag}(\boldsymbol{y})X \in \mathbb{R}^{n\times d}$ for notational simplicity, and $\boldsymbol{z}_{i\cdot} \in \mathbb{R}^d$, $\boldsymbol{z}_{\cdot j} \in \mathbb{R}^n$, and $z_{ij} \in \mathbb{R}$ are also defined similarly.

We consider the following class of *convex regularized empirical risk minimization problems*:

Definition 2.2. *A* convex regularized empirical risk minimization *problem for binary classification is defined as:*

$$\boldsymbol{w}^* := \operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} \mathcal{P}(\boldsymbol{w}),$$
$$\mathcal{P}(\boldsymbol{w}) := \tfrac{1}{n}\textstyle\sum_{i\in[n]} \phi(\boldsymbol{z}_{i\cdot}^\top\boldsymbol{w}) + \psi(\boldsymbol{w}), \tag{2}$$

where $\phi : \mathbb{R} \to \mathbb{R}_+$ is a convex loss function and $\psi : \mathbb{R}^d \to \mathbb{R}_+$ is a convex penalty function.

The dual problem *of* (2) *derived by Fenchel's duality theorem (see, e.g., Corollary 31.2.1 in (Rockafellar 1970)) is written using the convex conjugates ϕ^* and ψ^* as*

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}\in\mathbb{R}^n} \mathcal{D}(\boldsymbol{\alpha}),$$
$$\mathcal{D}(\boldsymbol{\alpha}) := -\tfrac{1}{n}\textstyle\sum_{i\in[n]} \phi^*(-\alpha_i) - \psi^*\left(\tfrac{1}{n}Z^\top\boldsymbol{\alpha}\right), \tag{3}$$

where $\mathcal{P}(\boldsymbol{w}^*) = \mathcal{D}(\boldsymbol{\alpha}^*)$ holds under certain conditions.

If we know either of the optimal solutions of (2) or (3), the other can be easily computed: In fact, the following conditions known as the *KKT condition* must hold ((Rockafellar 1970), Section 31):

$$w_j^* \in -\partial\psi_j^*\left(\frac{1}{n}\boldsymbol{z}_{\cdot j}{}^\top\boldsymbol{\alpha}^*\right), \tag{4}$$

$$\alpha_i^* \in \partial\phi(\boldsymbol{z}_{i\cdot}{}^\top\boldsymbol{w}^*). \tag{5}$$

Example 2.1. *As a working example, we study the* smoothed-hinge L2-regularized SVM. *For tuning parameters $\lambda, \gamma > 0$, it is obtained by setting ϕ and ψ as follows:*

$$\phi(r) := \begin{cases} 0 & (r > 1), \\ 1 - r - \frac{\gamma}{2} & (r < 1 - \gamma), \\ \frac{1}{2\gamma}(1 - r)^2 & (\text{otherwise}), \end{cases} \tag{6a}$$

$$\psi(\boldsymbol{w}) := \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2. \tag{6b}$$

ϕ^* and ψ^* are computed as follows:

$$\phi^*(r) = \begin{cases} r + \frac{\gamma}{2}r^2 & (-1 \leq r \leq 0), \\ +\infty & (\text{otherwise}), \end{cases} \tag{7a}$$

$$\psi^*(\boldsymbol{w}) = \frac{1}{2\lambda}\|\boldsymbol{w}\|_2^2. \tag{7b}$$

2.2 Small Data Modification

In this study, we consider a situation in which a small portion of the input matrix X is modified.

Definition 2.3. *The set of modified elements in X is denoted as $\mathcal{M} \subseteq \{[n] \times [d]\}$ and its size is denoted as $|\mathcal{M}|$. Furthermore, we define $\mathcal{M}_i := \{i \in [n] \mid \exists j \in [d] \text{ s.t. } (i,j) \in \mathcal{M}\}$ and $\mathcal{M}_j := \{j \in [d] \mid \exists i \in [n] \text{ s.t. } (i,j) \in \mathcal{M}\}$.*

We refer to the problems before and after the data modification as the old *problem and the* new *problem, respectively. Superscripts "old" and "new" indicate variables for the old and the new problems, respectively. Furthermore, we denote the primal problem* (2) *and the dual problem* (3) *for the data Z^{old} by $\mathcal{P}^{\text{old}}$ and $\mathcal{D}^{\text{old}}$, respectively, and the optimal solutions by $\boldsymbol{w}^{*\text{old}}$ and $\boldsymbol{\alpha}^{*\text{old}}$, respectively. $\mathcal{P}^{\text{new}}$, $\mathcal{D}^{\text{new}}$, $\boldsymbol{w}^{*\text{new}}$ and $\boldsymbol{\alpha}^{*\text{new}}$ are similarly defined.*

The primary contribution of this paper is to present methods for computing bounds of primal variables $L[w_j^{*\text{new}}]$, $U[w_j^{*\text{new}}]$ ($j \in [d]$) and dual variables $L[\alpha_i^{*\text{new}}]$, $U[\alpha_i^{*\text{new}}]$ ($i \in [n]$) with the computational cost depending only on the number of modified elements $|\mathcal{M}|$.

