FSVM: Federated Support Vector Machines for Smart City

Lichuan Ma[1(✉)], Lizhen Tang[1], Longxiang Gao[2], Qingqi Pei[1], and Ming Ding[3]

[1] Shannxi Key Laboratory of Blockchain and Secure Computing, Xidian University, Xi'an, China
lcma@xidian.edu.cn, lztang@stu.xidian.edu.cn, qqpei@mail.xidian.edu.cn

[2] Qilu University, Jinan, China
gaolx@sdas.org

[3] Information Privacy and Security Group, Data61, Commonwealth Scientific and Industrial Research Organization, Silicon Valley, Australia
ming.ding@data61.csiro.au

Abstract. By putting digital technology and vast volume of data together, smart city becomes an emerging city paradigm for intelligent city management and operation. As one of the most popular artificial intelligent algorithms, support vector machines (SVMs) have been widely adopted for classification in various smart city applications. Due to the explosion of data and rigorous privacy requirements, an SVM classifier needs to be trained in a distributed and privacy-preserving manner. To achieve this, a federated SVM (FSVM) scheme is proposed to collaboratively and privately train an SVM classifier by combining the alternating direction method of multipliers (ADMM) with secret sharing. Specifically, the FSVM consists of FSVM-C and FSVM-S to deal with two cases of data partitioning by examples and features, respectively. By implementing the FSVM scheme on the real-word dataset MNIST, the efficiency and effectiveness of both FSVM-S and FSVM-C are verified by comprehensive experimental results.

Keywords: Federated Support Vector Machines · Privacy Preserving · ADMM

1 Introduction

Benefiting from the prevalence of cloud computing, IoT, and artificial intelligence (AI), smart city has become an emerging future city paradigm for intelligent city management and operation by putting vast volume of data and digital technology together [1]. For making better decisions, various AI algorithms, such as machine learning and deep learning algorithms, are adopted. As one of the most widely utilized AI algorithms, support vector machines (SVMs) have gained popularity in many typical smart city applications, such as fraud detection [2], disease diagnosis [3], and cyber security situation prediction [4].

S. Yu et al. (Eds.): TridentCom 2022, LNICST 489, pp. 149–167, 2023.
https://doi.org/10.1007/978-3-031-33458-0_11

Generally, SVM-based smart city applications rely on a centralized cloud server for gathering training data and obtaining the classifier by solving a convex optimization problem. However, due to the pervasive implementation of IoTs, the amount of data to be gathered explode and thus heavy burdens on bandwidth have been introduced for gathering these data. To tackle this problem, one new computing paradigm, namely edge computing, comes up and many smart city applications have been built on this paradigm [5]. Here, raw training data are distributed across different entities and the derivation of an SVM classifier needs their collaboration. However, privacy concerns might hinder such kind of collaboration as locally held data may include sensitive information about the holders [6]. Situations become much worse with the enforcement of a series of privacy laws including the General Data Protection Regulation by the European Union and the California Consumer Privacy Act by the US [7]. As a result, SVM classifiers should be derived in a distributed and privacy-preserving manner for current smart city applications.

For deriving SVM classifiers in a distributed manner, the authors of [8] present a method based on the alternating direction method of multipliers (ADMM). In this solution, the centralized SVM problem is split into a set of convex subproblems (one per participant) with consensus constraints on the classifier parameters. Via this solution, there is no need to exchange locally held training data and the privacy problem is addressed to some extent. Unfortunately, the method in [8] only considers the case of data partitioning by examples and it restricts the wide adoption of this method as there are many application scenarios where data are partitioned by features [9]. In addition, according to [10] and [11], the intermediate states to be exchanged when running ADMM still reveal sensitive information about the locally held training data.

To further protect the intermediate states, two types of solutions are offered by [10] and [11]. In [10], controllable noises are added to the intermediated sates such that the requirements of differential privacy are satisfied. As additional noises are introduced, the privacy goal is achieved with sacrificing accuracy. Even though the tradeoff between privacy and accuracy is explored by the authors, the speed for converging to the optimal classifier would be always decreased. By proposing a new ADMM algorithm that allows time-varying penalty matrices, the partially homomorphic encryption scheme is incorporated to protect the intermediate states in [11]. Here, the accuracy is not sacrificed but heavy computation overhead is introduced. Moreover, all the methods in [8,10] and [11] only consider the case of data partitioning by examples and it makes their adoptions to be restricted.

Thus in this paper, to conquer the drawbacks of existing works and derive SVM classifiers in a distributed and privacy-preserving manner, the federated SVM (FSVM) scheme is proposed in this paper. Here, the term "federated" is borrowed from federated learning where multiple data holders collaborate in solving a machine learning problem based on locally stored raw data without compromising data privacy [12]. The contributions of our work are summarized as following.

- Firstly, we present the system model for the proposed FSVM scheme where the general optimization problems for the cases of data partitioning by examples and features are offered for deriving SVM classifiers.
- Then, for the case of data partitioning by examples, the FSVM-C is proposed by incorporating secret sharing with the new ADMM in [11] that allows time-varying matrices. By doing this, the privacy-preserving goal can be achieved in a more efficient manner without introducing a trusted third-party for generating keys.
- After that, the FSVM-S is put forward to deal with the case of data partitioning by features. Here, we derive the closed form of the ADMM iterations that converge to the optimal solution to construct SVM classifiers. The Shamir secret sharing is introduced for protecting the intermediate states that are required to exchanged.
- By implementing the FSVM scheme on the real-word dataset MNIST, the efficiency and effectiveness of both FSVM-S and FSVM-C are verified by comprehensive experimental results.

The rest of the paper is organized as following. The preliminaries for ADMM and secret sharing are offered in Sect. 2. In Sect. 3, the FSVM system model is present. Sections 4 and 5 elaborate FSVM-C and FSVM-S for the cases of data partitioning by examples and features, respectively. Comprehensive experimental results are demonstrated in Sect. 6 to verify the efficiency and effectiveness of the proposed scheme. At the end, Sect. 7 concludes this paper.

2 Preliminaries

In this section, we offer a simple introduction to two building blocks of the proposed FSVM scheme. One is the ADMM algorithm by which global SVM classifiers can be derived without requiring to collect all the training data together. The other is the secret sharing that enables to preserve the privacy of intermediate parameters (or states) when multiple data holders collaboratively run the ADMM algorithm.

2.1 The ADMM Algorithm

Since most machine learning models are derived by solving large-scale convex optimizations problems, there is an urgent need to put forward efficient algorithms to solve these problems. Fortunately, the ADMM algorithm is one of the popular solutions. By splitting the original problems into several sub-problems, this algorithm makes it possible to solve large-scale convex optimizations in a distributed manner. When moving to current smart city scenarios where raw training data are held by multiple entities, ADMM-based solutions offer an efficient way to train a more accurate model via their collaboration. Moreover, raw training data are locally kept when running this algorithm. Thus in the proposed FSVM scheme, the ADMM algorithm is adopted for multiple data holders to train a global SVM classifier.

