EXACT INCREMENTAL AND DECREMENTAL LEARNING FOR LS-SVM

Wei-Han Lee[1], *Bong Jun Ko*[1], *Shiqiang Wang*[1], *Changchang Liu*[1], *Kin K. Leung*[2]

[1]IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
[2]Imperial College London, UK

ABSTRACT

In this paper, we present a novel incremental and decremental learning method for the least-squares support vector machine (LS-SVM). The goal is to adapt a pre-trained model to changes in the training dataset, without retraining the model on all the data, where the changes can include addition and deletion of data samples. We propose a *provably exact* method where the updated model is exactly the same as a model trained from scratch using the entire (updated) training dataset. Our proposed method only requires access to the updated data samples, the previous model parameters, and a unique, fixed-size matrix that quantifies the effect of the previous training dataset. Our approach can significantly reduce the storage requirement of model updating, preserve the privacy of unchanged training samples without loss of model accuracy, and enhance the computational efficiency. Experiments on real-world image dataset validate the effectiveness of our proposed method.

***Index Terms*—** Least-squares support vector machine (LS-SVM), machine learning, model updating

1. INTRODUCTION

Machine learning models in image and vision applications usually need to be trained with a large amount of data. In many practical scenarios, the samples in the training dataset can change over time, due to the addition of new data samples (e.g., those that have been collected and labeled recently) and removal of existing data samples (e.g., those that are too noisy or incorrectly labeled). Retraining the model from scratch every time when there is a change in the training dataset is too time-consuming. It is more efficient to update the model by including or excluding the influence of specific data samples, which is known as *incremental and decremental learning*.

Least-squares support vector machine (LS-SVM) has been broadly applied in various machine learning tasks and image/vision applications [1–3]. The benefit of LS-SVM is that there exists an analytical solution to the optimal model parameters for a given training dataset. However, it is still computationally intensive to recompute the model parameters on the entire dataset when only a few samples in the dataset change.

In this paper, we propose a novel approach that uses analytical expressions to *exactly update* the LS-SVM model using the added and deleted data samples only. Our approach utilizes an auxiliary matrix that is defined to capture some essential information in the previous training dataset, which has a much smaller size than the original dataset and does *not* grow with the size of the dataset. The updated model is *provably the same as* a model trained on the entire dataset. Hence, the proposed model updating process is much faster and requires less storage than retraining the model from scratch while retaining the same model accuracy. In addition, because our model updating algorithm does not require knowledge of those raw data samples that remain unchanged (i.e., not added or deleted), it preserves the privacy of unchanged data samples in a distributed system setting.

1.1. Motivating Application Scenarios

We first outline some application scenarios of our work.

In a *cloud computing system* [4], an end user may need to retrain his/her model (that is originally trained by the service provider) frequently when dealing with concept drift (e.g., authentication system) or streaming data (e.g., autonomous driving). To reduce the communication cost between the service provider and the end user, it is more efficient for the end user to update his/her model on the local device, which also protects the privacy of the newly arrived data since they are no longer required to be uploaded to the cloud.

In a *distributed/federated learning system* [5], one agent computes its gradients based on its local data and then transmits the gradients to the next agent for model updating. However, this gradient updating process usually requires a number of rounds/iterations to converge to a global solution. By leveraging our method, each agent can update the model directly based on his/her local data and then the updated model (instead of the updated gradients) are transmitted to the next agent for model updating, which only requires one round of computation to obtain the final model.

In *k-fold cross-validation* [6], the original data is randomly partitioned into k folds, among which $k-1$ folds are used as training dataset and the remaining one fold as test/validation set. This process is repeated for k times until every fold has been used as a test set. Using our decremental learning method, one can first train the model with the entire dataset, and then can simply remove the influence of the i-th fold to obtain the new model for i-th round of validation, which is validated/tested with that i-th fold. This significantly reduces the time needed for the cross-validation without any loss of validation accuracy.

1.2. Related Work

Support vector machine (SVM) and LS-SVM have been widely used in image and vision applications such as image classification [2, 7, 8], object tracking [3], visual quality prediction [9], and splicing detection [10].