2.3 Decision Making Using the Bounds of the New Solution

Using these bounds, we can perform several decision making tasks on the new solution $(\boldsymbol{w}^{*\text{new}}, \boldsymbol{\alpha}^{*\text{new}})$. First, consider the situation in which we want to classify a test instance $\boldsymbol{x}' \in \mathbb{R}^d$ based on the new classifier. In this task, we compute a lower bound and an upper bound of the classification score $f(\boldsymbol{x}') = \boldsymbol{x}'^\top\boldsymbol{w}^{*\text{new}}$. Using the lower and the upper bounds, the class label $y' = f(\boldsymbol{x}')$ of the test instance $\boldsymbol{x}'$ can be determined as

$$U[\boldsymbol{x}'^\top \boldsymbol{w}^{*\mathrm{new}}] < 0 \quad \Rightarrow \quad y' = -1; \tag{8a}$$

$$L[\boldsymbol{x}'^\top \boldsymbol{w}^{*\mathrm{new}}] \geq 0 \quad \Rightarrow \quad y' = +1. \tag{8b}$$

(If neither of them holds, y' is not determined by the bound.) Equation (8) suggests that, even if the new optimal solution $\boldsymbol{w}^{*\mathrm{new}}$ is not available, if its bounds are sufficiently tight, then the classification task can be completed for some instances.

Next, consider the situation in which the data matrix X is constantly changing. In such a situation, an important decision making task is to determine when we should actually re-train the classifier. Suppose that we have a tolerance threshold $\theta > 0$ for the degree to which the classifier can be different from the optimal classifier. If we quantify the difference by L_2 norm $\|\boldsymbol{w}^{*\mathrm{new}} - \boldsymbol{w}^{*\mathrm{old}}\|_2$, we can re-train the classifier only when

$$U[\|\boldsymbol{w}^{*\mathrm{new}} - \boldsymbol{w}^{*\mathrm{old}}\|_2] \geq \theta. \tag{9}$$

Equations (8) and (9) indicate that, even if the optimal solution $\boldsymbol{w}^{*\mathrm{new}}$ is not available, we can make sure that these relationships about $\boldsymbol{w}^{*\mathrm{new}}$ hold using $L[w_j^{*\mathrm{new}}]$ and $U[w_j^{*\mathrm{new}}]$ $(j \in [d])$.

2.4 Performance Measure for Bounds

Given a bound for $\boldsymbol{w}^{*\mathrm{new}}$ and a test data set $\{(y_i^{\mathrm{test}}, \boldsymbol{x}_i^{\mathrm{test}})\}_{i\in[m]}$, we measure the performance of the bound by

$$\frac{|T_{-1}| + |T_{+1}|}{m}, \text{ where}$$

$$T_{-1} := \{i \mid i \in [m], U[\boldsymbol{x}_i^{\mathrm{test}\top} \boldsymbol{w}^{*\mathrm{new}}] < 0\boldsymbol{w}^{*\mathrm{new}}] \geq 0\},$$

$$T_{+1} := \{i \mid i \in [m], L[\boldsymbol{x}_i^{\mathrm{test}\top} \boldsymbol{w}^{*\mathrm{new}}] \geq 0\},$$

(i.e. the ratio of test samples that satisfies either of (8)), which we call the *label determination rate*. The rate expresses the availability of the bound as a classifier. Moreover, we can interpret the bound is closer to the true training result if the rate is higher, because in such a case the region that the bound cannot determine the true classification result (green region in Figure 2(d)) becomes smaller and therefore closer to the true training result (Figure 2(c)).

3 Efficient Bound Computation after a Small Data Modification

Our main results are presented in this section. The goal of the study is to efficiently compute lower and upper bounds of the new optimal solutions $(\boldsymbol{w}^{*\mathrm{new}}, \boldsymbol{\alpha}^{*\mathrm{new}})$ using the old optimal solutions $(\boldsymbol{w}^{*\mathrm{old}}, \boldsymbol{\alpha}^{*\mathrm{old}})$ in $\mathcal{O}(|\mathcal{M}|)$ time. When the number of modified elements is much smaller than the total number of the elements nd, it is quite beneficial to be able to compute the bounds of the new optimal solution in such an efficient manner.

Theorem 3.1. *Assume that the following quantities*

$$\boldsymbol{z}_{i\cdot}^{\mathrm{old}\top} \boldsymbol{w}^{*\mathrm{old}},\ \|\boldsymbol{x}_{i\cdot}^{\mathrm{old}}\|_2,\ \boldsymbol{z}_{\cdot j}^{\mathrm{old}\top} \boldsymbol{\alpha}^{*\mathrm{old}},\ \|\boldsymbol{x}_{\cdot j}^{\mathrm{old}}\|_2,$$