As in [9], the optimization problem to solved is in the following format

$$\min \ f(\boldsymbol{x}) + g(\boldsymbol{z}) \quad s.t. \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c} \tag{1}$$

Authors of [9] offer the solution for this problem as

$$\boldsymbol{x}^{k+1} := \text{argmin}_{\boldsymbol{x}} L_\rho(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{y}^k) \tag{2}$$

$$\boldsymbol{z}^{k+1} := \text{argmin}_{\boldsymbol{z}} L_\rho(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{y}^k) \tag{3}$$

$$\boldsymbol{y}^{k+1} := \boldsymbol{y}^k + \rho(\boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}) \tag{4}$$

where L_ρ is the augmented Lagrangian. By iteratively computing (basic-solution-1) (basic-solution-3), $\boldsymbol{x}^k$ and $\boldsymbol{z}^k$ converge to the optimal solution to the problem defined by (1). Here, $\boldsymbol{x}$ and $\boldsymbol{z}$ are updated alternatively and this accounts for the term *alternating direction*. By introducing the dual variable $\boldsymbol{y}$, the minimization of $\boldsymbol{x}$ and $\boldsymbol{z}$ are separated and this equips the ADMM algorithm with the property of being run in a distributed manner.

2.2 Secret Sharing

Secret sharing schemes serve as the foundation for most of state-of-the-art privacy preserving techniques [13]. These schemes are suitable for multiple parties with comparable capabilities to collaboratively evaluate a variety of functions without revealing their own inputs to others. The main advantage of these schemes is that their implementations do not rely on any trusted third-party. According to [14], there are two categories of secret sharing schemes, namely arithmetic sharing and boolean sharing.

With respect to arithmetic sharing, the first scheme is referred as Shamir Secret Sharing, where the shares are the points of a polynomial [15]. Specifically, given a value s that a party want to share with other parties, the shares of s, denoted by $\{\langle s\rangle_1^A, \ldots, \langle s\rangle_n^A\}$ are computed as following

- The holder of s randomly picks a polynomial function $f(x) \in_r \mathbb{F}_p$ that satisfies $f(0) = s$.
- This holder randomly chooses s_i $(i = 1, \ldots, n)$ from $\mathbb{F}_p$ and computes $f(s_i)$. By this way, the i-th share of s is a tuple $\langle s\rangle_i^A = (s_i, f(s_i))$.

Note that, the degree of $f(x)$ determines the least number of shareholders required to reconstruct s. Let t denote the degree of $f(x)$ and at least $t + 1$ shareholders are needed for the reconstruction. Therefore, given $k > t$, the reconstruction can be done via

$$s = \sum_{i=1}^{k} f(s_i)\Big(\prod_{j=1, j\neq i}^{k} \frac{s_j}{s_i - s_i}\Big) mod(p) \tag{5}$$

When there are only two parties, such sharing becomes the form that is proposed in [14]. Arithmetic sharing has been proved to be efficient to collaboratively

privately evaluate functions containing arithmetic operations, like additions and multiplications.

As for boolean sharing, it is usually utilized in the two-party case. Here, the secret to be shared is represented by its binary format and each of the bits is shared via XOR-operations. Given the secret s to be shared, its binary format is $[s_1, \ldots, s_l]$. For the i-th bit s_i, its boolean shares, $\langle s_i \rangle_0^B$ and $\langle s_i \rangle_0^B$, are generated by

- The holder of s randomly chooses one bit r from $\{0, 1\}$ and set $\langle s_i \rangle_0^B = r$ that is kept by himself.
- This holder sets $\langle s_i \rangle_1^B = s_i \oplus r$ and sends $\langle s_i \rangle_1^B$ to the other party.

To reconstruct s, it is just needed to compute $\langle s_i \rangle_0^B \oplus \langle s_i \rangle_1^B$. The types of operations tackled by boolean sharing are XOR and AND. Thus, boolean sharing is good at privately evaluating functions that mainly contain comparison operation. For further details, we refer the readers to [14].

3 System Model

For traditional SVMs, all the data entries are collected and trained to derive a classification model by a centralized server. Assume there are M data entries. For the i-th entry, we have $\boldsymbol{x}_i \in \boldsymbol{R}^D$ and the label $y_i \in \{-1, 1\}$. Let $\boldsymbol{\xi}$ denote the vector of slack variables. The goal is to obtain the classification model as a maximum-margin linear discriminant function $g^* = \boldsymbol{x}^T \boldsymbol{\omega}^* + b^*$ by solving the following convex optimization problem

$$\begin{aligned} &\min_{\boldsymbol{\omega}, b, \boldsymbol{\xi}} \ \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{M} \xi_i \\ &s.t. \quad y_i(\boldsymbol{\omega}\boldsymbol{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (i = 1, \cdots, M) \end{aligned} \tag{6}$$

Since in many smart city applications, training data are from different data holders and they collaborate with each other to obtain a more accurate classification model. Assume that there are N participants to act as data holders in the system model for the proposed FSVM scheme. For the i-th participant, the number of data entries that he holds is assumed to be M_i and $\boldsymbol{X}_i$ is the matrix that stores these data entries. Let $\boldsymbol{x}_{ij}$ and y_{ij} $(j = 1, \ldots, M_i)$ be the j-th data entry and the corresponding label, respectively. Here, $y_{ij} \in \{-1, 1\}$. In addition, the total number of features that are considered in our scheme is D. As stated in [12], when referring to the scenario where multiple data holders collaboratively train a model, there are two types of data partitioning conditions, namely partitioning by examples and partitioning by features. Thus in the proposed FSVM scheme, two cases related to the different data partitioning conditions should be considered.

The first one is related to data partitioning by examples. This means that the data entries of each participant have the same number of features. Thus,

$\boldsymbol{X}_i \in \mathbb{R}^{M_i \times D}$ and $\boldsymbol{x}_{ij}^T \in \mathbb{R}^D$ $(i = 1, \ldots, N)$. The superscript T means transposition. Here, to collaboratively obtain the maximum-margin linear discriminant function, the optimization problem to be solved changes from the one defined by (6) to

$$\begin{aligned} &\min_{\boldsymbol{\omega}_i, b_i, \xi_{i,j}} \ \frac{1}{2}\sum_{i=1}^{N} \|\boldsymbol{\omega}_i\|^2 + NC\sum_{i=1}^{N}\sum_{j=1}^{M_i}\xi_{ij} \\ &s.t. \begin{cases} y_{ij}(\boldsymbol{x}_{ij}\boldsymbol{\omega}_i + b_i) \geq 1 - \xi_{ij}, \quad \xi_{ij} \geq 0 \\ (i = 1, \cdots, N,\ j = 1, \cdots, M_i) \\ \boldsymbol{\omega}_i = \boldsymbol{\omega}_k, b_i = b_k (i, k = 1, \ldots, N \text{ and } i \neq k) \end{cases} \end{aligned} \tag{7}$$

Here, $\boldsymbol{\omega}_i \in \mathbb{R}^D$ $(i = 1, \ldots, N)$. After solving (7), each participant obtains the same $\boldsymbol{\omega}^*$ and b^*. This is also named consensus-based SVM [8]. For simplicity, the proposed FSVM scheme for this case is denoted by FSVM-C.

The second case is related to data partitioning by features. Here, each participant has the same number of data entries but the number of features of each entry is different. An example for this case is when different companies that have an overlapping set of customers and they collaborate to derive a global classifier that takes richer features into consideration. Let D_i denote the number of data features that the i-th participant holds. We have $M_1 = \cdots = M_N = M$, $\sum_{i=1}^{N} D_i = D$, $\boldsymbol{X}_i \in \mathbb{R}^{M \times D_i}$ and $\boldsymbol{x}_{ij} \in \mathbb{R}^{D_i}$.

$$\begin{aligned} &\min_{\boldsymbol{\omega}_i, b_i, \xi_{i,j}} \ \frac{1}{2}\sum_{i=1}^{N} \|\boldsymbol{\omega}_i\|^2 + \sum_{j=1}^{M}\xi_j \\ &s.t. \quad \begin{cases} y_j \sum_{i=1}^{N}(\boldsymbol{x}_{ij}\boldsymbol{\omega}_i + b_i) \geq 1 - \xi_j, \quad \xi_j \geq 0 \\ (j = 1, \cdots, M) \end{cases} \end{aligned} \tag{8}$$

Here, $\boldsymbol{\omega}_i \in \mathbb{R}^{D_i}$ $(i = 1, \ldots, N)$. Following [9], this is the sharing-based SVM and the proposed FSVM scheme for this case is called FSVM-S.

