IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 March 2025, accepted 9 April 2025, date of publication 17 April 2025, date of current version 8 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3562053

RESEARCH ARTICLE

SVF: Support Vector Federation

MIRKO POLATO, ROBERTO ESPOSITO, AND LORENZO SCIANDRA
Computer Science Department, University of Turin, 10149 Turin, Italy

Corresponding author: Mirko Polato (mirko.polato@unito.it)

This work was supported by the spoke "FutureHPC and BigData" of the ICSC–Centro Nazionale di Ricerca in High-Performance Computing, Big Data and Quantum Computing funded by European Union–NextGenerationEU.

ABSTRACT In recent years, Federated Learning applied to neural networks has garnered significant attention, yet applying this approach to other machine learning algorithms remains underexplored. Support Vector Machines (SVMs), in particular, have seen limited exploration within the federated context, with existing techniques often constrained by the necessity to share the weight vector of the linear classifier. Unfortunately, this constraint severely limits the method's utility, restricting its application to linear feature spaces. This study addresses and overcomes this limitation by proposing an innovative approach: instead of sharing weight vectors, we advocate sharing support vectors while safeguarding client data privacy through vector perturbation. Simple random perturbation works remarkably well in practice, and indeed we provide a bound on the approximation error of the learnt model which goes to zero as the number of input features grows. We also introduce a refined technique that involves strategically moving the support vectors along the margin of the decision function, which we empirically show to slightly improve the performances. Through extensive experimentation, we demonstrate that our proposed approach achieves state-of-the-art performance and consistently enables the federated classifier to match the performance of classifiers trained on the entire dataset.

INDEX TERMS Federated learning, support vector machines, kernel method.

I. INTRODUCTION

Federated Learning (FL) [1], [2] is a collaborative and distributed Machine Learning (ML) framework developed to address the privacy challenges associated with training a global model using users' private training data, while ensuring that other users cannot access or discover them. The scenario that we will analyze here is the centralized one, where a server orchestrates the training procedure and exchanges information with the clients. Other scenarios, where the parties communicate via a different topology are possible (e.g., [3], [4], [5]).

Lately, FL has gained a lot of momentum and the research is progressing very quickly [6], [7]. Still, most of the researchers' efforts have been devoted on federating models whose training procedure is based on stochastic gradient descent (like deep neural networks [8], [9], [10]) leaving several "classical" ML techniques underexplored. In this work, we aim to partly fill this gap by proposing a novel federated version of Support Vector Machines (SVMs) [11], [12] that, differently from the state-of-the-art, enables the federation of SVMs based on any kernel function of the form $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

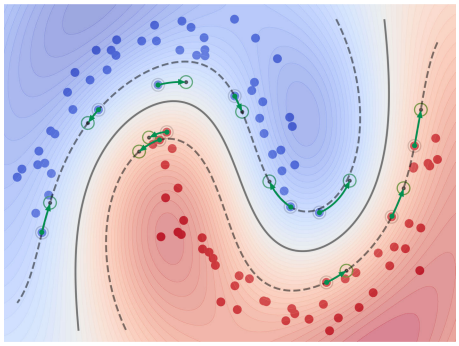


FIGURE 1. 2D visual representation of the pseudo-support vectors. The figure shows a kernel SVM learned by a client on its own dataset. The green arrows are the optimal displacement vectors, obtained by solving the minimization problem (11).

SVMs are a supervised machine learning algorithm that learns the optimal linear model by maximizing the minimum

The associate editor coordinating the review of this manuscript and approving it for publication was Jon Atli Benediktsson.

margin between the decision boundary and the examples. Upon solving the dual version of this optimization problem, it turns out that a linear combination of the support vectors (SVs), i.e. the data points with the minimum distance from the decision boundary, defines the classification hyperplane. The dual formulation also enables the use of the so-called ''kernel trick'' to efficiently learn non-linear decision boundaries in a higher-dimensional space [13], [14].

Most of the research in federating SVMs closely follows the standard approach of acquiring the federated model by averaging the models learnt by the clients solving the primal SVM problem [3], [15], [16]. This way of approaching the problem is surely sound and good in terms of communication cost; however, it constrains the learning process to a linear context where non-linear kernels cannot be used and need to be approximated, using for example Random Fourier Features (RFF) [17]. This represents a huge limitation when it comes to applying SVM with non-linear kernel functions besides kernels for which we can efficiently compute the feature map.

In this paper, we introduce Support Vectors Federation (SVF), a novel approach wherein clients exchange their SVs as a surrogate of their model instead of the actual one. The underlying rationale is that, in each round, every client enhances its understanding of the learning problem, while enabling them to use the most appropriate kernel for the task at hand. It is important to notice that these examples are private and sensitive, necessitating the addition of a privacy-preserving layer. Here, we propose to use additive noise to prevent privacy leakage. Thus, every client must find noise vectors, that, when applied to their SVs, ensure confidentiality. Also, to minimize changes in the SVM's model, we propose to compute displacements parallel to the decision boundary (Figure 1).

II. RELATED WORKS

Since its introduction in 2016 [1], FL has gained much interest from the research community. However, a significant part of the FL literature has been focused on problems related to the federation of neural network-based models [8], [9], [10], [18], [19]. Although, broadly speaking, neural networks are just the hottest topic in machine learning, there exists a non-negligible portion of application scenarios in which ''classical'' machine learning approaches are still used [20] especially when explainability plays an important role [21], [22], [23].

For this reason, we have recently witnessed a surge in the federated adaptation of non-neural machine learning approaches like decision trees and random forests [24], [25], [26], boosting techniques [27], [28], and rule-based methods [29], just to name a few. In this section, we provide an overview of the state-of-the-art on federated Support Vector Machines that closely relates to our proposal.