$$\inf_{\boldsymbol{\alpha}\in\mathrm{dom}\mathcal{D}^{\mathrm{new}}} n^{-1} \boldsymbol{z}_{\cdot j}^{\mathrm{old}\top} \boldsymbol{\alpha},\quad \sup_{\boldsymbol{\alpha}\in\mathrm{dom}\mathcal{D}^{\mathrm{new}}} n^{-1} \boldsymbol{z}_{\cdot j}^{\mathrm{old}\top} \boldsymbol{\alpha}, \tag{10}$$

for all $i \in [n]$ and $j \in [d]$, are stored in the memory with the size $\mathcal{O}(n + d)$, and the penalty function ψ is decomposable *in the sense that there exists d convex functions $\psi_j : \mathbb{R} \to \mathbb{R}$, $j \in [d]$, such that $\psi(\boldsymbol{w}) = \sum_{j\in[d]} \psi_j(w_j)$ for all $\boldsymbol{w} \in \mathbb{R}^d$. Then, in each of the following four cases (a1), (a2), (b1) and (b2), the lower and the upper bounds of $\{w_j^{*\mathrm{new}}\}_{j\in[d]}$ and $\{\alpha_i^{*\mathrm{new}}\}_{i\in[n]}$ can be evaluated with time complexity $\mathcal{O}(|\mathcal{M}|)$.*

(a1) *If* **the penalty function ψ is λ-strongly convex**, (11) *and* (12) *can be computed in $O(|\mathcal{M}|)$ time:*

$$\|\boldsymbol{w}^{*\mathrm{new}} - \boldsymbol{w}^{*\mathrm{old}}\|_2 \leq r_P, \tag{11}$$

$$w_j^{*\mathrm{old}} - r_P \leq w_j^{*\mathrm{new}} \leq w_j^{*\mathrm{old}} + r_P, \tag{12}$$

where $r_P := \sqrt{(2/\lambda)[\mathcal{P}^{\mathrm{new}}(\boldsymbol{w}^{*\mathrm{old}}) - \mathcal{D}^{\mathrm{new}}(\boldsymbol{\alpha}^{*\mathrm{old}})]}$. In addition, (13) *can be computed in $O(d + |\mathcal{M}|)$ time:*

For any $\boldsymbol{\eta} \in \mathbb{R}^d$:

$$\boldsymbol{\eta}^\top \boldsymbol{w}^{*\mathrm{old}} - r_P\|\boldsymbol{\eta}\|_2 \leq \boldsymbol{\eta}^\top \boldsymbol{w}^{*\mathrm{new}} \leq \boldsymbol{\eta}^\top \boldsymbol{w}^{*\mathrm{old}} + r_P\|\boldsymbol{\eta}\|_2. \tag{13}$$

(a2) *In addition to the condition in (a1), if* **the loss function ϕ is subdifferentiable and bounded in its domain**, *then* (14) *can be computed in $O(|\mathcal{M}|)$ time:*

$$\inf -\partial\phi\left(\boldsymbol{z}_{i\cdot}^{\mathrm{new}\top} \boldsymbol{w}^{*\mathrm{old}} + r_P\|\boldsymbol{z}_{i\cdot}^{\mathrm{new}}\|_2\right) \leq \alpha_i^{*\mathrm{new}}$$
$$\leq \sup -\partial\phi\left(\boldsymbol{z}_{i\cdot}^{\mathrm{new}\top} \boldsymbol{w}^{*\mathrm{old}} - r_P\|\boldsymbol{z}_{i\cdot}^{\mathrm{new}}\|_2\right). \tag{14}$$

(b1) *If* **the loss function ϕ is $(1/\mu)$-smooth**, (15) *and* (16) *can be computed in $O(|\mathcal{M}|)$ time:*

$$\|\boldsymbol{\alpha}^{*\mathrm{new}} - \boldsymbol{\alpha}^{*\mathrm{old}}\|_2 \leq r_D, \tag{15}$$

$$\alpha_i^{*\mathrm{old}} - r_D \leq \alpha_i^{*\mathrm{new}} \leq \alpha_i^{*\mathrm{old}} + r_D, \tag{16}$$

where $r_D := \sqrt{(2n/\mu)[\mathcal{P}^{\mathrm{new}}(\boldsymbol{w}^{*\mathrm{old}}) - \mathcal{D}^{\mathrm{new}}(\boldsymbol{\alpha}^{*\mathrm{old}})]}$. In addition, (17) *can be computed in $O(n + |\mathcal{M}|)$ time:*

For any $\boldsymbol{\zeta} \in \mathbb{R}^n$:

$$\boldsymbol{\zeta}^\top \boldsymbol{\alpha}^{*\mathrm{old}} - r_D\|\boldsymbol{\zeta}\|_2 \leq \boldsymbol{\zeta}^\top \boldsymbol{\alpha}^{*\mathrm{new}} \leq \boldsymbol{\zeta}^\top \boldsymbol{\alpha}^{*\mathrm{old}} + r_D\|\boldsymbol{\zeta}\|_2. \tag{17}$$