Considering the definition of threat model, we assume that all the enrolled participants in the proposed FSVM scheme are semi-honest. This implies that these participants act honestly following the proposed scheme but are curious about what they have received from other participants. In addition, it is assumed that there are no colluding participants. The same as [10] and [11], the privacy preserving goal is to protect both the raw data and intermediate parameters (i.e. the output of each iteration when adopting the ADMM algorithm) against semi-honest collaborators when solving the problems defined by (7) and (8).

4 The Proposed FSVM-C Scheme

As in this paper, the proposed FSVM scheme contains the FSVM-C and FSVM-S for data partitioning by examples and features, respectively. In this section, we first explicitly present how FSVM-C works.

4.1 Problem Reformulation

Following [8], the problem defined by (7) that is solved by FSVM-C in a privacy-preserving manner can be reformulated into a more compact form. Set $\boldsymbol{v}_i = [\boldsymbol{\omega}_i^T, b_i]^T \in \mathbb{R}^{D+1}$, $\boldsymbol{B}_i = [\boldsymbol{X}_i, \mathbf{1}_i]$ ($\mathbf{1}_i \in \mathbb{R}^{M_i}$), $\boldsymbol{Y}_i = diag([y_{i1}, \cdots, y_{iM_i}])$, and $\boldsymbol{\xi}_i = [\xi_{i1}, \cdots, \xi_{iM_i}]^T \in \mathbb{R}^{M_i}$. Moreover, let $\boldsymbol{I}_{D+1}$ be the identity matrix of dimension $D+1$ and $\boldsymbol{\Pi}_{D+1} \in \mathbb{R}^{(D+1)\times(D+1)}$ be a matrix of zeros except $\boldsymbol{\Pi}_{D+1,D+1} = 1$. The more compact form of the optimization problem defined by (7) becomes

$$\begin{aligned} &\min_{\boldsymbol{v}_i, \boldsymbol{\xi}_i} \ \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{v}_i^T(\boldsymbol{I}_{D+1} - \boldsymbol{\Pi}_{D+1})\boldsymbol{v}_i + NC\sum_{i=1}^{N}\mathbf{1}_i^T\boldsymbol{\xi}_i \\ &s.t. \quad \begin{cases} \boldsymbol{Y}_i\boldsymbol{B}_i\boldsymbol{V}_i \succeq \mathbf{1}_i - \boldsymbol{\xi}_i, \quad \boldsymbol{\xi_i} \succeq \mathbf{0}_i(\forall i) \\ \boldsymbol{A}\boldsymbol{V} = 0 \end{cases} \end{aligned} \tag{9}$$

Here, $\boldsymbol{V} = [\boldsymbol{v}_1^T, \ldots, \boldsymbol{v}_N^T]^T \in \mathbb{R}^{N(D+1)}$. As for $\boldsymbol{A} \in \mathbb{R}^{N(D+1)\times(N-1)(D+1)}$, it is defined as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_{D+1} & -\boldsymbol{I}_{D+1} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \ddots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{I}_{D+1} & -\boldsymbol{I}_{D+1} \end{bmatrix}$$

Note that $\boldsymbol{A}$ is introduced to settle down the constraints $\boldsymbol{\omega}_i = \boldsymbol{\omega}_k$ and $b_i = b_k$ $(i, k = 1, \ldots, N$ and $i \neq k)$ in (7).

To solve the optimization problem defined by (9), the authors of [11] offer an ADMM-based solution with time-varying penalty matrices as

$$\{\boldsymbol{v}_i^{k+1}, \boldsymbol{\xi}_i^{k+1}\} = \arg\min_{\boldsymbol{v}_i, \boldsymbol{\xi}_i} L(\boldsymbol{v}_i, \boldsymbol{\xi}_i, \boldsymbol{\rho}^k, \boldsymbol{\lambda}^k) + \frac{r_i}{2}\|\boldsymbol{v}_i - \boldsymbol{v}_i^k\|^2 \tag{10}$$

$$\boldsymbol{\rho}^k \to \boldsymbol{\rho}^{k+1} \tag{11}$$

$$\lambda_{i,i+1}^{k+1} = \lambda_{i,i+1}^{k} + \rho_{i,i+1}^{k+1}(\boldsymbol{v}_i^{k+1} - \boldsymbol{v}_{i+1}^{k+1}) \tag{12}$$

where $L(\boldsymbol{v}, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\rho})$ is the augmented surrogate. Lagrangian function that is defined as

$$\begin{aligned} L(\boldsymbol{v}, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\rho}) &= \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{v}_i^T(\boldsymbol{I}_{D+1} - \boldsymbol{\Pi}_{D+1})\boldsymbol{v}_i + NC\sum_{i=1}^{N}\mathbf{1}_i^T\boldsymbol{\xi}_i \\ &+ \sum_{i=1}^{N-1}(\boldsymbol{\lambda}_{i,i+1}^T(\boldsymbol{v}_i - \boldsymbol{v}_{i+1}) + \frac{\boldsymbol{\rho}_{i,i+1}}{2}\|\boldsymbol{v}_i - \boldsymbol{v}_{i+1}\|^2) \end{aligned} \tag{13}$$

As for the set of penalties $\boldsymbol{\rho}$, the following two requirements guarantee the convergence to optimal solutions

$$\mathbf{0} \preccurlyeq \boldsymbol{\rho}^k \preccurlyeq \boldsymbol{\rho}^{k+1} \preccurlyeq \bar{\boldsymbol{\rho}} \tag{14}$$

$$\boldsymbol{Q}_P + \boldsymbol{I}_{ND} \succcurlyeq \boldsymbol{A}^T\bar{\boldsymbol{\rho}}\boldsymbol{A} \tag{15}$$

Here, $\boldsymbol{Q}_P = diag\{r_1, \ldots, r_N\} \otimes \boldsymbol{I}_D \in \mathbb{R}^{ND \times ND}$ ($\otimes$ refers to the Kronecker product).

From the iterated solution above, (10) can be locally solved by each enrolled participant and this is the key to form a distributed solution. Even though the locally hold data need not to be exchanged among the participants, the intermediate state $\boldsymbol{v}_i$ after each iteration should be sent to others when locally solving (10). This violates the privacy-preserving requirements as such intermediate states might reveal sensitive information related the locally hold data [10,11].

To protect $\boldsymbol{v}_i$ after each iteration, each penalty $\rho_{i,i+1}^k$ $(i = 1, \ldots, N-1)$ for iteration k is constructed as the multiplication of q_i and q_{i+1} which are secretly hold by the i-th and j-th participants, respectively. By introducing the homomorphic encryption scheme, the locally computed $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}^k$ are masked when computing $\rho_{i,i+1}^k(\boldsymbol{v}_i - \boldsymbol{v}_{i+1})$. Due to the limited number operations that can be supported by homomorphic encryption and the heavy computation overhead, we proposed a more efficient solution, namely FSVM-C, which is built upon the ADMM-based solution with time-varying penalty matrices.