For SVM, model updating methods have been proposed in [11–14]. However, all of these methods are heuristic without any theoretical guarantees, and they do not provide decremental learning methods. Another type of incremental learning algorithm is based on ensemble learning where new classifiers are trained for new batches of data samples and the decision is made by combining all the classifiers [15–17]. However, these methods do not provide exact incremental learning, while decremental learning is not available either. Exact incremental and decremental methods are proposed in [14, 18], but the algorithms need to store and access all the previous training data, although the complexity of model updating is lower than retraining on all the data.

LS-SVM uses the quadratic loss instead of the hinge loss in (regular) SVM, which can significantly reduce the training complexity. Exact incremental learning algorithms for LS-SVM are proposed in [19, 20], but they do not support decremental learning and require storing and accessing all the previous training data. In [21], an approximate linear dependence condition is used to decide whether to keep each training data sample, which is a heuristic method that does not bound the amount of training data that needs to be kept. The budget online LS-SVM algorithm in [22] does not require keeping all the training data, but the method is heuristic and previously learned knowledge may not be preserved after some number of new data samples arrive.

In summary, to our knowledge, there does *not* exist an approach of exact model updating for LS-SVM which 1) supports both incremental and decremental learning, and 2) does not require storing and accessing all the training data. We propose such an approach satisfying both 1) and 2) in this paper.

2. LEAST SQUARES SUPPORT VECTOR MACHINE

The goal of LS-SVM is to learn a model that assigns the correct label to a data sample with respect to minimizing the squared error in the assigned labels. Formally, it learns a function $f : \mathbb{R}^M \to \mathbb{R}$ which maps each data sample $\boldsymbol{x}$ to a label y with the optimal classifier [23]:

$$\boldsymbol{w}^* = \mathrm{argmin}_{\boldsymbol{w}\in\mathbb{R}^J} \rho\|\boldsymbol{w}\|^2 + \sum_{n=1}^{N} (\boldsymbol{w}^T \vec{\phi}(\boldsymbol{x}_n) - y_n)^2 \quad (1)$$

where N is the number of data samples, $\boldsymbol{x}_n \in \mathbb{R}^M$ represents the n-th data sample, M is the dimension of each data sample. The vector $\boldsymbol{w} \in \mathbb{R}^M$ represents a possible model and $\boldsymbol{w}^*$ is the optimal model that satisfies (1). For a training dataset containing N samples, we define an $M \times N$ matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$ and a vector $\boldsymbol{y} = [y_1, y_2, \cdots, y_N]$. The function $\vec{\phi}(\boldsymbol{x_i})$ is a kernel function that maps the original data $\boldsymbol{x}_i$ into a (possibly) higher-dimensional space (with J dimensions) for linearly separating the data samples [23]. We also define $\boldsymbol{\Phi} = \boldsymbol{\Phi}(\boldsymbol{X}) = [\vec{\phi}(\boldsymbol{x}_1), \vec{\phi}(\boldsymbol{x}_2), \cdots, \vec{\phi}(\boldsymbol{x}_N)]$. The objective function in (1) has an optimal solution given by [23]:

$$\boldsymbol{w}^* = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}_N]^{-1}\boldsymbol{y} = [\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T]^{-1}\boldsymbol{\Phi}\boldsymbol{y} \quad (2)$$

where $\boldsymbol{I}_k$ denotes the identity matrix of size k (for given k).

3. PROPOSED MODEL UPDATING APPROACH

We present our proposed exact incremental and decremental learning approach as follows. We first consider the general case where we need to update the LS-SVM model by adding N_a data samples $(\boldsymbol{X}_a, \boldsymbol{y}_a)$ to the training dataset and removing N_r data samples $(\boldsymbol{X}_r, \boldsymbol{y}_r)$ from the training dataset at the same time. Our main result is in Theorem 1, according to which we can update the model exactly without the need of accessing the unchanged data samples.