(b2) *In addition to the condition in (b1), if* **the convex conjugate of the penalty function ψ^* is subdifferentiable and bounded in its domain**, *then* (18) *can be computed in $O(|\mathcal{M}|)$ time:*

$$\inf \partial\psi_j^*\left(F\left(\tfrac{1}{n}[\boldsymbol{z}_{\cdot j}^{\mathrm{new}\top} \boldsymbol{\alpha}^{*\mathrm{old}} - r_D\|\boldsymbol{z}_{\cdot j}^{\mathrm{new}}\|_2]\right)\right) \leq w_j^{*\mathrm{new}}$$
$$\leq \sup \partial\psi_j^*\left(F\left(\tfrac{1}{n}[\boldsymbol{z}_{\cdot j}^{\mathrm{new}\top} \boldsymbol{\alpha}^{*\mathrm{old}} + r_D\|\boldsymbol{z}_{\cdot j}^{\mathrm{new}}\|_2]\right)\right), \tag{18}$$

where

$$F(t) := \begin{cases} \mathcal{L} := \inf_{\boldsymbol{\alpha}\in\mathrm{dom}\mathcal{D}^{\mathrm{new}}} \frac{1}{n}\boldsymbol{z}_{\cdot j}^{\mathrm{new}\top}\boldsymbol{\alpha}, & \textit{(if } t \leq \mathcal{L}\textit{)} \\ \mathcal{U} := \sup_{\boldsymbol{\alpha}\in\mathrm{dom}\mathcal{D}^{\mathrm{new}}} \frac{1}{n}\boldsymbol{z}_{\cdot j}^{\mathrm{new}\top}\boldsymbol{\alpha}, & \textit{(if } t \geq \mathcal{U}\textit{)} \\ t. & \textit{(otherwise)} \end{cases} \tag{19}$$

$F(t)$ assures that the argument for $\partial\psi^*$ does not take infeasible values.

The proof of Theorem 3.1 is presented in Appendix A.

Remark 3.1. *The assumption that all of the quantities in* (10) *are available in the memory is reasonable because they can be computed when the old problem was solved for obtaining* $\boldsymbol{w}^{*\text{old}}$ *and* $\boldsymbol{\alpha}^{*\text{old}}$.

The assumption on decomposability is satisfied in many penalty functions including the L_q^q*-norm with* $q \geq 1$.

Remark 3.2. *If both of the conditions in (a1) and (b2) hold, then the bounds of* $w_j^{*\text{new}}$ *can be computed in two ways. In this case, we can use their intersection as the bounds of* $w_j^{*\text{new}}$*. Similarly, if both of the conditions in (a2) and (b1) hold, then the bounds of* $\alpha_i^{*\text{new}}$ *can be computed in two ways.*

Remark 3.3. *If the conditions in (a1) hold, then* (8) *and* (9) *in* §*2.3 are directly computed with* (13) *and* (11)*, respectively. If the conditions in (b2) hold, then* (8) *and* (9) *can be computed as follows:*

$$L[\boldsymbol{x}'^\top \boldsymbol{w}^{*\text{new}}] = \sum_{j|x'_{ij}\geq 0} x'_{ij} L[w_j^{*\text{new}}] + \sum_{j|x'_{ij}<0} x'_{ij} U[w_j^{*\text{new}}],$$

$$U[\boldsymbol{x}'^\top \boldsymbol{w}^{*\text{new}}] = \sum_{j|x'_{ij}\geq 0} x'_{ij} U[w_j^{*\text{new}}] + \sum_{j|x'_{ij}<0} x'_{ij} L[w_j^{*\text{new}}],$$

$$U[\|\boldsymbol{w}^{*\text{new}} - \boldsymbol{w}^{*\text{old}}\|_2] = \sqrt{\sum_{j\in[d]} \max\{w_j^{*\text{old}} - L[w_j^{*\text{new}}], U[w_j^{*\text{new}}] - w_j^{*\text{old}}\}^2}.$$

Example 3.1. *Both the* L_2 *penalty* $\psi(\boldsymbol{w}) := (\lambda/2)\|\boldsymbol{w}\|_2^2$ *and the elastic-net penalty* $\psi(\boldsymbol{w}) := (\lambda/2)\|\boldsymbol{w}\|_2^2 + \kappa\|\boldsymbol{w}\|_1$ *satisfy the conditions of both (a1) and (b2). However, the* L_1 *penalty* $\psi(\boldsymbol{w}) := \kappa\|\boldsymbol{w}\|_1$ *does not satisfy the conditions of either (a1) or (b2).*

The smoothed hinge loss ($\phi(r)$ *in* (6a)*), the squared hinge loss* $\phi(r) := \max\{0, 1-r\}^2$ *and the logistic regression loss* $\phi(r) := \log(1 + e^{-r})$ *all satisfy the conditions of both (b1) and (a2). The vanilla hinge loss* $\phi(r) := \max\{0, 1-r\}$ *satisfies only the conditions of (a2).*

In Appendix B, we present the bounds in Theorem 3.1 for the smoothed-hinge SVM as an example.