4.2 The FSVM-C Scheme

To depict how the proposed FSVM-C works, we focus on the operations of i-th participant. For the $(k+1)$-th iteration, the first for this participant is to solve the optimization problem defined by (10). According to the formulation of the augmented surrogate Lagrangian function in (13), $\boldsymbol{v}_i^{k+1}$ satisfies

$$\nabla_{\boldsymbol{v}_i}(L(\boldsymbol{v}, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\rho}) + \frac{r_i}{2}\|\boldsymbol{v}_i - \boldsymbol{v}_i^k\|^2) = \boldsymbol{0} \tag{16}$$

Following the $\boldsymbol{\lambda}$-update in (11), $\boldsymbol{\lambda}_{i-1,i}^k$ and $\boldsymbol{\lambda}_{i,i+1}^k$ are computed as

$$\boldsymbol{\lambda}_{i-1,i}^k = \boldsymbol{\lambda}_{i-1,i}^{k-1} + \rho_{i-1,i}^k(\boldsymbol{v}_{i-1} - \boldsymbol{v}_{i+1}) \tag{17}$$

$$\boldsymbol{\lambda}_{i,i+1}^k = \boldsymbol{\lambda}_{i,i+1}^{k-1} + \rho_{i,i+1}^k(\boldsymbol{v}_i - \boldsymbol{v}_{i+1}) \tag{18}$$

From (16), (17), and (18), it is explicit that the i-th participant needs to collect $\boldsymbol{v}_{i-1}^k$ and $\boldsymbol{v}_{i+1}^k$ to solve the equation in (16) to obtain $\boldsymbol{v}_i^{k+1}$. Considering the privacy-preserving goal, $\boldsymbol{v}_{i-1}^k$ and $\boldsymbol{v}_{i+1}^k$ should not be exposed to the i-th participant. In the following, we show how to achieve this in a more efficient manner compared to [11]. Without loss of generality, we show the detailed process for computing $\rho_{i,i+1}^k(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k)$ without revealing $\boldsymbol{v}_{i+1}^k$ and $\boldsymbol{v}_i^k$ to each other. As for computing $\rho_{i-1,i}^k(\boldsymbol{v}_{i-1}^k - \boldsymbol{v}_i^k)$, the privacy-preserving goal can be achieved in the same manner.

In [11], $\rho_{i,i+1}^k$ is constructed as $\rho_{i,i+1}^k = q_{i,i+1}^k \cdot q_{i+1,i}^k$ where $q_{i,i+1}^k$ and $q_{i+1,i}^k$ are privately held by the i-th and j-th participants, respectively. With the help of the additional homomorphic encryption scheme, $\rho_{i-1,i}^k(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k)$ can be derived while keeping $\boldsymbol{v}_{i+1}^k$ and $\boldsymbol{v}_i^k$ secret. However, this kind of method introduces heavy computation overhead. In the proposed FSVM-C scheme, we split $\rho_{i,i+1}^k$ as $\rho_{i,i+1}^k = q_{i,i+1}^k + q_{i+1,i}^k$ and propose a new off-line protocol to securely

generate $q_{i,i+1}^k$ and $q_{i+1,i}^k$. Then, the algorithm to solve the optimization problem defined by (7) in an efficient and privacy-preserving manner is given.

To satisfying the conditions of (14) and (15) for $\boldsymbol{\rho}$-update, we have the following observations

- For r_i $(i = 1, \ldots, N)$, they are predefined and public parameters and their values are kept unchanged during the iterations for deriving the optimal solution.
- When focusing on the i-th participant, $\rho_{i,i+1}^k \leq \rho_{i,i+1}^{k+1}$ and the upper bound is determined by r_i.
- Once r_i is determined, a series of $\rho_{i,i+1}$ can be derived to satisfying the conditions of (14) and (15) and this process is independent from the iterations to derive the optimal solution.

From this observation, for the i-th and $i+1$-th participants, they should collaboratively generate two sequences of $\{q_{i,i+1}^k\}$ and $\{q_{i+1,i}^k\}$ that the sequence of $\{q_{i,i+1}^k + q_{i+1,i}^k\}$ is in an ascending order as k increases. Let Q denotes the size of such sequence and Q is large enough for the iterations of (10)–(12) converging to an optimal solution. The basic idea for generating these two sequences is that these two participants first generate two random sequences of size $Q+1$, namely $\boldsymbol{q}_{i,i+1}$ and $\boldsymbol{q}_{i+1,i}$, and then rearrange these two sequences such that $\{q_{i+1,i}^k + q_{i+1,i}^k\}$ is in an ascending order as k increases. The core operation in this process, referred as CMP, is to compare $q_{i,i+1}^k + q_{i+1,i}^k$ and $q_{i,i+1}^{k+1} + q_{i+1,i}^{k+1}$ without exposing $q_{i,i+1}^k$, $q_{i,i+1}^{k+1}$, $q_{i+1,i}^k$ and $q_{i+1,i}^{k+1}$ to the other party.

In the following, we elaborate how to achieve the function of CMP.

- **Step 1**: For the i-th participant, he transforms $q_{i,i+1}^k$ to its l-bit binary denotation as $[q_{i,i+1}^k]$. Then, two l-bit binary numbers, $\langle q_{i,i+1}^k \rangle_1^B$ and $\langle q_{i,i+1}^k \rangle_2^B$ are randomly selected such that

$$[q_{i,i+1}^k] = \langle q_{i,i+1}^k \rangle_1^B \oplus \langle q_{i,i+1}^k \rangle_2^B$$

  where $\oplus$ denotes the XOR operation. Here, $\langle q_{i,i+1}^k \rangle_1^B$ and $\langle q_{i,i+1}^k \rangle_2^B$ are the boolean shares of $[q_{i,i+1}^k]$. In the same manner, the boolean shares of $q_{i,i+1}^{k+1}$, $q_{i+1,i}^k$ and $q_{i+1,i}^{k+1}$ are generated.
- **Step 2**: Before utilizing boolean shares to privately compare $q_{i,i+1}^k + q_{i+1,i}^k$ and $q_{i,i+1}^{k+1} + q_{i+1,i}^{k+1}$, we need to construct the corresponding boolean circuits beforehand. Here, the improved boolean circuits for addition and comparison proposed by [16] are introduced. Figure 1 shows the boolean circuit for the CMP operation.
- **Step 3**: Following the process shown in Fig. 1, the i-th and $i+1$-th generated the shares of the final comparison result. After exchanging the shares, these two participants finally obtain the comparison result. For the detailed process, we refer the readers to [16].

With the help of CMP operation, the i-th and $i+1$-th participants can generate $\{q_{i,i+1}^k\}$ and $\{q_{i+1,i}^k\}$ (of size $Q+1$) such that $\{q_{i,i+1}^k + q_{i+1,i}^k\}$ is in an

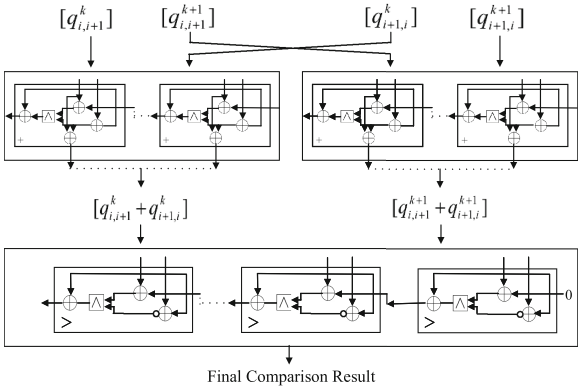

Fig. 1. The boolean circuit for CMP operation

ascending order. $q_{i,i+1}^{Q+1}$ and $q_{i+1,i}^{Q+1}$ are published and $q_{i,i+1}^{Q+1} + q_{i+1,i}^{Q+1}$ is utilized to generate $\boldsymbol{Q}_P$ satisfying the condition of (15).