Theorem 1. *For a given model*

$$\boldsymbol{w} = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}_N]^{-1}\boldsymbol{y} \quad (3)$$

and auxiliary matrix

$$\boldsymbol{C} = \boldsymbol{\Phi}[\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\boldsymbol{I}_N]^{-1}\boldsymbol{\Phi}^T, \quad (4)$$

when adding new training data $(\boldsymbol{X}_a, \boldsymbol{y}_a)$ and removing existing training data $(\boldsymbol{X}_r, \boldsymbol{y}_r)$, we can compute the new values of $\boldsymbol{w}$ and $\boldsymbol{C}$ using

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c \left(\rho\boldsymbol{I} - \boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\right)^{-1} \left(\boldsymbol{\Phi}_c'^T\boldsymbol{w} - \boldsymbol{y}_c\right) \quad (5)$$

$$\boldsymbol{C}_{\text{new}} = \boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c \left(\rho\boldsymbol{I} - \boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_c\right)^{-1} \boldsymbol{\Phi}_c'^T\left(\boldsymbol{C} - \boldsymbol{I}_J\right) \quad (6)$$

where we define $\boldsymbol{\Phi}_c = (\boldsymbol{\Phi}_a, \boldsymbol{\Phi}_r)$, $\boldsymbol{\Phi}_c' = (\boldsymbol{\Phi}_a, -\boldsymbol{\Phi}_r)$, and $\boldsymbol{y}_c = (\boldsymbol{y}_a, -\boldsymbol{y}_r)$.

Proof. For the updated training dataset, we get $\boldsymbol{\Phi}_{\text{new}} = (\boldsymbol{\Phi}, \boldsymbol{\Phi}_a|\boldsymbol{\Phi}_r)$ and $\boldsymbol{y}_{\text{new}} = (\boldsymbol{y}, \boldsymbol{y}_a|\boldsymbol{y}_r)$, where $(\boldsymbol{Z}_1|\boldsymbol{Z}_2)$ denotes removing the columns (of a matrix) or elements (of a vector) in $\boldsymbol{Z}_2$ from $\boldsymbol{Z}_1$. Using (2) and (3), we can compute the new model $\boldsymbol{w}_{\text{new}}$ as

$$\begin{aligned}
\boldsymbol{w}_{\text{new}} &= \boldsymbol{\Phi}_{\text{new}}[\boldsymbol{\Phi}_{\text{new}}^T\boldsymbol{\Phi}_{\text{new}} + \rho\boldsymbol{I}_{N+N_a-N_r}]^{-1}\boldsymbol{y}_{\text{new}} \\
&= [\rho\boldsymbol{I}_J + \boldsymbol{\Phi}_{\text{new}}\boldsymbol{\Phi}_{\text{new}}^T]^{-1}\boldsymbol{\Phi}_{\text{new}}\boldsymbol{y}_{\text{new}} \\
&= \left[\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_a\boldsymbol{\Phi}_a^T - \boldsymbol{\Phi}_r\boldsymbol{\Phi}_r^T\right]^{-1}[\boldsymbol{\Phi}\boldsymbol{y} + \boldsymbol{\Phi}_a\boldsymbol{y}_a - \boldsymbol{\Phi}_r\boldsymbol{y}_r] \\
&= \left[\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_c\boldsymbol{\Phi}_c'^T\right]^{-1}[\boldsymbol{\Phi}\boldsymbol{y} + \boldsymbol{\Phi}_c\boldsymbol{y}_c] \quad (7)
\end{aligned}$$

where we note that $\boldsymbol{\Phi}_c = (\boldsymbol{\Phi}_a, \boldsymbol{\Phi}_r)$, $\boldsymbol{\Phi}_c' = (\boldsymbol{\Phi}_a, -\boldsymbol{\Phi}_r)$ and $\boldsymbol{y}_c = (\boldsymbol{y}_a, -\boldsymbol{y}_r)$ by definition.

According to the Woodbury matrix identity [24], we have $(\boldsymbol{A}+\boldsymbol{U}\boldsymbol{B}\boldsymbol{V})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}\left(\boldsymbol{I}+\boldsymbol{B}\boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}\right)\boldsymbol{B}\boldsymbol{V}\boldsymbol{A}^{-1}$. We define $\boldsymbol{A} = \rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T$, $\boldsymbol{B} = 1$, $\boldsymbol{U} = \boldsymbol{\Phi}_c$, $\boldsymbol{V} = \boldsymbol{\Phi}_c'^T$, and $\boldsymbol{\Psi} = \boldsymbol{I} + \boldsymbol{\Phi}_c'^T\boldsymbol{A}^{-1}\boldsymbol{\Phi}_c$. From (7), we can obtain the following updating process for $\boldsymbol{w}$:

$$\boldsymbol{w}_{\text{new}} = \left(\boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T\boldsymbol{A}^{-1}\right)[\boldsymbol{\Phi}\boldsymbol{y} + \boldsymbol{\Phi}_c\boldsymbol{y}_c]$$

$$
\begin{aligned}
&= \boldsymbol{w} + \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{y}_c - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T\boldsymbol{w}\\
&\quad - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T\boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{y}_c\\
&= \boldsymbol{w} + \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{y}_c - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c^T\boldsymbol{w}\\
&= \boldsymbol{w} - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\left(\boldsymbol{\Phi}_c'^T\boldsymbol{w} - \boldsymbol{y}_c\right)\\
&= \boldsymbol{w} + (\boldsymbol{C}-\boldsymbol{I}_J)\,\boldsymbol{\Phi}_c\left(\rho\boldsymbol{I}+\boldsymbol{\Phi}_c'^T(\boldsymbol{C}-\boldsymbol{I}_J)\,\boldsymbol{\Phi}_c\right)^{-1}\left(\boldsymbol{\Phi}_c'^T\boldsymbol{w}-\boldsymbol{y}_c\right).
\end{aligned}
$$

The last equality holds because

$$
\begin{aligned}
\rho\boldsymbol{A}^{-1} + \boldsymbol{C} &= \rho\left[\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]^{-1} + \boldsymbol{\Phi}\left[\rho\boldsymbol{I}_N + \boldsymbol{\Phi}^T\boldsymbol{\Phi}\right]^{-1}\boldsymbol{\Phi}^T\\
&= \rho\left[\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]^{-1} + \left[\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^T\\
&= \left[\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right]^{-1}\left[\rho\boldsymbol{I}_J + \boldsymbol{\Phi}\boldsymbol{\Phi}^T\right] = \boldsymbol{I}.
\end{aligned}
$$

Similarly, we can compute $\boldsymbol{C}_{\text{new}}$ as

$$
\begin{aligned}
\boldsymbol{C}_{\text{new}} &= \boldsymbol{\Phi}_{\text{new}}[\boldsymbol{\Phi}_{\text{new}}^T\boldsymbol{\Phi}_{\text{new}} + \rho\boldsymbol{I}_{N+N_a-N_r}]^{-1}\boldsymbol{\Phi}_{\text{new}}^T\\
&= [\rho\boldsymbol{I}_J + \boldsymbol{\Phi}_{\text{new}}\boldsymbol{\Phi}_{\text{new}}^T]^{-1}\boldsymbol{\Phi}_{\text{new}}\boldsymbol{\Phi}_{\text{new}}^T\\
&= [\boldsymbol{A} + \boldsymbol{\Phi}_a\boldsymbol{\Phi}_a^T - \boldsymbol{\Phi}_r\boldsymbol{\Phi}_r^T]^{-1}[\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_a\boldsymbol{\Phi}_a^T - \boldsymbol{\Phi}_r\boldsymbol{\Phi}_r^T]\\
&= [\boldsymbol{A} + \boldsymbol{\Phi}_c\boldsymbol{\Phi}_c'^T]^{-1}[\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_c\boldsymbol{\Phi}_c'^T]\\
&= \left(\boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T\boldsymbol{A}^{-1}\right)[\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Phi}_c\boldsymbol{\Phi}_c'^T]\\
&= \boldsymbol{C} + \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Phi}_c'^T - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T\boldsymbol{C}\\
&\quad - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T\boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Phi}_c'^T\\
&= \boldsymbol{C} + \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T\boldsymbol{C}\\
&= \boldsymbol{C} - \boldsymbol{A}^{-1}\boldsymbol{\Phi}_c\boldsymbol{\Psi}^{-1}\boldsymbol{\Phi}_c'^T(\boldsymbol{C} - \boldsymbol{I}_J)\\
&= \boldsymbol{C} + (\boldsymbol{C}-\boldsymbol{I}_J)\,\boldsymbol{\Phi}_c\left(\rho\boldsymbol{I}+\boldsymbol{\Phi}_c'^T(\boldsymbol{C}-\boldsymbol{I}_J)\,\boldsymbol{\Phi}_c\right)^{-1}\boldsymbol{\Phi}_c'^T(\boldsymbol{C}-\boldsymbol{I}_J).
\end{aligned}
$$