A preliminary attempt at developing a federated SVM is presented in [30] where the authors explore a federated ranking algorithm based on a ranking SVM pairwise loss function optimized with Rprop [31]. In terms of communication, the collaborative learning follows the typical federated pattern, but the optimization steps take place on the server. In each round, clients send the gradients of their loss function to the server. The server subsequently averages these gradients and executes a single iteration of the Rprop algorithm. While this paper is about SVMs, only the SVM ranking loss is employed within a traditional federated framework.

Polato et al. [32] introduced a method for constructing a global kernel matrix while preserving privacy through Secure Aggregation. This approach is tailored for binary data and supports any dot-product kernel. While it bears similarities with our proposed method, two key differences exist. First, the kernel machine used is not an SVM. Second, example sharing occurs via the (masked) kernel matrix. In contrast, our learning process follows the standard federated protocol, with only a subset of perturbed support vectors being exchanged.

A federated SVM has been used as a mobile packet classifier to detect personally identifiable information, ad requests, and other activities [15]. This study formulates the problem using the primal SVM approach, where each client learns a classification hyperplane. During each round, only a fraction C of clients transmit their decision boundaries to the server, which then aggregates them using a weighted average based on the number of client examples.

The approach proposed in [16] for failure prediction in a production line shares the same idea as [15], differing only in the server-side aggregation method, which employs the arithmetic mean. A slightly distinct approach is presented in [3], which addresses a decentralized scenario. The proposed method employs the Alternating Direction Method of Multipliers framework [33] to train SVMs on large datasets distributed across various organizations. Each local server learns the optimal hyperplane by considering the distance to the boundaries obtained by its neighboring servers. In contrast with the other approaches the exchange of the hyperplane parameters occurs exclusively between one-hop neighbors. The main limitation of all these techniques [3], [15], [16] is that due to the need of explicitly computing the parameters of the hyperplane, they cannot support kernel functions, i.e., they only work with the primal SVM formulation, which only allows handling problems in the input space or in a feature space that can be explicitly and efficiently computed.

Finally, Fed-KSVM, an SVM based on the RFF approximation of the RBF kernel, has been proposed with a primary focus on developing a local method for clients called 'Block Boosting' to reduce space complexity [34]. This approach primarily operates with the primal formulation of SVM, where clients transmit their learned decision hyperplanes to the server and receive an updated global version, obtained by averaging the parameters of all hyperplanes. Specifically, the optimization problem they solve is a primal-dual one, proposed in [35], aiming to minimize the impact of non-i.i.d. data distribution while reducing the distance with the global hyperplane. Although this work represents the first real effort to federate a kernel SVM, it's important to note that, by using

RFF, this approach is limited to shift-invariant kernels and cannot be generalized to other kernels for which we cannot explicitly compute the feature mapping. SVF differs from Fed-KSVM as it focuses on the exchange of surrogates for the support vectors while preserving clients' privacy, enabling learning any kernel f ($f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$) based decision function.

III. SVF

This work is about federating SVMs by sharing support vectors without compromising the privacy of the data held by clients. We will introduce briefly relevant concepts about SVMs, and then proceed to show how to compute pseudo-support vectors that can be privately shared.

A. SVMS

We assume that the federated clients are willing to cooperate in building a common (soft-margin kernel) SVM model, i.e., they are willing to construct a non-linear model by solving the dual optimization problem [11]:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \kappa(\mathbf{x}_i,\mathbf{x}_j) - \sum_{i=1}^{n}\alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C \quad \forall i \in \{1,\dots,n\} \end{aligned} \tag{1}$$

where α_i are the Lagrange multipliers, y_i the labels, $\mathbf{x}_i$ the examples, $\kappa(\cdot,\cdot)$ the kernel function and C the regularization hyperparameter. It is worth recalling that the optimal hyperplane $\mathbf{w}$ in the feature space can be expressed as a linear combination of the support vectors $\mathbf{x}_i$ after mapping them onto the feature space:

$$\mathbf{w} = \sum_{\mathbf{x}_i \in S} \alpha_i y_i \phi(\mathbf{x}_i) \tag{2}$$

where $\phi(\cdot)$ is the feature mapping function, and S is the set containing all learned support vectors. The decision function for a new example $\mathbf{x}$ is:

$$f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + b = \sum_{\mathbf{x}_i \in S} \alpha_i y_i \kappa(\mathbf{x}_i,\mathbf{x}) + b \tag{3}$$

where b is the bias term. The sign of $f(\mathbf{x})$ determines the class of $\mathbf{x}$. As a consequence of (2), $\mathbf{w}$ can be also obtained by training an SVM on S in lieu of the complete dataset.

B. SVF OVERVIEW

In contrast to previous approaches that focused on exchanging the model, our proposal focuses on the exchange of support vectors, with clients that iteratively update their local models based on the received support vectors. The primary drawback of other approaches lies in the restriction that explicit model exchange can only occur in the input space or when the feature mapping can be explicitly computed. This limitation impedes the ability to work with various kernels, either due to the infeasible computation of ϕ (as in the case of the Gaussian kernel) or because it is excessively costly (as in the case of high-degree polynomials). In fact, SVF allows the evaluation of the federated decision function f locally on the clients through the kernel function, leveraging the access to a global set of support vectors acquired during the training. The main counter-argument to this approach is that support vectors must remain private and, therefore, cannot be shared directly. To address this privacy concern, we propose introducing noise to the support vectors before transmitting them to the server.