For the k-th iteration, given $\{q_{i,i+1}^k\}$ and $\{q_{i+1,i}^k\}$, the i-th and $i+1$-th participants privately derive $\rho_{i,i+1}^k(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k)$ by the following procedures

- **Step 1**: For the i-th participant, $\langle q_{i,i+1}^k \rangle_1^A$ and $\langle q_{i,i+1}^k \rangle_2^A$ are randomly selected such that $\langle q_{i,i+1}^k \rangle_1^A + \langle q_{i,i+1}^k \rangle_2^A = q_{i,i+1}^k$. Here, $\langle q_{i,i+1}^k \rangle_1^A$ and $\langle q_{i,i+1}^k \rangle_2^A$ are the arithmetic shares of $q_{i,i+1}^k$. Then, the arithmetic shares of $-\boldsymbol{v}_i^k$ and $-q_{i,i+1}\boldsymbol{v}_i^k$, namely $\langle -\boldsymbol{v}_i^k \rangle_1^A$, $\langle -\boldsymbol{v}_i^k \rangle_2^A$, $\langle -q_{i,i+1}\boldsymbol{v}_i^k \rangle_1^A$ and $\langle -q_{i,i+1}\boldsymbol{v}_i^k \rangle_2^A$, are generated. $\langle q_{i,i+1}^k \rangle_2^A$, $\langle -\boldsymbol{v}_i^k \rangle_2^A$, and $\langle -q_{i,i+1}\boldsymbol{v}_i^k \rangle_2^A$ are sent to the i+1-th participant.
- **Step 2**: By the same way, the $i+1$-th participant generates $\langle q_{i+1,i}^k \rangle_1^A$, $\langle q_{i+1,i}^k \rangle_2^A$, $\langle \boldsymbol{v}_{i+1}^k \rangle_1^A$, $\langle \boldsymbol{v}_{i+1}^k \rangle_2^A$, $\langle q_{i+1,i}^k \boldsymbol{v}_{i+1}^k \rangle_1^A$ and $\langle q_{i+1,i}^k \boldsymbol{v}_{i+1}^k \rangle_2^A$. $\langle q_{i+1,i}^k \rangle_1^A$, $\langle \boldsymbol{v}_{i+1}^k \rangle_1^A$ and $\langle q_{i+1,i}^k \boldsymbol{v}_{i+1}^k \rangle_1^A$ are sent to the i-th participant.
- **Step 3**: Given $\langle q_{i+1,i}^k \rangle_1^A$, the i-th participant can locally compute its share of $-q_{i+1,i}^k \boldsymbol{v}_i^k$ and $q_{i,i+1}^k \boldsymbol{v}_{i+1}^k$, referred to $\langle -q_{i+1,i}^k \boldsymbol{v}_i^k \rangle_1^A$ and $\langle q_{i,i+1}^k \boldsymbol{v}_{i+1}^k \rangle_1^A$, with the help of pre-computed multiplication triple [17]. Then, $\langle (q_{i,i+1}^k + q_{i+1,i}^k)(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) \rangle_1^A$ is derived by

$$\begin{aligned}\langle (q_{i,i+1}^k + q_{i+1,i}^k)(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) \rangle_1^A &= \langle q_{i+1,i}^k \boldsymbol{v}_{i+1}^k \rangle_1^A q_{i,i+1}^k \boldsymbol{v}_{i+1}^k \rangle_1^A \\ &+ \langle -q_{i+1,i}^k \boldsymbol{v}_i^k \rangle_1^A + \langle -q_{i,i+1}\boldsymbol{v}_i^k \rangle_1^A\end{aligned}$$

  Similarly, the $i+1$-th participant is capable of locally computing $\langle (q_{i,i+1}^k + q_{i+1,i}^k)(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) \rangle_2^A$
- **Step 4**: Finally, these two participants exchange $\langle (q_{i,i+1}^k + q_{i+1,i}^k)(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) \rangle_1^A$ and $\langle (q_{i,i+1}^k + q_{i+1,i}^k)(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) \rangle_2^A$ to reconstruct $\rho_{i,i+1}(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) = (q_{i,i+1}^k + q_{i+1,i}^k)(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) = \langle (q_{i,i+1}^k + q_{i+1,i}^k)(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) \rangle_1^A + \langle (q_{i,i+1}^k + q_{i+1,i}^k)(\boldsymbol{v}_{i+1}^k - \boldsymbol{v}_i^k) \rangle_2^A$. Note that all the computations here are all mod 2^l.

After securely computing $\rho^k_{i,i+1}(\boldsymbol{v}^k_{i+1} - \boldsymbol{v}^k_i)$, utilizing the iterations of (10)–(12) to find the optimal solution will not reveal the intermediate state of each enrolled participant. Even though the proposed method introduces more interactions, they can be pre-computed and the heavy computation overhead introduced by homomorphic encryptions in [11] can be dramatically reduced.

5 The Proposed FSVM-S Scheme

To collaboratively train a SVM classifier with privacy concerns for the case of data partitioning by features, the FSVM-S scheme is proposed here. Like Sect. 4, we begin with reformulating the optimization problem defined by (8) to a more compact form. After that, the ADMM-based solution is presented. Even though locally held data are not required to be exchanged by such a solution, intermediate states still reveal sensitive information of raw data. Thus, the Shamir secret sharing scheme is introduced to protect such intermediate states.

5.1 Problem Reformulation

Similar to the FSVM-C scheme, we first reformulate the problem defined by (8) to make it easier to derive an ADMM-based solution. Let $(x)_+$ denote the function that $(x)_+ = \max\{0, x\}$. By introducing such a function, the slack variables can be eliminated and (8) becomes

$$\min \sum_{i=1}^{N} \frac{1}{2}\|\boldsymbol{\omega}_i\|^2 + \mathbf{1}_M^T(\mathbf{1}_M - \boldsymbol{Y}\sum_{i=1}^{M}(\boldsymbol{X}_i\boldsymbol{\omega}_i + b_i\mathbf{1}_M))_+ \tag{19}$$

where $\boldsymbol{Y} = diag[y_1, \ldots, y_M]$. In the sequel, set $\boldsymbol{v}_i = [[\boldsymbol{\omega}_i^T], b_i]^T \in \boldsymbol{R}^{D_i+1}$ and $\boldsymbol{B}_i = [\boldsymbol{X}_i^T, \mathbf{1}_M] \in \boldsymbol{R}^{M\times(D_i+1)}$. As a result, the problem of (19) is further transformed to

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} \frac{1}{2}\boldsymbol{v}_i^T(\boldsymbol{I}_{D_i+1} - \boldsymbol{\Pi}_{D_i+1})\boldsymbol{v}_i \\ & + \mathbf{1}_M^T(\mathbf{1}_M - Y\sum_{i=1}^{N}\boldsymbol{B}_i\boldsymbol{v}_i)_+ \end{aligned} \tag{20}$$

Recalling the standard problem defined by (1) that is solved by the ADMM algorithm, we introduce additional variables $\boldsymbol{z}_i \in \mathbb{R}^M$ $(i = 1, \ldots, N)$ to enable a distributed algorithm for solving the problem of (20). To achieve this, we put the constraint on $\boldsymbol{z}_i$ that $\boldsymbol{z}_i = \boldsymbol{B}_i\boldsymbol{v}_i$. Consequently, the problem defined by (20) is transformed to