Algorithm: Our algorithm works as follows. Based on Theorem 1, when the model is trained for the first time, our algorithm computes the model $\boldsymbol{w}$ and the auxiliary matrix $\boldsymbol{C}$ using the kernel functions computed on the entire training dataset ($\boldsymbol{\Phi}$), according to (3) and (4) respectively. Note that (3) is the same as (2) and hence it is optimal. When data samples are added or removed, we update $\boldsymbol{w}$ and $\boldsymbol{C}$ according to (5) and (6), respectively. The proof of Theorem 1 has shown that the new $\boldsymbol{w}$ and $\boldsymbol{C}$ are respectively equal to $\boldsymbol{w}$ and $\boldsymbol{C}$ computed on the entire updated dataset. By considering the complexity of matrix inversion, multiplication, and addition/subtraction, we can obtain that the complexity of this algorithm is $O(L^3 + JL^2 + J^2L + J^3)$, where L is defined as the total number of samples that are added and removed in the batch (referred to as the *batch size*), and J is the dimension of the kernel function output as previously defined.

For special cases where we only perform incremental or decremental learning (but not both), either for one sample or a batch of samples, we have the following corollaries that can be obtained directly from Theorem 1. The update equations for these special cases, which are given in the corollaries below, are simpler than the update equations (5) and (6) for the general case.

Corollary 1 (Incremental learning of a single data sample)**.** *We can update $\boldsymbol{w}$ and $\boldsymbol{C}$ to include the influence of a new training data sample $(\boldsymbol{x}_{N+1}, y_{N+1})$ using*

$$
\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + \frac{(\boldsymbol{C} - \boldsymbol{I}_J)\vec{\phi}(\boldsymbol{x}_{N+1})(\vec{\phi}(\boldsymbol{x}_{N+1})^T\boldsymbol{w} - y_{N+1})}{\vec{\phi}(\boldsymbol{x}_{N+1})^T\vec{\phi}(\boldsymbol{x}_{N+1}) + \rho - \vec{\phi}(\boldsymbol{x}_{N+1})^T\boldsymbol{C}\vec{\phi}(\boldsymbol{x}_{N+1})}
$$

$$
\boldsymbol{C}_{\text{new}} = \boldsymbol{C} + \frac{(\boldsymbol{C} - \boldsymbol{I}_J)\vec{\phi}(\boldsymbol{x}_{N+1})\vec{\phi}(\boldsymbol{x}_{N+1})^T(\boldsymbol{C} - \boldsymbol{I}_J)}{\vec{\phi}(\boldsymbol{x}_{N+1})^T\vec{\phi}(\boldsymbol{x}_{N+1}) + \rho - \vec{\phi}(\boldsymbol{x}_{N+1})^T\boldsymbol{C}\vec{\phi}(\boldsymbol{x}_{N+1})}.
$$

Corollary 2 (Decremental learning of a single data sample)**.** *We can update $\boldsymbol{w}$ and $\boldsymbol{C}$ to remove the influence of an existing training data sample $(\boldsymbol{x}_r, y_r)$ using*

$$
\boldsymbol{w}_{\text{new}} = \boldsymbol{w} - \frac{(\boldsymbol{C} - \boldsymbol{I}_J)\vec{\phi}(\boldsymbol{x_r})\left(\vec{\phi}(\boldsymbol{x_r})^T\boldsymbol{w} - y_r\right)}{-\vec{\phi}(\boldsymbol{x_r})^T\vec{\phi}(\boldsymbol{x_r}) + \rho + \vec{\phi}(\boldsymbol{x_r})^T\boldsymbol{C}\vec{\phi}(\boldsymbol{x_r})}
$$

$$
\boldsymbol{C}_{\text{new}} = \boldsymbol{C} - \frac{(\boldsymbol{C} - \boldsymbol{I}_J)\vec{\phi}(\boldsymbol{x_r})\vec{\phi}(\boldsymbol{x_r})^T(\boldsymbol{C} - \boldsymbol{I}_J)}{-\vec{\phi}(\boldsymbol{x_r})^T\vec{\phi}(\boldsymbol{x_r}) + \rho + \vec{\phi}(\boldsymbol{x_r})^T\boldsymbol{C}\vec{\phi}(\boldsymbol{x_r})}.
$$