4 Partial Optimization for Obtaining Tighter Bounds

Thus far, we considered computing the bounds of $w_j^{*\text{new}}$ and $\alpha_i^{*\text{new}}$ in $\mathcal{O}(|\mathcal{M}|)$ time in Theorem 3.1. In this section, we consider obtaining tighter bounds by *partially* solving the optimization problem, which is more costly than $\mathcal{O}(|\mathcal{M}|)$ but much less than solving the full optimization problem (2) or (3). As stated in Theorem 3.1, the bounds of $w_j^{*\text{new}}$ and $\alpha_j^{*\text{new}}$ becomes tighter if r_P and r_D becomes smaller. To make them smaller with a small computational cost, instead of just computing $\mathcal{P}^{\text{new}}(\boldsymbol{w}^{*\text{old}})$, we consider optimizing it for the subset of their variables $\mathcal{J} \subset [d]$. We can take a similar approach for $\mathcal{D}^{\text{new}}$. This fact is formally stated as follows.

Theorem 4.1. *Assume that the penalty function is decomposable as in Theorem 3.1. Consider partially optimizing the primal problem* (2) *(resp. dual problem* (3)*) only w.r.t. a subset of the primal variables* $\{w_j\}_{j\in\mathcal{J}\subseteq[d]}$ *(resp. dual variables* $\{\alpha_i\}_{i\in\mathcal{I}\subseteq[n]}$*) while other variables are replaced with* $\boldsymbol{w}^{*\text{old}}$ *(resp.* $\boldsymbol{\alpha}^{*\text{old}}$*).*

Let $\breve{\boldsymbol{w}}^* \in \mathbb{R}^d$ *(resp.* $\breve{\boldsymbol{\alpha}}^* \in \mathbb{R}^n$*) be a vector for which the elements corresponding to* $\mathcal{J}$ *(resp.* $\mathcal{I}$*) are the solution of the partial problem, while the other elements are the old solutions* $\{w_j^{*\text{old}}\}_{j\in[d]\setminus\mathcal{J}}$ *(resp.* $\{\alpha_i^{*\text{old}}\}_{i\in[n]\setminus\mathcal{I}}$*). Then, unless* $\breve{\boldsymbol{w}}^* = \boldsymbol{w}^{*\text{old}}$ *(resp.* $\breve{\boldsymbol{\alpha}}^* = \boldsymbol{\alpha}^{*\text{old}}$*), all of the bounds in Theorem 3.1 is strictly tightened by replacing* $\boldsymbol{w}^{*\text{old}}$ *with* $\breve{\boldsymbol{w}}^*$ *(resp.* $\boldsymbol{\alpha}^{*\text{old}}$ *with* $\breve{\boldsymbol{\alpha}}^*$*).*

The proof is shown in Appendix C. We show examples for $|\mathcal{J}| = 1$ and $|\mathcal{I}| = 1$ in Appendix D.

Finally, for each of the three scenarios depicted in Figure 1, we consider specific partial optimization strategy. First, for the spot modification scenario, we set $\mathcal{J} := \mathcal{M}_j$ and $\mathcal{I} := \mathcal{M}_i$; i.e., we optimize the primal (resp. dual) variables if there is at least one modification at the corresponding columns (resp. rows) of X. For the instance modification scenario, we set $\mathcal{J} := \emptyset$ and $\mathcal{I} := \mathcal{M}_i$; i.e., we optimize only the dual variables corresponding to the modified rows of X. Similarly, for the feature modification scenario, we set $\mathcal{J} := \mathcal{M}_j$ and $\mathcal{I} := \emptyset$; i.e., we optimize only the primal variables corresponding to the modified columns of X.

5 Experiment

In this section, we demonstrate the performance of the proposed OSB through numerical experiments. In all of the experiments, we used the smoothed-hinge SVM ((6) and (7)) with $\gamma = 1$.

5.1 An Illustrative Example on MovieLens Dataset

We first illustrate how to use the proposed OSB in practice by considering the scenario in Figure 1. We used a part of the MovieLens dataset (Harper and Konstan 2015) converted to a binary classification problem (see Appendix F) to predict whether a user would evaluate a movie (ID 356) as interesting. The goal of this illustrative study is to demonstrate how the proposed OSB can be used to maintain the quality of the classifier when the data matrix X is gradually changing. Here, the quality of the classifier is measured by the label determination rate (§2.4). Following the data preparation process in Appendix F, the training data matrix has 69,260 instances (users) and 38,187 features (movies) with 13,810,969 non-zero values (evaluations for movies; sparsity is 13,810,969/(69,260 $\times$ 38,187) $= 0.52\%$). We also prepared a separate validation dataset of 17,315 instances.