It is worth emphasizing that random noise may be insufficient to acquire a good federated model. In fact, while this simple strategy works surprisingly well in many cases (see Section IV), it poses the risk of distorting the decision boundary [36], potentially hindering the federated model. An illustration of this problem on a toy 2D dataset is shown in Figure 2 (left), where two clients converge to decision surfaces significantly different from the centralized one. Although this risk exists, we prove that as the input dimension increases, the approximation error of an SVM trained on displaced support vectors rapidly decreases. Still, to mitigate this potential issue, we suggest incorporating noise vectors that, once mapped into the feature space, are parallel to the learned hyperplane. Although this approach may not entirely eliminate the risk of tilting the decision boundary, empirical results indicate a reduction of the problem (Figure 2 (right)). Unfortunately, determining the displacements in the input space to produce the desired translations in the feature space is challenging (unless assuming linearity of ϕ) and necessitates solving a distinct non-convex constrained optimization problem.

C. NOTATION AND ASSUMPTIONS

We assume a (cross-silo) centralized federated learning setting with a central aggregator (i.e., the *server*) and a set of clients that collaborate to learn a global SVM. The set of all clients is referred to as C. To denote that an object o belongs to a client c we will use the subscript o_c, while to indicate that o is related to a specific round t the superscript notation $o^{(t)}$ will be employed. Each client c is assumed to own a local dataset $D_c \equiv \{(\mathbf{x}_{j,c}, y_{j,c})\}_{j=1}^{n_c}$ with n_c examples. The complete dataset D is defined as:

$$D \equiv \bigcup_{c\in C} D_c \equiv \bigcup_{c\in C} \{(\mathbf{x}_{j,c}, y_{j,c})\}_{j=1}^{n_c}. \tag{4}$$

For the sake of clarity, we will refer to S as the set of support vectors obtained by training an SVM classifier on the complete dataset D, while, locally, $S_c^{(t)}$ will denote the set of support vectors for client $c \in C$ at round t. The i-th support vector of client c will be denoted as $\mathbf{x}_{i,c}$, or simply $\mathbf{x}_i$ when clear from the context. For a given support vector $\mathbf{x}$, we will use $\boldsymbol{\delta}$ to represent the displacement vector, such that $\widetilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ is the pseudo-support vector corresponding to $\mathbf{x}$.

Unfortunately, there is no straightforward relation between the ideal set of support vectors S and the corresponding sets of client support vectors $S_c^{(t)}$. Ideally, as the iterations t progress

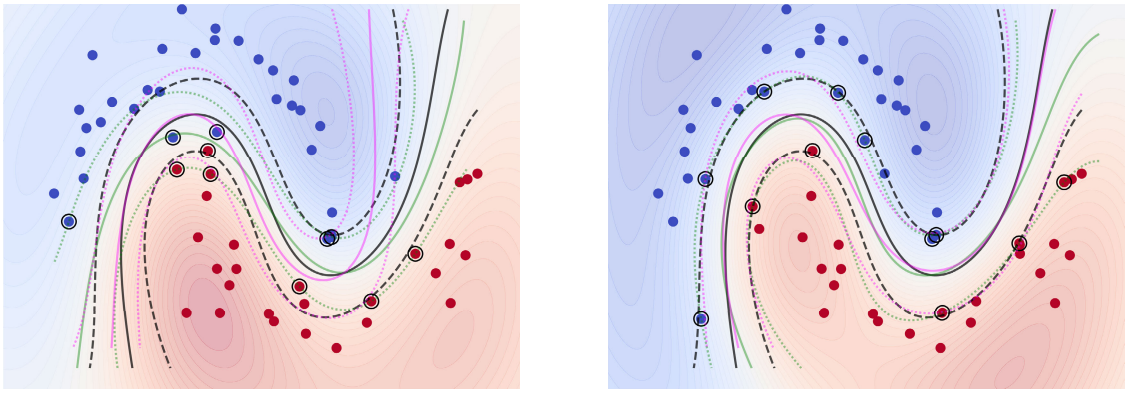


FIGURE 2. Decision boundaries learnt on the moons dataset (generated via sklearn [37]) by two federated clients exchanging support vectors according to the SVF protocol described in Section III-E. In green and magenta the boundaries learnt by the clients, in black the centralized one. The left image depicts the model of the clients using random displacements, as can be clearly observed, increases the disagreement between clients' solutions with respect to the margin displacements, showed on the right plot.

and clients exchange their pseudo support vectors $S_c^{(t)}$, if

$$S \subseteq \bigcup_t S_c^{(t)} \subseteq \bigcup_t D_c^{(t)} \qquad \forall c \in C \tag{5}$$

then $\forall c \in C$

$$\lim_{t\to\infty} \mathbf{w}_c^{(t)} = \lim_{t\to\infty} \sum_{i=1}^{|S_c^{(t)}|} y_{i,c} \cdot \alpha_{i,c} \cdot \phi(\widetilde{\mathbf{x}}_{i,c}) \tag{6}$$

$$= \lim_{t\to\infty} \sum_{i=1}^{|S_c^{(t)}|} y_{i,c} \cdot \alpha_{i,c} \cdot \phi(\mathbf{x}_{i,c} + \boldsymbol{\delta}_i) \tag{7}$$

$$\approx \sum_{i=1}^{|S|} y_i \cdot \alpha_i \cdot \phi(\mathbf{x}_i) = \mathbf{w}, \tag{8}$$

where $\mathbf{w}$ is the centralized SVM model learned on D. It is trivial to show that the last step of the derivation holds for $\boldsymbol{\delta}_i \to 0$, however, to ensure privacy, the norm of the displacement vectors must be non-zero ($\|\boldsymbol{\delta}_i\|_2 > 0$) and the equality condition becomes an approximation. Nonetheless, we argue that the clients can still learn a hypothesis close to the centralized one as long as the displacements are chosen carefully. From this perspective, it becomes evident that the primary objective is to identify suitable displacement vectors $\boldsymbol{\delta}_i$ for the clients' support vectors $\mathbf{x}_i$, so that the difference between every $\mathbf{w}_c$ and $\mathbf{w}$ is negligible and does not adversely impact the overall accuracy.