Corollary 3 (Incremental learning of a batch of data samples)**.** *We can update $\boldsymbol{w}$ and $\boldsymbol{C}$ to include the influence of a batch of new training data samples $(\boldsymbol{X}_a, \boldsymbol{y}_a)$ using*

$$
\boldsymbol{w}_{\text{new}} = \boldsymbol{w} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_a\left(\rho\boldsymbol{I} - \boldsymbol{\Phi}_a^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_a\right)^{-1}\left(\boldsymbol{\Phi}_a^T\boldsymbol{w} - \boldsymbol{y}_a\right)
$$

$$
\boldsymbol{C}_{\text{new}} = \boldsymbol{C} + (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_a\left(\rho\boldsymbol{I} - \boldsymbol{\Phi}_a^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_a\right)^{-1}\boldsymbol{\Phi}_a^T(\boldsymbol{C} - \boldsymbol{I}_J)
$$

where $\boldsymbol{\Phi}_a = \boldsymbol{\Phi}(\boldsymbol{X}_a)$.

Corollary 4 (Decremental learning of a batch of data samples)**.** *We can update $\boldsymbol{w}$ and $\boldsymbol{C}$ to remove the influence of a batch of existing training data samples $(\boldsymbol{X}_r, \boldsymbol{y}_r)$ using*

$$
\boldsymbol{w}_{\text{new}} = \boldsymbol{w} - (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_r\left(\rho\boldsymbol{I} + \boldsymbol{\Phi}_r^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_r\right)^{-1}\left(\boldsymbol{\Phi}_r^T\boldsymbol{w} - \boldsymbol{y}_r\right)
$$

$$
\boldsymbol{C}_{\text{new}} = \boldsymbol{C} - (\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_r\left(\rho\boldsymbol{I} + \boldsymbol{\Phi}_r^T(\boldsymbol{C} - \boldsymbol{I}_J)\boldsymbol{\Phi}_r\right)^{-1}\boldsymbol{\Phi}_r^T(\boldsymbol{C} - \boldsymbol{I}_J)
$$

where $\boldsymbol{\Phi}_r = \boldsymbol{\Phi}(\boldsymbol{X}_r)$.

4. EXPERIMENTATION RESULTS

The performance of our proposed model updating method is evaluated using experiments. We use the MNIST dataset [25] that includes images of handwritten digits from 0 to 9, which contains a training dataset of $60,000$ samples and a test dataset of $10,000$ samples. We train an LS-SVM to classify even vs. odd digits.

We first consider the incremental learning process where we separate the training data into five groups according to their labels (digits in the MNIST dataset), i.e., 0&1, 2&3, 4&5, 6&7 and 8&9. The initial model is trained with digits 0 and 1. We sequentially add each group of data to update the model using our proposed approach, to evaluate the scenario where data in different subcategories are added over time. The model accuracy is evaluated on the entire MNIST test dataset which includes all the digits. From Fig. 1(a), we observe that the model accuracy generally increases after adding a new group of data, which shows that our method does not forget the previously learned knowledge. The minor decrease of accuracy after adding digits 4 and 5 is due to different similarities between unlearned digits and learned digits.

To investigate the effectiveness of our proposed decremental learning method, we assign 50% of the original training data with randomly assigned erroneous labels based on which we train an initial model. Then, we update the model by gradually removing the incorrectly-labeled training data. From Figure 1(b), we observe that the accuracy increases as more incorrectly-labeled training data samples are removed.