We assume that some elements, rows, and columns of the training data matrix X are initially missing and are constantly being filled in, as indicated by cases (a), (b) and (c) in Figure 1. To implement this situation, we first remove approximately 3% of the evaluations (approximately 1% each for of (a), (b) and (c), and approximately 13,810,969$\times$3%$\approx$414,329 evaluations are removed). Then

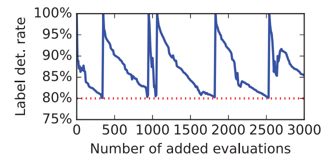
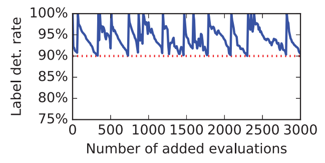
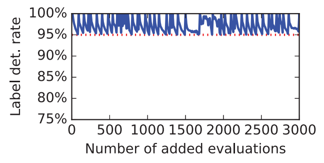
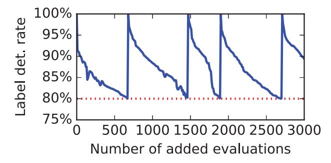
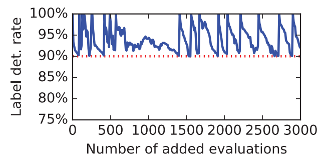
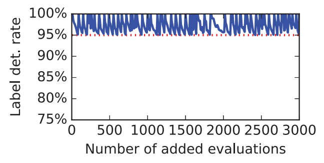
Threshold of the rate: 80%	Threshold of the rate: 90%	Threshold of the rate: 95%
		
		

Figure 3: Changes of label determination rates for the first 3000 new added evaluations in MovieLens dataset experiment (two examples for each threshold of the label determination rate). Here, the goal is to keep the rate to be higher than 80%, 90% or 95% by re-computing the classifier when necessary. The proposed OSB method determines the degree to which the current classifier can be changed by adding a sequence of new evaluations, and determines when we should re-compute the classifier.

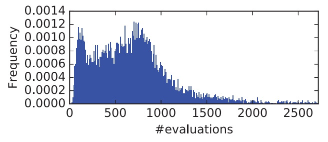
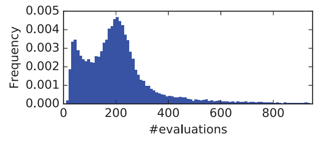
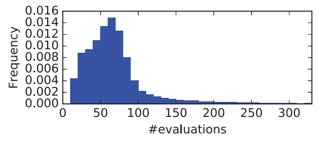
Threshold of the rate: 80%	Threshold of the rate: 90%	Threshold of the rate: 95%
Average: 707.39 ($\pm$520.76)	Average: 213.24 ($\pm$180.50)	Average: 68.41 ($\pm$67.33)
		

Figure 4: Number of evaluations added until the label determination rates become lower than the threshold. ("$\pm$": standard deviations)

we add these evaluations back in a random order by one of the following three operations: (*spot addition*) adding new spot evaluations in X, (*instance addition*) adding a new row of X, or (*feature addition*) adding a new column of X. See Appendix F for additional details.

First, we train $\boldsymbol{w}^*$ by solving (2). Then, whenever 10 evaluations are added to X in the order described above, we compute the label determination rate. If the rate is lower than the threshold (80%, 90% or 95%), then we discard the current $\boldsymbol{w}^*$ and re-compute $\boldsymbol{w}^*$ by solving (2) again (note that the rate becomes 100% just after re-computation). Since solving (2) is much more costly than computing the label determination rate by the proposed OSB, the frequency of re-computation of the classifier $\boldsymbol{w}^*$ would be the dominating cost. For each of the thresholds, we run the experiment with 15 different random seeds. We conducted the experiment with $\lambda = 1$ and no partial optimization.

The results are shown in Figures 3 and 4. The results suggest that, in this illustrative study, for a movie evaluation database with approximately 13 million evaluations, if we want to maintain the label determination rate to be higher than 80%, we only need to recompute the classifier after an average of 707 new evaluations have been added to the database. It is also interesting to note that there are variations in the interval between re-computations as shown in Figure 3. We conjecture that this occurs because the amount of the possible change of the classifier $\boldsymbol{w}^*$ depends on which evaluations are newly added to the database. An advantage of OSB is that it can provide a quantitative assurance concerning the degree to which the classifier can change. Such an assurance is difficult to obtain if we re-compute the classifier in a fixed interval.

5.2 Performance Evaluations

Next, we demonstrate the performances of OSB with larger datasets. In this section, first we show the tightness of the bounds measured by the label determination rate. Then we show comparisons of the computation times for the three settings: the warm-start method as an existing incremental learning method, the proposed bounds in §3 and the proposed bounds using the partial optimization (§4). Due to space limitations, we present a portion of our results in the main text. The complete experimental results are presented in Appendix G, including the results for five datasets, other λ's and evaluations for the difference of the model (9) as well as the label determination rate.