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} \frac{1}{2}\boldsymbol{v}_i^T(\boldsymbol{I}_{D_i+1} - \boldsymbol{\Pi}_{D_i+1})\boldsymbol{v}_i + \mathbf{1}_M^T(\mathbf{1}_M - Y\sum_{i=1}^{N}\boldsymbol{z}_i)_+ \\ s.t. \quad & \boldsymbol{z}_i = \boldsymbol{B}_i\boldsymbol{v}_i,\ i = 1, \ldots, N \end{aligned} \tag{21}$$

By doing this, the augmented Lagrangian for (21) becomes

$$\begin{aligned} L_\rho &= \sum_{i=1}^{N} \frac{1}{2} \boldsymbol{v}_i^T (\boldsymbol{I}_{D_i+1} - \boldsymbol{\Pi}_{D_i+1}) \boldsymbol{v}_i + \mathbf{1}_M^T (\mathbf{1}_M - Y \sum_{i=1}^{N} \boldsymbol{z}_i)_+ \\ &+ \sum_{i=1}^{N} \boldsymbol{s}_i^T (\boldsymbol{B}_i \boldsymbol{v}_i - \boldsymbol{z}_i) + \frac{\rho}{2} \sum_{i=1}^{N} \|\boldsymbol{z}_i - \boldsymbol{B}_i \boldsymbol{v}_i\|^2 \\ &= \sum_{i=1}^{N} f(\boldsymbol{v}_i) + g(\sum_{i=1}^{N} \boldsymbol{z}_i) + \sum_{i=1}^{N} \boldsymbol{s}_i^T (\boldsymbol{B}_i \boldsymbol{v}_i - \boldsymbol{z}_i) \\ &+ \frac{\rho}{2} \sum_{i=1}^{N} \|\boldsymbol{z}_i - \boldsymbol{B}_i \boldsymbol{v}_i\|^2 \end{aligned} \tag{22}$$

In this way, the ADMM-based solution to the problem (21) is directed as

$$\boldsymbol{v}^{k+1} = \text{argmin}_{\boldsymbol{v}} L_\rho(\boldsymbol{v}, \boldsymbol{z}^k, \boldsymbol{s}^k) \tag{23}$$

$$\boldsymbol{z}^{k+1} = \text{argmin}_{\boldsymbol{z}} L_\rho(\boldsymbol{v}^{k+1}, \boldsymbol{z}, \boldsymbol{s}^k) \tag{24}$$

$$\boldsymbol{s}^{k+1} = \boldsymbol{s}^k + \rho(\sum_{i=1}^{N} \boldsymbol{B}_i \boldsymbol{v}_i^{k+1} - \boldsymbol{z}_i^{k+1}) \tag{25}$$

where $\boldsymbol{z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$. Considering the $\boldsymbol{z}$-update, there are $N \times M$ variables that increase the hardness for updating $\boldsymbol{z}$. In the proposed FSVM-S scheme, we first simplify the process of updating $\boldsymbol{z}$ by substituting $\boldsymbol{z}$ with $\overline{\boldsymbol{z}} = \sum_{i=1}^{N} \boldsymbol{z}_i / N$. Then, the Shamir secret sharing scheme is introduced to protect the intermediate state of each participant.

5.2 The Proposed FSVM-S Scheme

According to (24), updating $\boldsymbol{z}$ requires to solve a minimization problem that contains $N \times M$ variables. Once the number of enrolled participants in the FSVM-S scheme is large, the solution becomes dramatically complex. Fortunately, inspired by [9], $\boldsymbol{z}$ can be substituted by $\overline{\boldsymbol{z}} = \sum_{i=1}^{N} \boldsymbol{z}_i / N$. In the following, we will show how to achieve this.

Following [9], we first transform the iterations of (23)–(25) to their scaled form as

$$\boldsymbol{v}_i^{k+1} = \text{argmin}_{\boldsymbol{v}_i} f_i(\boldsymbol{v}_i) + \frac{\rho}{2} \|\boldsymbol{B}_i \boldsymbol{v}_i - \boldsymbol{z}_i^k + \boldsymbol{u}_i^k\|^2 \tag{26}$$

$$\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} g(\sum_{i=1}^{N} \boldsymbol{z}_i) \frac{\rho}{2} \sum_{i=1}^{N} \|\boldsymbol{z}_i - \boldsymbol{B}_i \boldsymbol{v}_i^{k+1} - \boldsymbol{u}_i^k\|^2 \tag{27}$$

$$\boldsymbol{u}_i^{k+1} = \boldsymbol{u}_i^k + \boldsymbol{B}_i \boldsymbol{v}_i^{k+1} - \boldsymbol{z}_i^{k+1} \tag{28}$$

Focusing on (27), let $\boldsymbol{a}_i^k = \boldsymbol{B}_i \boldsymbol{v}_i^{k+1} + \boldsymbol{u}_i^k$. To update $\boldsymbol{z}$ is equivalent to solve the following optimization problem.

$$\min g(N\overline{\boldsymbol{z}}) + \frac{\rho}{2} \sum_{i=1}^{N} \|\boldsymbol{z}_i - \boldsymbol{a}_i^k\| \quad s.t. \quad N\overline{\boldsymbol{z}} = \sum_{i=1}^{N} \boldsymbol{z}_i \tag{29}$$

After solving (29), for $i = 1, \ldots, N$, one can derive

$$\boldsymbol{z}_i^{k+1} = \boldsymbol{a}_{\boldsymbol{i}}^{\boldsymbol{k}} + \overline{\boldsymbol{z}}^{k+1} - \overline{\boldsymbol{a}}^k \tag{30}$$

where

$$\overline{\boldsymbol{a}}^k = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{a}_i^k = \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{B}_i \boldsymbol{v}_i^{k+1} + \boldsymbol{u}_i^k) = \overline{\boldsymbol{u}}^k + \overline{\boldsymbol{B}\boldsymbol{v}}^{k+1}$$

Substituting $\boldsymbol{z}_i^{k+1}$ computed by (30) into (28) for updating $\boldsymbol{u}_i$, we derive

$$\boldsymbol{u}_i^{k+1} = \overline{\boldsymbol{a}}^k - \overline{\boldsymbol{z}}^{k+1} = \overline{\boldsymbol{u}}^k + \overline{\boldsymbol{B}\boldsymbol{v}}^{k+1} - \overline{\boldsymbol{z}}^{k+1} \tag{31}$$

From (31), all the values of $\boldsymbol{u}_i^{k+1}$ $(i = 1, \ldots, N)$ are the same. Hence, the iterations to derive the solution become

$$\boldsymbol{v}_i^{k+1} = \arg\min_{\boldsymbol{v}_i} f_i(\boldsymbol{v}_i) + \frac{\rho}{2} \|\boldsymbol{B}_i \boldsymbol{v}_i - \boldsymbol{B}_i \boldsymbol{v}_i^k - \overline{\boldsymbol{z}}^k + \overline{\boldsymbol{B}\boldsymbol{v}}^k + \boldsymbol{u}^k\|^2 \tag{32}$$

$$\bar{\boldsymbol{z}}^{k+1} = \arg\min_{\bar{\boldsymbol{z}}} g(N\bar{\boldsymbol{z}}) + \frac{N\rho}{2} \|\bar{\boldsymbol{z}} - \overline{\boldsymbol{B}\boldsymbol{v}}^{k+1} - \boldsymbol{u}^k\|^2 \tag{33}$$

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \overline{\boldsymbol{B}\boldsymbol{v}}^{k+1} - \overline{\boldsymbol{z}}^{k+1} \tag{34}$$

After simplifying (27) to (33), one can derive the expression of $\overline{\boldsymbol{z}}^{k+1}$ in the component level, i.e. $\overline{\boldsymbol{z}}_j^{k+1}$ $(j = 1, \ldots, M)$

$$\overline{\boldsymbol{z}}_j^{k+1} = \begin{cases} \overline{\boldsymbol{a}}_j^k + y_j/\rho, & y_j \overline{\boldsymbol{a}}_j^k \leq 1/N - 1/\rho \\ \frac{1}{N}, & 1/N - 1/\rho < y_j \overline{\boldsymbol{a}}_j^k < 1/N \\ \overline{\boldsymbol{a}}_j^k, & y_j \overline{\boldsymbol{a}}_j^k \geq 1/N \end{cases} \tag{35}$$

where $\overline{\boldsymbol{a}} = \overline{\boldsymbol{B}\boldsymbol{v}}^{k+1} + \boldsymbol{u}^k \in \mathbb{R}^M$.