Empirically, for SVF, the relation in (5) seems to hold, as shown in Table 1. Details about the experimental setting are provided in Section IV.

D. COMPUTING THE DISPLACEMENT VECTORS

In SVF, the displacement vectors represent the key to ensure privacy. In the following, we will propose two strategies to generate such vectors. In both of these strategies, the client must select, for each displacement vector $\boldsymbol{\delta}$ that is going to generate, a secret $\bar{\delta} \in \mathbb{R}$ such that $0 < \|\boldsymbol{\delta}\|_2 \leq \bar{\delta}$. $\bar{\delta}$ controls the magnitude of the applied displacement and hence higher values of $\bar{\delta}$ can ensure more privacy at the cost of a potential loss in accuracy.

TABLE 1. Comparison of the average number of support vectors in the final model of the clients versus the number of support vectors of the model using the entire dataset.

Dataset	# SVs centralized	Avg. # SVs client No Sampling	Avg. # SVs client Sampling
skin	328	320.6	328.6
breast	65	66.9	64.5
krvskp	544	546.6	544.2
malware	170	169.4	168.4
australian	228	221.8	224.8
uwb	200	202.0	202.3

Attack Model: We consider an attack model where an attacker is a participant of the federation and thus has access to the plain (i.e., not encrypted) displaced support vectors of the other participants. Let $\mathbf{x} \in \mathbb{R}^d$ be a support vector for a client $c \in C$, and let $\boldsymbol{\delta}$ be a random displacement vector such that $\widetilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ with $0 < \|\delta\|_2 \leq \bar{\delta}$. Then, for such an attacker, that is unaware of the specific value of $\bar{\delta}$, and without any additional information, recovering $\mathbf{x}$ from $\widetilde{\mathbf{x}}$ would require searching in a d-dimensional hyper-sphere of radius m centered in $\widetilde{\mathbf{x}}$, where m is an upper-bound for $\bar{\delta}$, i.e., $0 < \bar{\delta} \leq m$.

1) RANDOM DISPLACEMENTS

Random displacement vectors, as the name suggests, are vectors with entries independently drawn from a uniform distribution. Namely, we sample $\boldsymbol{\delta}'$ uniformly from the d-dimensional sphere with radius $\bar{\delta}'$, then we set the perturbed vector $\tilde{\mathbf{x}} = (\mathbf{x} + \boldsymbol{\delta}')/\|\mathbf{x} + \boldsymbol{\delta}'\|$. By standard computations, it can be shown that the magnitude of the perturbation $\boldsymbol{\delta} = \tilde{\mathbf{x}} - \mathbf{x}$ is bounded from above by $\bar{\delta} = \frac{2\bar{\delta}'}{1-\bar{\delta}'}$. From a privacy perspective, this strategy is considered the most private, as there is no correlation between the displacement vectors, and their selection is random based on the chosen

probability distribution. This strategy is denoted as *random* in Algorithm 2. The potential issue with random displacements is that the exchanged pseudo-vectors may not accurately identify the decision surface of the client's SVM. While this effect does exist, its effects can be bounded by a term that grows linearly with the size of the displacement and goes to zero when $d \to \infty$.

Theorem 1: Let f and $\tilde{f}$ be two SVM models with Mercer's kernel κ and regularization parameter λ, trained on datasets $D = \{(\mathbf{x}, y)\}$ and $D' = \{(\mathbf{x}+\boldsymbol{\delta}, y) : (\mathbf{x}, y) \in D\}$, respectively. Assume that the data points $(\mathbf{x}, y)$ are drawn i.i.d. from the distribution P_1 over the surface $\mathcal{S}$ of the unit sphere, and that $\boldsymbol{\delta}$ is a random perturbation vector independent from $\mathbf{x}$, drawn i.i.d. from a distribution P with $\|\boldsymbol{\delta}\| \leq \bar{\delta}$, such that $\mathbf{x}+\boldsymbol{\delta} \in \mathcal{S}$. Further, assume the following conditions hold:

- *The SVM loss function and its derivative are Lipschitz continuous with Lipschitz constants ℓ and ℓ', respectively.*
- *The probability density function of P_1 is Lipschitz continuous with Lipschitz constant L.*

Then, the infinity norm of the difference between the decision functions of f and $\tilde{f}$ is bounded by:

$$\|f - \tilde{f}\|_\infty \leq 2 \cdot \frac{\ell}{\ell'} \cdot L \cdot \bar{\delta} \cdot \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)}.$$

The proof is provided in the Appendix. In Section IV, we provide empirical evidence that the proposed approach works well in practice.

2) MARGIN DISPLACEMENTS

Given that the success of the learning relies on the pseudo-support vectors, it is crucial that they are as much reliable as possible.

For this reason, we propose a second technique to compute the displacement vectors that leverage the properties of the SVM's decision surface. Specifically, we propose to move the support vectors along a direction that is perpendicular to $\mathbf{w}$, thus keeping them on the margin of the decision surface[1] (see Figure 1).

a: SINGLE δ

An initial attempt to implement margin displacement involves using the same displacement vector for all SVs. In this scenario, we can prove that the hyperplane identified by the resulting pseudo-SVs remains unchanged.