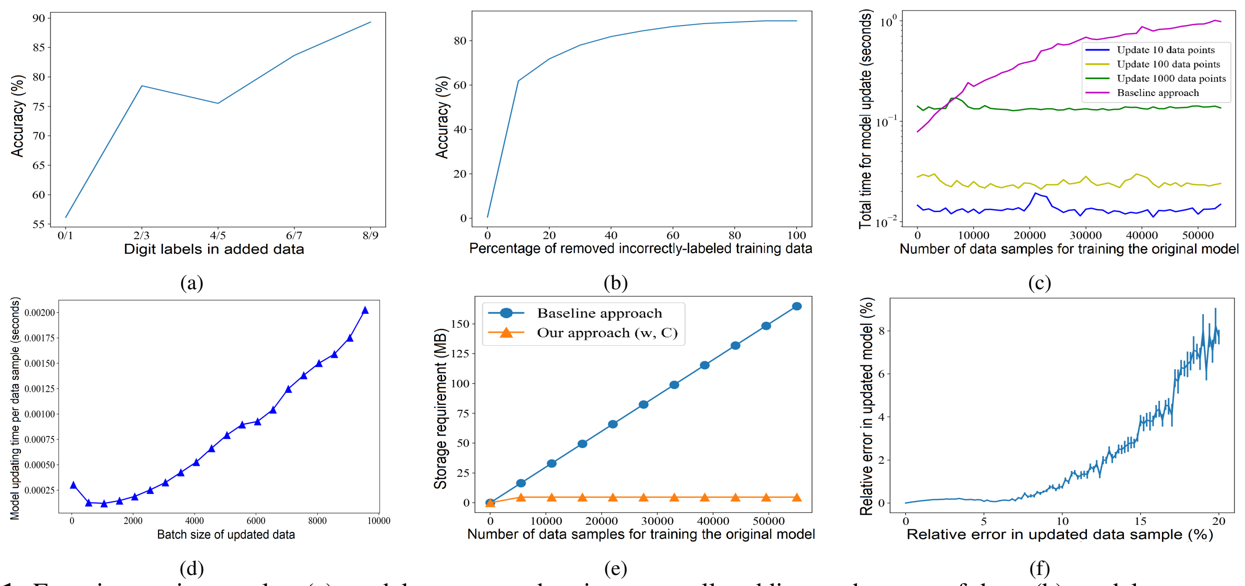

Fig. 1. Experimentation results: (a) model accuracy when incrementally adding each group of data; (b) model accuracy after removing different percentages of incorrectly-labeled training data; (c) total time for model update/retraining; (d) normalized model updating time per data sample (total updating time/batch size) of our proposed method; (e) storage requirement; (f) error of updated model using our proposed method caused by inaccuracies in removed training data samples.

We then compare the time needed for incremental learning (for one batch with a given size) with the time needed in the baseline approach that retrains the model from scratch. All the time measurements are collected on a MacBook Pro with 2.6 GHz Intel Core i7, 32 GB memory. We use a given amount (up to $55,000$) of training samples to train the original model. Then, we incrementally update the model with a given batch size of new data chosen from the remaining $5,000$ training samples in the MNIST dataset. Note that the computational complexity of incremental and decremental model update is the same, so we only focus on incremental update here. For the baseline approach, we consider a batch size of 10 new data samples. The results are shown in Fig. 1(c), where we see that our proposed approach significantly outperforms the baseline in most cases (note the logarithmic scale in the y-axis), confirming the benefit of our approach.

As discussed in Section 3, the complexity of model updating is cubic of the batch size. To illustrate this behavior, we plot the batch-size normalized running time in Fig. 1(d). We see that the model updating process is the most efficient when the batch size is around $1,000$.[1] As the model updating is exact, updating a large batch is equivalent to updating multiple small batches in a sequence. The latter approach can significantly reduce the model updating time for large batches.

The baseline approach needs to store all the data used to train the original model, in order to retrain from scratch. This is in contrast to our approach which only needs to store the vector $\boldsymbol{w}$ and matrix $\boldsymbol{C}$, which significantly reduces the storage requirement, as shown in Fig. 1(e).

In scenarios where the training data samples that need to be removed are not available, the proposed model updating method may still work if estimations of such data samples exist. Fig. 1(f) considers removing the influence of one data sample and shows the relative error in the updated model parameter $\boldsymbol{w}_{\text{new}}$ when there exists a certain relative error in the data sample to be removed. We see that when the error in the data sample is below 10%, the resulting error in the model is less than 1%, which is negligible.

5. CONCLUSION

We have proposed a model updating process for LS-SVM. By leveraging a unique auxiliary matrix, the proposed approach can update the model incrementally or decrementally without direct access to the previous training data. Our proposed method can achieve exact model updating while protecting the privacy of non-updated training data. The effectiveness of our approach has been confirmed through experiments.

6. ACKNOWLEDGEMENT

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

[1]The higher updating time with small batch size (less than about 1,000) is due to the overhead of the repetitive function calls to perform many such updates. This can be avoided if they can be executed in parallel.

7. REFERENCES

[1] Johan AK Suykens, Tony Van Gestel, and Jos De Brabanter, *Least Squares Support Vector Machines*, World Scientific, 2002.