We used two benchmark datasets: `kdd-a` ($n =$8,407,752, $d =$20,216,830) and `url` ($n =$2,396,130, $d =$3,231,961) obtained from libsvm data repository (Chang and Lin 2011), where 80% of the instances are used as the training set and the rest are used as the test set. We considered three scenarios in Figure 1. In the spot modification scenario, we modified randomly chosen $|\mathcal{M}|$ elements in X for $|\mathcal{M}| \in \{1, 100, 10000\}$. In the instance modification scenario, we modified randomly chosen Δn rows of X for $\Delta n \in \{1, 10, 100\}$. In the feature modification scenario, we modified randomly chosen Δd columns of X for $\Delta d \in \{1, 10, 100\}$. We set $\lambda = 0.001$ (results with other λ's are

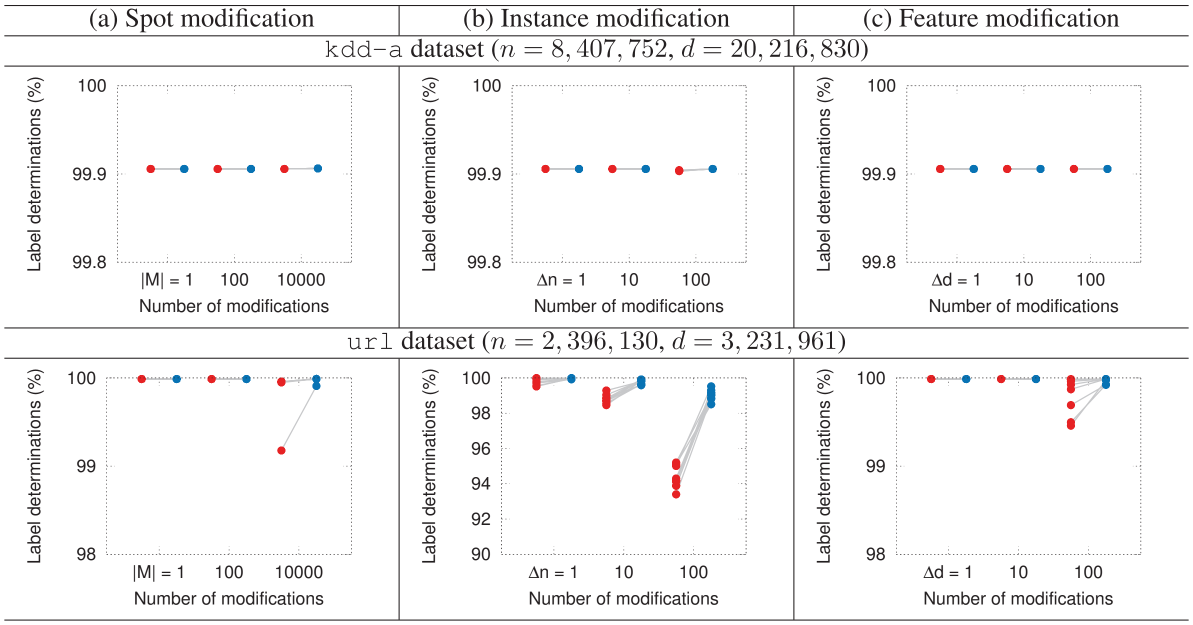


Figure 5: Label determination rates (%) for various data matrix modifications.

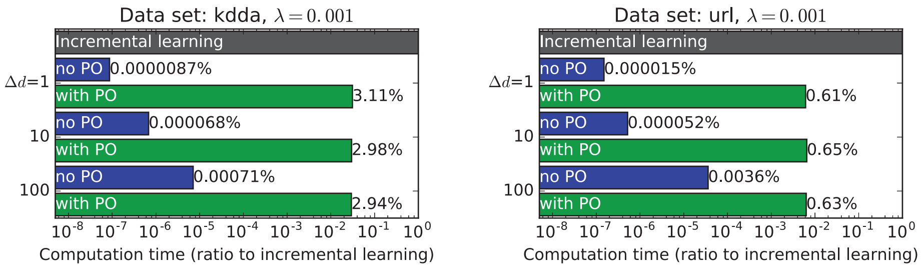


Figure 6: Ratio of the computation time of the proposed OSB to that of incremental learning (warm-start), on a log-scale for the feature modification scenario. (“PO”: partial optimization)

presented in Appendix G). For each condition, we run the experiment with 10 different random seeds.

Tightness of the Bounds The red points in Figure 5 indicate the performances of the bounds computed by the proposed method in §3. For `kdd-a` dataset, more than 99.9% of the test labels were determined using the bounds.We conjecture that, since this dataset is huge ($n > 8$ million and $d > 20$ million), a small data modification did not change the solution significantly, and these bounds could nicely capture this phenomenon. For `url` dataset, the label determination rates were slightly worse especially when 100 rows were modified in the instance modification scenario. We conjecture that this performance deterioration might happen when several influential row vectors were modified and the change in the solution was relatively large.

The blue points in Figure 5 show the performances of the bounds lifted by the partial optimization approach discussed in §4. The black lines connecting the red and blue points indicate the correspondence in 10 random trials. As described in Theorem 4.1, the bounds in blue points are always tighter than those in red points (higher values in the label determination rate indicate tighter bounds). For `kdd-a` dataset, the performances of the red and blue points are approximately the same because the original bounds (red points) were already sufficiently tight. For `url` dataset, partial optimizations seemed to work well. In particular, when the performances of the original bounds in red points were relatively poor (e.g., the $\Delta n = 100$ case), the improvements by partial optimization were significant.