Theorem 2: Let $\mathbf{w}^\star = \sum_{\mathbf{x}_i \in S} \alpha_i y_i \phi(\mathbf{x}_i)$ denote the optimal hyperplane acquired by training a kernel SVM on a dataset D. Let D' be a dataset obtained from D, by moving all the support vectors $\mathbf{x}_i \in S \subseteq D$ according to a displacement vector $\boldsymbol{\delta}$ s.t. $(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_i + \boldsymbol{\delta})) \perp \mathbf{w}^\star$. Then, $\mathbf{w}^\star$ is an optimal hyperplane for a kernel SVM trained on D'.

The proof is provided in the Appendix.

Finding such a displacement vector $\boldsymbol{\delta}$ is trivial in the case of SVM with linear kernel: given a d-dimensional input space, we randomly generate $d-1$ entries of $\boldsymbol{\delta}$ and then we compute its last dimension as $\delta_d = \frac{\mathbf{w}_{-1} \cdot \boldsymbol{\delta}_{-1}}{w_d}$, where the notation $\mathbf{v}_{-1}$ means the vector $\mathbf{v}$ excluding its last element. Finally, we rescale $\boldsymbol{\delta}$ to ensure that its norm is equal to the client's secret $\bar{\delta}$, i.e., $\boldsymbol{\delta} \leftarrow \frac{\bar{\delta}}{\|\boldsymbol{\delta}\|_2}\boldsymbol{\delta}$. In the following, we will refer to this method as `NoOptSD` since it solves the `S`ingle `D`elta case and it does `No`t require to compute the solution of an `Opt`imization problem.

However, in the general case of SVMs with non-linear kernels, it does not usually exist a single displacement vector (in the input space) that causes all support vectors to be moved in a direction that follows the non-linear decision boundary.

b: MULTIPLE δS

Using a single margin displacement vector has the advantage of being very efficient and theoretically sound in the case of linear kernel SVM, but it fails in generalizing for any other type of kernels and it also may raise some concerns in terms of privacy, (since each client uses only one secret $\bar{\delta}$). A natural evolution is associating a different $\boldsymbol{\delta}_i$ to each support vector $\mathbf{x}_i$ (similarly to what we do in the random case). In the following, we use the notation $\boldsymbol{\Delta} \in \mathbb{R}^{|S_c| \times d}$ to refer to the matrix where each row $\boldsymbol{\Delta}_i$ is the displacement vector associated with the i-th support vector of the client c.

In the case of linear kernel, we can resort to the same procedure used for the single $\boldsymbol{\delta}$ case, by applying it to all $\boldsymbol{\Delta}_i$ independently. In the following, we will refer to this method as `NoOptMD`. Unfortunately, there is no closed form solution to this problem in the case of a non-linear kernel, and hence finding the $\boldsymbol{\Delta}_i$s requires solving an optimization problem client-side. Specifically, every client $c \in C$, after fitting the local SVM on their current D_c, must solve the following problem:

$$\begin{aligned} \min_{\boldsymbol{\Delta}} \quad & \sum_{i=1}^{|S_c|} (\|\boldsymbol{\Delta}_i\|_2 - \bar{\delta}_i)^2 \\ \text{s.t.} \quad & (\phi(\mathbf{x}_i + \boldsymbol{\Delta}_i) - \phi(\mathbf{x}_i)) \cdot \mathbf{w} = 0 \quad \forall \mathbf{x}_i \in S_c \end{aligned} \tag{9}$$

The objective function aims at ensuring that the magnitude of $\boldsymbol{\delta}_i$ complies with the client's secrets $\bar{\delta}_i$, while the set of constraints assures that the displacement vectors will move the SVs parallel to the previously learned hyperplane (i.e., in a perpendicular direction w.r.t. $\mathbf{w}$) in the kernel space. We note that in this case, at each round, the client must generate $|S_c^{(t)}|$ secrets instead of a single one. Recalling that, for a client c, $\mathbf{w}$ can be defined as in Eq. (2) then the constraints can be reformulated as:

$$\begin{aligned} & \sum_{j=1}^{|S_c|} \alpha_j y_j \phi(\mathbf{x}_j) \cdot (\phi(\mathbf{x}_i + \boldsymbol{\Delta}_i) - \phi(\mathbf{x}_i)) = \\ & \sum_{j=1}^{|S_c|} \alpha_j y_j \left[\kappa(\mathbf{x}_j, \mathbf{x}_i + \boldsymbol{\Delta}_i) - \kappa(\mathbf{x}_j, \mathbf{x}_i)\right] = 0 \quad \forall \mathbf{x}_i \in S_c \end{aligned} \tag{10}$$

where $\kappa(\cdot,\cdot)$ is the kernel function. We can now put all the constraints in the objective function with a Lagrangian

[1] This is true for all support vectors that are not margin errors. Margin errors simply moves perpendicularly w.r.t. to $\mathbf{w}$.

relaxation whose multipliers λ_i, since are free in sign, are set equal to the corresponding constraint so to obtain a convex function easy to optimize. The final version of the minimization problem that the clients must solve is:

$$\begin{aligned}\boldsymbol{\Delta}_c^* &= \arg\min_{\boldsymbol{\Delta}} \ \mathcal{L}(\boldsymbol{\Delta}, S_c) \\ &= \arg\min_{\boldsymbol{\Delta}} \ \sum_{j=1}^{|S_c|} \left(\|\boldsymbol{\Delta}_j\|_2 - \bar{\delta}_j\right)^2 + \\ &\quad \sum_{i=1}^{|S_c|} \left(\sum_{j=1}^{|S_c|} \alpha_j y_j \left[\kappa(\mathbf{x}_j, \mathbf{x}_i + \boldsymbol{\Delta}_i) - \kappa(\mathbf{x}_j, \mathbf{x}_i)\right] \right)^2 \end{aligned} \tag{11}$$