[2] MN Vimalkumar, K Helenprabha, and A Surendar, “Classification of mammographic image abnormalities based on EMO and LS-SVM techniques,” *Research Journal of Biotechnology*, vol. 12, no. 1, pp. 35–40, 2017.

[3] Yuefang Gao, Xin Shan, Zexi Hu, Dong Wang, Ya Li, and Xuhong Tian, “Extended compressed tracking via random projection based on MSERs and online LS-SVM learning,” *Pattern Recognition*, vol. 59, pp. 245–254, 2016.

[4] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al., “A view of cloud computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al., “Communication-efficient learning of deep networks from decentralized data,” *arXiv preprint arXiv:1602.05629*, 2016.

[6] Ron Kohavi et al., “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *IJCAI*, 1995, vol. 14, pp. 1137–1145.

[7] Mayank Arya Chandra and SS Bedi, “Survey on SVM and their application in image classification,” *International Journal of Information Technology*, pp. 1–11, 2018.

[8] Min-Kook Choi, Hyun-Gyu Lee, and Sang-Chul Lee, “Weighted SVM with classification uncertainty for small training samples,” in *IEEE ICIP*, 2016, pp. 4438–4442.

[9] Ching-Ti Lin, Tsung-Jung Liu, and Kuan-Hsien Liu, “Visual quality prediction on distorted stereoscopic images,” in *IEEE ICIP*, 2017, pp. 3480–3484.

[10] Thales Pomari, Guillherme Ruppert, Edmar Rezende, Anderson Rocha, and Tiago Carvalho, “Image splicing detection through illumination inconsistencies and deep learning,” in *IEEE ICIP*, 2018, pp. 3788–3792.

[11] N Syed, H Liu, and K Sung, “Incremental learning with support vector machines,” in *International Joint Conference on Artificial Intelligence*, 1999.

[12] Stefan Ruping, “Incremental learning with support vector machines,” in *IEEE International Conference on Data Mining*, 2001, pp. 641–642.

[13] Carlotta Domeniconi and Dimitrios Gunopulos, “Incremental support vector machine construction,” in *IEEE International Conference on Data Mining*, 2001, pp. 589–592.

[14] Gert Cauwenberghs and Tomaso Poggio, “Incremental and decremental support vector machine learning,” in *Advances in Neural Information Processing Systems*, 2001, pp. 409–415.

[15] Piyabute Fuangkhon and Thitipong Tanprasert, “An incremental learning algorithm for supervised neural network with contour preserving classification,” in *ECTI-CON*, 2009, vol. 2, pp. 740–743.

[16] Yi-Min Wen and Bao-Liang Lu, “Incremental learning of support vector machines by classifier combining,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2007, pp. 904–911.

[17] Youlu Xing, Furao Shen, Chaomin Luo, and Jinxi Zhao, “L3-SVM: a lifelong learning method for SVM,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015.

[18] Christopher P Diehl and Gert Cauwenberghs, “SVM incremental learning, adaptation and optimization,” in *International Joint Conference on Neural Networks*, 2003, vol. 4, pp. 2685–2690.

[19] Hoi-Ming Chi and Okan K Ersoy, “Recursive update algorithm for least squares support vector machines,” *Neural Processing Letters*, vol. 17, no. 2, pp. 165–173, 2003.

[20] Zhifeng Hao, Shu Yu, Xiaowei Yang, Feng Zhao, Rong Hu, and Yanchun Liang, “Online LS-SVM learning for classification problems based on incremental chunk,” in *International Symposium on Neural Networks*, 2004, pp. 558–564.

[21] Yaakov Engel, Shie Mannor, and Ron Meir, “The kernel recursive least squares algorithm,” *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[22] Ling Jian, Shuqian Shen, Jundong Li, Xijun Liang, and Lei Li, “Budget online learning algorithm for least squares SVM,” *IEEE Trans. on Neural Networks and Learning Systems*, vol. 28, no. 9, pp. 2076–2087, 2017.

[23] Sun Yuan Kung, *Kernel Methods and Machine Learning*, Cambridge University Press, 2014.

[24] Max A Woodbury, “Inverting modified matrices,” *Memorandum report*, vol. 42, no. 106, pp. 336, 1950.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.