Computation Times Figure 6 shows the computational costs of the proposed bound computation methods (with/without partial optimization) along with those of retraining with warm-start (an existing incremental learning) for feature modifications $\Delta d \in \{1, 10, 100\}$. The results for other settings are shown in Appendix G. The cost of the proposed bound computation without partial optimization is almost negligible, as compared to that for the warm-start. Using partial optimization, the cost is still much smaller. Similar results were obtained for the spot/instance modification scenarios.

6 Conclusions

In this paper, we present a method for quickly evaluating the data modification effect on the classifier. The proposed method provides bounds on the optimal solution with a cost proportional to the size of the data modification. The experimental results indicate that the bound computation method proposed in §3 is highly effective when the number of modified elements is much smaller than the total dataset size. In addition, the partial optimization approach is also effective, especially when the bounds in Theorem 3.1 are not good enough.

Acknowledgment

This work was partially supported by MEXT KAKENHI (17H00758, 16H06538), JST CREST (JPMJCR1302, JPMJCR1502; JPMJCR1302 includes AIP challenge program), RIKEN Center for Advanced Intelligence Project, and JST support program for starting up innovation-hub on materials research by information integration initiative.

References

Cauwenberghs, G., and Poggio, T. 2001. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems*, 409–415.

Chang, C.-C., and Lin, C.-J. 2011. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):27.

Chitrakar, R., and Huang, C. 2014. Selection of candidate support vectors in incremental svm for network intrusion detection. *computers & security* 45:231–241.

DeCoste, D., and Wagstaff, K. 2000. Alpha seeding for support vector machines. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 345–349.

El Ghaoui, L.; Viallon, V.; and Rabbani, T. 2012. Safe feature elimination for the lasso and sparse supervised learning problems. *Pacific Journal of Optimization* 8(4):667–698.

Gabel, M.; Keren, D.; and Schuster, A. 2015. Monitoring least squares models of distributed streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 319–328. ACM.

Gâlmeanu, H.; Sasu, L. M.; and Andonie, R. 2016. Incremental and decremental svm for regression. *International Journal of Computers Communications & Control* 11(6):755–775.

Gu, B.; Sheng, V. S.; Tay, K. Y.; Romano, W.; and Li, S. 2015a. Incremental support vector learning for ordinal regression. *IEEE Transactions on Neural networks and learning systems* 26(7):1403–1416.

Gu, B.; Sheng, V. S.; Wang, Z.; Ho, D.; Osman, S.; and Li, S. 2015b. Incremental learning for ν-support vector regression. *Neural Networks* 67:140–150.

Harper, F. M., and Konstan, J. A. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5(4).

Hiriart-Urruty, J.-B., and Lemarechal, C. 1993. *Convex Analysis and Minimization Algorithms II*. Springer-Verlag Berlin Heidelberg.

Karasuyama, M., and Takeuchi, I. 2010. Multiple incremental decremental learning of support vector machines. *IEEE Transactions on Neural Networks* 21:1048–1059.

Laskov, P.; Gehl, C.; Kruger, S.; and Muller, K. R. 2006. Incremental support vector learning: analysis, implementation and applications. *Journal of Machine Learning Research* 7:1909–1936.

Nakagawa, K.; Suzumura, S.; Karasuyama, M.; Tsuda, K.; and Takeuchi, I. 2016. Safe pattern pruning: An efficient approach for predictive pattern mining. In *22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD2016)*.

Ndiaye, E.; Fercoq, O.; Gramfort, A.; and Salmon, J. 2015. Gap safe screening rules for sparse multi-task and multi-class models. In *Advances in Neural Information Processing Systems*, 811–819.

Ogawa, K.; Suzuki, Y.; and Takeuchi, I. 2013. Safe screening of non-support vectors in pathwise svm computation. In *Proceedings of the 30th International Conference on Machine Learning*, 1382–1390.

Okumura, S.; Suzuki, Y.; and Takeuchi, I. 2015. Quick sensitivity analysis for incremental data modification and its application to leave-one-out cv in linear classification problems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 885–894.

Rockafellar, R. T. 1970. *Convex analysis*. Princeton university press.

Shibagaki, A.; Suzuki, Y.; Karasuyama, M.; and Takeuchi, I. 2015. Regularization path of cross-validation error lower bounds. In *Advances in Neural Information Processing Systems*, 1666–1674.

Shibagaki, A.; Karasuyama, M.; Hatano, K.; and Takeuchi, I. 2016. Simultaneous safe screening of features and samples in doubly sparse modeling. In *Proceedings of the 33rd International Conference on Machine Learning*, 1577–1586.

Takada, T.; Hanada, H.; Yamada, Y.; Sakuma, J.; and Takeuchi, I. 2016. Secure approximation guarantee for cryptographically private empirical risk minimization. In *The 8th Asian Conference on Machine Learning*, 126–141.

Tsai, C.-H.; Lin, C.-Y.; and Lin, C.-J. 2014. Incremental and decremental training for linear classification. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 343–352. ACM.

Zhu, L.; Pang, S.; Chen, G.; and Sarrafzadeh, A. 2012. Class imbalance robust incremental lpsvm for data streams learning. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 1–8. IEEE.