Problem (11) clearly generalizes the case with a single $\boldsymbol{\delta}$ and the linear kernel. In such a case, the problem formulation can be simplified as shown in Eq. (12):

$$\begin{aligned}\boldsymbol{\delta}_c^* &= \arg\min_{\boldsymbol{\delta}} \ \mathcal{L}(\boldsymbol{\delta}, S_c) \\ &= \arg\min_{\boldsymbol{\delta}} \ \sum_{j=1}^{|S_c|} \left(\|\boldsymbol{\delta}\|_2 - \bar{\delta}\right)^2 + \\ &\quad \sum_{i=1}^{|S_c|} \left(\sum_{j=1}^{|S_c|} \alpha_j y_j \left[\mathbf{x}_j \cdot (\mathbf{x}_i + \boldsymbol{\delta}) - \mathbf{x}_j \cdot \mathbf{x}_i\right] \right)^2 \\ &= \arg\min_{\boldsymbol{\delta}} \ \sum_{j=1}^{|S_c|} \left(\|\boldsymbol{\delta}\|_2 - \bar{\delta}\right)^2 + \sum_{i=1}^{|S_c|} \left(\sum_{j=1}^{|S_c|} \alpha_j y_j (\mathbf{x}_j \cdot \boldsymbol{\delta}) \right)^2 \end{aligned} \tag{12}$$

Problems (11) and (12) can be efficiently solved via gradient descent. In the following, we will refer to SVF with the `Opt`imized `M`ultiple $\boldsymbol{\delta}$s with `OptMD`, and to the one with the `Opt`imized `S`ingle $\boldsymbol{\delta}$ with `OptSD`.

E. SVF PROTOCOL AND COMMUNICATION

The overall federated protocol of SVF is very similar to the usual one. The learning happens in rounds where clients train a local SVM on their current dataset, and then compute the displacement vectors for the SVs that intends to send. The server, once received all the pseudo-SVs from the clients, shares them with all clients. Once a client receives the pseudo-SVs from the server, it updates its local dataset $D_c^{(t)}$ and a new round will start until convergence is reached.

The communication cost of SVF depends solely on the average number of pseudo-SVs exchanged by the clients in each round. Therefore, it is crucial to refrain from sending SVs that are no longer useful in the learning process. This is particularly applicable in cases where the same SV appears multiple times during the FL process. Consequently, each client must avoid resending the same SV more than once. Additionally, when a client receives new pseudo-SVs from the federation, these are also known by the other clients, and thus, the client should refrain from re-sending them. If new clients enter the federation, the server can share with them the overall set of support vectors collected so far.

As typical in FL, the client models in the initial rounds are generally distant from the eventual global model. Therefore, in a sense, at the start of the federation, the client models are less reliable. Building on this insight, to enhance communication efficiency without compromising the overall learning, we propose incorporating a client-side sampling step. The concept is to share, during a specific round, only a sample of all the SVs. As the learning advances and the client models approach the final state, the sample size will correspondingly increase. In Algorithm 2 we refer to the sampling with the function *sampling*. We will give implementation details about the sampling procedure in Section IV.

F. SVF ALGORITHM

In this section, we describe the overall SVF algorithm. Let's begin with the server-side algorithm, the pseudo-code of which is provided in Algorithm 1. Lines 1-3 cover the initialization phase, during which the server sends the hyperparameters of the kernel SVM to all clients. These hyperparameters include the kernel type, kernel hyperparameters, and additional parameters related to features handling, for example, type of pre-processing or the use of feature transformations like Random Fourier Features.

Lines 4 to 17 represent each federated round, where the server collects all the support vectors from the clients (lines 5-9) and then broadcasts them to all clients while avoiding sending back the same support vectors to the original client (lines 13-15). Before broadcasting the support vectors, the server checks if the procedure has converged by examining the size of the received set of support vectors. If this set is empty, it indicates that all clients have converged, and the method can stop (lines 10 to 12).

Algorithm 2 outlines the client-side SVF algorithm. In lines 1-2, the client pre-processes its examples and SVM hyperparameters based on those received from the server. The federated rounds start from line 5: first, the client c trains the local SVM on its current dataset (line 6) and then samples the support vectors ($\overrightarrow{S_c}$) to be sent (line 7). Before transmitting the SVs, the client computes the displacement vectors according to the chosen strategy (line 8: random displacements or line 9:margin displacement) and then sends the pseudo-SVs to the server (line 10). Finally, it awaits the pseudo-SVs ($\overleftarrow{S_c}$) from the server (line 11), updates its current dataset (line 13), and the set of SVs ($\overleftrightarrow{S_c}$) that should not be resent (line 12).