After deriving the iterations to obtain the optimal solution to the problem defined by (21), the next is to analyze what intermediate states are exposed to others. From the iterations of (32)–(34), if all the participants can collaboratively compute $\overline{\boldsymbol{B}\boldsymbol{v}} = \sum i = 1^N (\boldsymbol{B}_i \boldsymbol{v}_i)/N$, $\boldsymbol{v}_i$ $(i = 1, \ldots, N)$, $\overline{\boldsymbol{z}}$ and $\boldsymbol{u}$ can be locally updated by each participant in a distributed manner. Hence, in the proposed FSVM-S scheme, how to compute $\overline{\boldsymbol{B}\boldsymbol{v}}$ collaboratively and privately should be thorough addressed. To achieve this, the detailed procedures are as following.

– **Step 1**: Fixing the iteration index k, the i-th participant picks a polynomial function $f_i^k(x) \in_r \mathbb{F}_p$ of degree t $(t + 1 \leq N)$.

– **Step 2**: Focusing on the i-th participant $(i = 1, \ldots, N)$, $\boldsymbol{B}_i\boldsymbol{v}_i^k \in \mathbb{R}^M$ is computed. For the j-th element $(\boldsymbol{B}_i\boldsymbol{v}_i^k)_j$ $(j = 1, \ldots, M)$, set $f_i^k(0) = (\boldsymbol{B}_i\boldsymbol{v}_i)_j$. Then, generate N random values as $\{h_{ij}^1, \ldots, h_{ij}^N\}$. The shares of $(\boldsymbol{B}_i\boldsymbol{v}_i^k)_j$ can be obtained as

$$\langle(\boldsymbol{B}_i\boldsymbol{v}_i^k)_j\rangle_n = (h_{ij}^n, f_i^k(h_{ij}^n)) \quad (n = 1, \ldots, N) \tag{36}$$

This participant keeps on share by himself and sends the left $N-1$ shares to the other participants.

– **Step 3**: After receiving the shares from the other participants, the i-th participant computes

$$\langle(N\overline{\boldsymbol{B}\boldsymbol{v}})_j^k\rangle_i = \sum_{n=1}^{N}\langle(\boldsymbol{B}_n\boldsymbol{v}_n^k)_j\rangle_i \quad (j = 1, \ldots, M) \tag{37}$$

– **Step 4**: Following (5), $N(\overline{\boldsymbol{B}\boldsymbol{v}}^k)_j$ can be reconstructed by any $t+1$ participants and by further dividing N, $(\overline{\boldsymbol{B}\boldsymbol{v}}^k)_j$ is derived.

In such a manner, $\overline{\boldsymbol{B}\boldsymbol{v}}^k$ can be privately computed when $N \geq 3$. When $N = 2$, given the reconstructed result and his own intermediate state, one participant can derive the intermediate state of the other. This is inevitable here but we can adopt the method in [10] to alleviate this problem. The basic idea is to add controllable noise to the intermediate states to satisfy the requirements of differential privacy. Consequently, given $\overline{\boldsymbol{B}\boldsymbol{v}}^k$, the iterations of (32)–(34) can be locally computed without revealing $\boldsymbol{B}_i\boldsymbol{v}_i$ to others.

6 Performance Evaluation

The performance of the proposed FSVM protocol is twofold, efficiency and effectiveness. Before demonstrating experimental results, we first offer the settings of the experiments. The following experiments are carried on a desktop running the 64-bit Ubuntu-18.04.1 operating system with Intel Core i7 CPU and 64 GB memory. Different participants are simulated by different threads in this desktop. To achieve the CMP function, the open-source ABY framework proposed by [14] is adopted. Specifically, we set the bit length $l = 32$ and thus all the arithmetic shares are in the range of $(0, 2^l)$. The secret sharing scheme is implemented by C++ and multiplication triples are generated following [18]. By these settings, the time efficiency is evaluated. As for the effectiveness, the FSVM is implemented on the MNIST, which is a widely utilized dataset of handwritten digits in the area of artificial intelligence[1].

[1] http://yann.lecun.com/exdb/mnist/.

6.1 Efficiency Evaluation

Since secret sharing is introduced to achieve privacy preserving goals, there is a great concern on the time efficiency when combining secret sharing with ADMM iterations. In the experiments, the communications among the participants are realized via sockets and information exchanging is in a pair-wise manner (referred as the client and server in sockets). Thus, the running time for any two participants privately exchanging intermediates in one iteration is evaluated and the final result is derived as the average of the outputs for running each experiment 100 times.

Here, the number of operations and the amount of data to be exchanged are determined by the number of features D. Table 1 presents experimental results when D varies from 100 to 1000 with a step size of 100. As there is a precomputing phase to generate $\{q^k_{i,i+1}\}$ and $\{q^k_{i+1,i}\}$ for protecting intermediate states, the implementation of FSVM-C is divided by an offline phase for this precomputation and an online phase for secretly exchanging intermediate states. In addition, the running time for the method in [11] is evaluated for comparison with FSVM-C when data partitioning by examples. Table 1 presents the experimental results related to time efficiency.

Table 1. Efficiency evaluation for FSVM (s)

D	FSVM-C			Method in [11]	FSVM-S
	Q	Offline	Online		
100	100	0.214	$2.76E-4$	0.0698	$2.19E-4$
200	200	0.810	$3.10E-4$	0.155	$2.75E-4$
300	300	1.83	$3.33E-4$	0.217	$2.92E-4$
400	400	3.24	$3.70E-4$	0.297	$3.07E-4$
500	500	5.13	$4.01E-4$	0.362	$3.20E-4$
600	600	7.46	$4.16E-4$	0.432	$3.32E-4$
700	700	10.02	$4.53E-4$	0.509	$3.44E-4$
800	800	13.21	$4.82E-4$	0.615	$3.67E-4$
900	900	16.60	$5.08E-4$	0.647	$3.79E-4$
1000	1000	20.56	$5.35E-4$	0.722	$3.98E-4$

Apparently, when D increases, more data are needed to be computed and transmitted and there is no doubt that the running time increases for the method in [11], FSVM-C and FSVM-S. Focusing on the running time of the offline phase in FSVM-C, the goal is to generate enough random numbers that satisfy the conditions of (14) and (15) and each is utilized in one iteration. Thus the running time for this phase is independent of D determined by the number of random numbers to be generated (referred as Q). When Q linearly increases from 100 to

1000 with a step size of 100, the running time approximately increases quadratically. This is because when the operation CMP is introduced in this phase, it needs Q additions and $Q(Q-1)/2$ comparisons via secret sharing. Note that additions are very fast and the running time is dominated by the number of comparisons. With respect to the running time for the online phase in FSVM-C, the method in [11] and FSVM-S, it linearly increases when D increases. Specifically, in the case of data partitioning by examples, benefiting from the offline phase, the running time for FSVM-C is about 1350 times lower than that for the method [11]. This significant improvement is due to the heavy computation overhead introduced by additional homomorphic encryption in [11]. As for FSVM-S, data are partitioned by features and the number of features to be considered by the enrolled participants is smaller than that of FSVM-C. It is direct that the running time for FSVM-S is smaller than of that for FSVM-C.