IV. EXPERIMENTS

In this section, we empirically evaluate SVF (with all of its variants) against different baselines. Primarily, the method with which we intend to draw comparisons is Fed-KSVM [34], as it currently stands as the most effective approach proposed in the literature for a federated version of SVMs. Since the authors of [34] did not make the code available, we attempted to replicate their experimental setting as accurately as possible, and the results reported for Fed-KSVM are taken from their paper. Finally, we compare the performance with

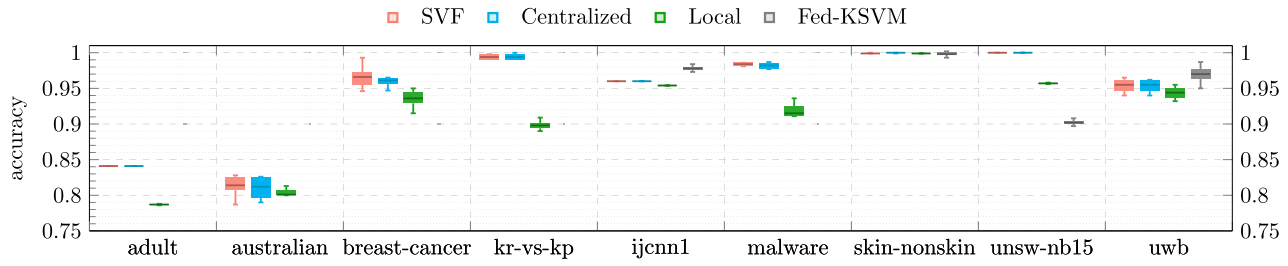


FIGURE 3. i.i.d. case. Performance achieved by SVF compared with the centralized SVM, Fed-KSVM [34], and local clients. The y-axis denotes the average accuracy, represented by the darker line inside the variance box in the segment for the quartiles.

Algorithm 1 SVF - Server

Data: Set of clients C, kernel parameters k
for *client* $c \in C$ *(in parallel)* **do**
  send_kernel_params(c, k)
end
for *round* $t \in 1$ ***to*** T **do**
  $S_t \leftarrow \emptyset$
  for *client* $c \in C$ *(in parallel)* **do**
    $S_c \leftarrow$ **receive_svs**(c)
    $S_t \leftarrow S_t \cup S_c$
  end
  if $S_t \equiv \emptyset$ **then**
    break
  end
  for *client* $c \in C$ *(in parallel)* **do**
    send_svs$(c, S_t \setminus S_c)$
  end
end
stop all clients

Algorithm 2 SVF - Client c

Data: Local dataset $D \equiv (\mathbf{X}, \mathbf{y})$
$k \leftarrow$ **receive_kernel_params**()
$\kappa, \mathbf{X} \leftarrow \text{init}(k)$
$\overleftrightarrow{S_c} \leftarrow \emptyset$ ▷ avoid resending the same SVs
$t \leftarrow 0$
while *non stop* **do**
  $S_c^{(t)}, \boldsymbol{\alpha} \leftarrow \text{SVM}(D, \kappa)$ ▷ we assume S_c contains *ys*
  $\overrightarrow{S_c} \leftarrow \text{sampling}(S_c \setminus \overleftrightarrow{S_c})$
  $\boldsymbol{\Delta}^\star \leftarrow \text{random}()$ ▷ random displacement
  $\boldsymbol{\Delta}^\star \leftarrow \arg\min_{\boldsymbol{\Delta}} \mathcal{L}(\boldsymbol{\Delta}, \overrightarrow{S_c})$ ▷ margin displacement
  send_svs(server, $\overrightarrow{S_c} + \boldsymbol{\Delta}^\star$)
  ▷ Wait for the server to send the SVs
  $\overleftarrow{S_c} \leftarrow$ **receive_svs**(server)
  $\overleftrightarrow{S_c} \leftarrow \overleftrightarrow{S_c} \cup \overleftarrow{S_c}$
  $D \leftarrow D \cup \overleftarrow{S_c}$
  $t \leftarrow t + 1$
end

the centralized SVM (using all the available data) and the client-side local SVMs. Experiments have been run on a dedicated server with 16 Intel® Xeon® cores (CPU E5-2643 @ 3.30GHz) and 128GB of RAM. The source code is available at https://anonymous.4open.science/r/svf-8BFF.

A. EXPERIMENTAL SETTINGS

In this set of experiments, we assume total participation of the clients, however, as stated in Section III-E, SVF can easily adapt to the partial participation regime. We performed our experimentation on several benchmark datasets whose characteristics are summarized in Table 2 along with the number of clients used to create the federation and the complete set of hyper-parameters used for the SVM. For all datasets, we applied the following pre-processing: (*i*) categorical features are one-hot encoded, and (*ii*) examples are standardized. When available, we adopted the training/test split provided by the dataset authors. In all other cases, we adopted a random 80-20% training-test split. In the table, we also reported the C hyper-parameter of the SVM, the number of Random Fourier Features, and the γ used for the RBF kernel. The γ hyperparameter has been always set to the default value assumed by the `scikit-learn` package (i.e., the reciprocal of the number of features $1/d$). The C value has been chosen as in the competitors papers or set to a high value such as 10 or 100 when such values were not available.

Each experiment has been repeated 5 times with different random seeds , and here we report the average accuracy along with the corresponding standard deviation. For SVF, we experimented with three kernels: linear, polynomial (degree 2), and RBF. When utilizing the linear kernel, we transform the examples using Random Fourier Features (RFF) [17] as done in [34] to approximate the RBF kernel. The number of random features was set to the value used in the competitor papers or to $1,000$ when such value was not available. In the case of random displacements, we used a fixed client secret, i.e., $\bar{\delta}_i = 0.4$, while in the margin displacement case $\bar{\delta}_i \sim \mathcal{U}(0.1, 0.4)$.

We tested both the standard i.i.d. setting, where the datasets are uniformly distributed across the clients and the non-i.i.d. covariate-shift setting. According to the methodology outlined

TABLE 2. Characteristics of the datasets used in the experiments.