6.2 Effectiveness Evaluation

To evaluated the effectiveness of the proposed FSVM scheme, we implement it on the popular MNIST dataset. For binary classifications, the images of digit 2 and digit 9 are chosen with positive and negative labels. As each image is of size 28 by 28 pixels, each data entry consists of 784 features. After filtering, there are 5470 and 5454 images of digit 2 and 9, respectively. Thus, for the case of data partitioning by examples, there are 10924 data entries that are randomly distributed to N participants. For the case of data partitioning by features, M is set to 5000 and N participants evenly share these features. Here, we set $N = 2, 3, 4,$ and 5. For the test set, the numbers of images of 2 and 9 are both 500. Due to space limitation, the prediction accuracy, denoted by P_a, is utilized here as the performance matrix for effectiveness evaluation.

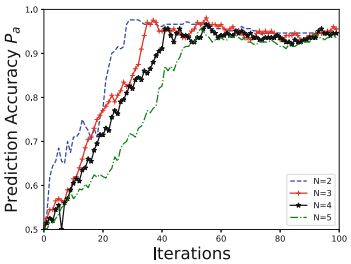

Fig. 2. Prediction Accuracy P_a versus iterations, where the FSVM-C is implemented and N = 2, 3, 4, and 5.

As is shown in Fig. 2, the prediction accuracy of FSVM-C is offered as the number of iterations increases. At the beginning, the initial states are randomly selected and the prediction here is like a random guess. Thus, all the curves starts from around 0.5. It is explicit that all the P_as converges to around 0.95

when the values of N vary but the speeds for convergence are different. In the case where $N = 2$, P_a converges with the highest speed and the convergence speed for $N = 5$ is the lowest. This is because as N increases larger, the number of data entries hold by each participant becomes smaller and less data entries means that the participant mainly relies on the received intermediate states from others to derive a more generalized classifier. For the worst case where $N = 5$, P_a converges after around 50 iterations and this implies that the number of random values to be generated for privacy preserving by each participant, namely Q, can be set slightly larger than 50.

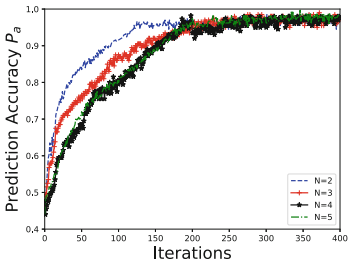

Fig. 3. Prediction Accuracy P_a versus iterations, where the FSVM-S is implemented and N = 2, 3, 4, and 5.

Figure 3 depicts the prediction accuracy P_a of FSVM-S versus the number of iterations where N varies from 2 to 5. The experimental results here are similar to those of FSVM-C. All the curves starts from near 0.5 due to the random selection of initial states and larger N means slower convergence speed. Unlike FSVM-C where each participant can obtain partial information of the global data distribution from the locally held data, any independent participant in FSVM-S can only have some of the features to be considered for classification and the relationships among these features might not be obvious. As a result, less information about the global data distribution would be derived when FSVM-S is implemented where data are partitioned by features. Hence, P_a of FSVM-S has a slower convergence speed here than that of FSVM-C and even for the best case when $N = 2$, P_a converges after more than 100 iterations.

7 Conclusion

In this paper, we propose the FSVM scheme to construct SVM classifiers in a distributed and privacy-preserving manner. Since there are two cases where data are partitioned by examples and features, the FSVM consists of FSVM-C and FSVM-S to deal with these two cases, respectively. In FSVM-C, the new ADMM in [11] that allows time-varying matrices is incorporated with secret sharing to achieve the privacy-preserving goal in a more efficient manner. Also, there is

no need for a trusted third-party to generate keys. With respect to FSVM-S, we derive the closed form of the ADMM iterations and the Shamir secret sharing is introduced to protect intermediate states. By implementing the FSVM scheme on the dataset MNIST, the efficiency and effectiveness of both FSVM-S and FSVM-C are verified by comprehensive experimental results. Moreover, these experimental results also present an explicit guideline for implementing the proposed FSVM scheme in practical smart city applications.

Acknowledgement. This work is supported by the National Key Research and Development Program of China under Grant 2021YFB2700600, the National Natural Science Foundation of China under Grant 62132013 and 61902292, the Key Research and Development Programs of Shaanxi under Grants 2021ZDLGY06-03.

References

1. Dong, Y., Guo, S., Liu, J., Yang, Y.: Energy-efficient fair cooperation fog computing in mobile edge networks for smart city. IEEE Internet Things J. **6**(5), 7543–7554 (2019)
2. Sadiq, A.S., Faris, H., Ala'M, A.-Z., Mirjalili, S., Ghafoor, K.Z.: Fraud detection model based on multi-verse features extraction approach for smart city applications. In: Smart Cities Cybersecurity and Privacy, pp. 241–251. Elsevier (2019)
3. Rasheed, W., Tang, T.B.: Anomaly detection of moderate traumatic brain injury using auto-regularized multi-instance one-class SVM. IEEE Trans. Neural Syst. Rehabil. Eng. **28**, 83–93 (2019)
4. Hu, J., Ma, D., Liu, C., Shi, Z., Yan, H., Hu, C.: Network security situation prediction based on MR-SVM. IEEE Access **7**, 130 937–130 945 (2019)
5. Naranjo, P.G.V., Pooranian, Z., Shojafar, M., Conti, M., Buyya, R.: FOCAN: a fog-supported smart city network architecture for management of applications in the internet of everything environments. J. Parallel Distrib. Comput. **132**, 274–283 (2019)
6. Qu, Y., Yu, S., Zhou, W., Peng, S., Wang, G., Xiao, K.: Privacy of things: emerging challenges and opportunities in wireless internet of things. IEEE Wirel. Commun. **25**(6), 91–97 (2018)
7. Qu, Y., et al.: Decentralized privacy using blockchain-enabled federated learning in fog computing. IEEE Internet of Things J. **7**, 5171–5183 (2020)
8. Forero, P.A., Cano, A., Giannakis, G.B.: Consensus-based distributed support vector machines. J. Mach. Learn. Res. **11**(May), 1663–1707 (2010)
9. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends® Mach. Learn. **3**(1), 1–122 (2011)
10. Zhang, T., Zhu, Q.: Dynamic differential privacy for ADMM-based distributed classification learning. IEEE Trans. Inf. Forensics Secur. **12**(1), 172–187 (2016)
11. Zhang, C., Ahmad, M., Wang, Y.: ADMM based privacy-preserving decentralized optimization. IEEE Trans. Inf. Forensics Secur. **14**(3), 565–580 (2018)
12. Kairouz, P., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019)
13. Mozaffari, H., Houmansadr, A.: Heterogeneous private information retrieval. In: NDSS (2020)

14. Demmler, D., Schneider, T., Zohner, M.: ABY - a framework for efficient mixed-protocol secure two-party computation. In: NDSS (2015)
15. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
16. Kolesnikov, V., Sadeghi, A.-R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 1–20. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_1
17. Beaver, D.: Efficient multiparty protocols using circuit randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_34
18. Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer and extensions for faster secure computation. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 535–548 (2013)