Dataset	# Examples	%Test	%Pos	# Features	# Clients	C (SVM)	# RFF	γ (RBF)
adult	48,842	33	23	124	100	10	1,000	0.0081
australian	690	20	45	14	5	10	1,000	0.07
breast-cancer	569	20	37	30	10	100	1,000	0.03
kr-vs-kp	3,196	20	52	73	20	100	1,000	0.014
ijcnn1	126,701	72	10	22	5	2	9,000	0.5
malware	1,478	20	57	241	5	10	1,000	0.004
skin-nonskin	245,057	20	79	3	5	10	900	5
unsw-nb15	257,673	68	64	45	10	1	1,000	10^{-5}
uwb	663	20	50	55	3	10	400	0.018

in [34], the covariate-shift is simulated by performing k-means clustering on the dataset, with $k = |C|$, and assigning a single cluster of examples to each client.

B. SAMPLING IMPLEMENTATION

To reduce the communication cost of SVF, we implemented a sampling procedure to select a subset of pseudo-SVs to be sent by the clients. The idea is to start with a small subset of pseudo-SVs in the first rounds due to the uncertainty of the models, and then increase it up to reach no sampling in the final rounds. We tested different strategies but the one we used in our final experimentation is based on the sigmoid function and parametrized by the current round t:

$$\zeta(t) = \sigma\left(M \cdot \frac{t}{T} - \gamma\right) = \frac{1}{1 + \exp(-M \cdot \frac{t}{T} + \gamma)} \qquad (13)$$

where γ shifts the sigmoid curve to the right in order to control how quickly the function reaches its maximum w.r.t. the number of total round T, and M is a multiplicative factor that stretches the sigmoid curve and controls its slope. In all our experiments we fixed $T = M = 10$ and $\gamma = 3$. The sampling procedure at line 7 of Algorithm 2 returns a random sub-sample of S containing $\zeta(t)|S|$ examples (where S is the input of the sampling procedure).

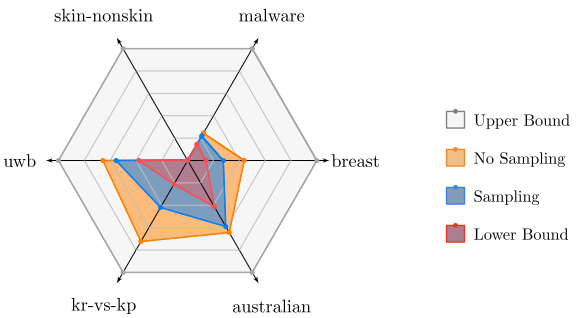


FIGURE 4. Relative communication cost.

As highlighted in Figure 4, the use of sampling drastically reduces the communication cost, with an average gain of around 22% and peaks at 42%. Additionally, Table 3 demonstrates that with sampling, on one hand, the number of exchanges is, on average, only 70% higher than the minimum required exchanges to share all the SVs necessary for a centralized SVM; on the other hand, the number of exchanges is, on average, only 33% of the size of the overall shared dataset.

C. RESULTS

1) COMPARISON WITH BASELINES

In Figure 3, a box plot presents the average accuracy of each method across all datasets in the i.i.d. case. The results reported for SVF are based on the best-performing kernel for the centralized SVM, which, across all datasets, happened to be the RBF kernel. Further details of the specific hyper-parameterization are provided in Table 2. Notably, SVF consistently achieves performance comparable to the centralized counterpart for all datasets, indicating the proper functioning of the federation. Additionally, across almost all datasets, the average performance of local SVMs is inferior to the federation (both SVF and Fed-KSVM). This underlines the beneficial impact of the federation in helping clients in enhancing their local models.

In comparison with Fed-KSVM, the assessment was conducted on four datasets (ijcnn1, skin-nonskin, unsw-nb15, and uwb) due to unavailability of their implementation. Fed-KSVM exhibits marginally superior performance on ijcnn1 and uwb. These findings are somewhat unexpected since the performances reported in [34] surpass even the performances they report for the centralized SVM with the corresponding RBF kernel (both in their reporting and in our experiments). On skin-nonskin, the performance is comparable, while on unswb-nb15,[2] SVF demonstrates a significantly higher accuracy than Fed-KSVM. However, despite our efforts to replicate the experimental conditions described in [34], it appears that certain details might be missing in this case. It is essential to emphasize that, for the sake of comparison, we adhere to the datasets and kernels utilized in [34]. It is noteworthy that SVF is more versatile and can be employed with any kernel function of the form $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$.

[2]For efficiency reasons, we randomly sampled 25% of the training set while keeping the test set unchanged.

TABLE 3. Communication costs of SVF with and without sampling. The numbers on the left are the count of exchanged SVs. UB refers to the number of vectors that would be exchanged if the whole dataset was exchanged, LB refers to the number of vectors that would be exchanged if only the SVs learnt by the centralized SVM were exchanged.

Dataset	# SVs exchanged				Ratio w.r.t. UB		Ratio w.r.t. LB	
	No sampling	Sampling	UB	LB	No sampling	Sampling	No sampling	Sampling
skin-nonskin	3680	2975	980228	1640	0.38%	0.30%	224.39%	181.40%
breast	1990	1260	4552	650	43.72%	27.68%	306.15%	193.85%
kr-vs-kp	37000	21480	51136	10880	72.36%	42.01%	340.07%	197.43%
malware	1440	1290	5912	850	24.36%	21.82%	169.41%	151.76%
australian	1775	1630	2760	1140	64.31%	59.06%	155.70%	142.98%
uwb	1044	882	1591.2	600	65.61%	55.43%	174.00%	147.00%

TABLE 4. Comparison of all methods in the i.i.d. setting. Best values are highlighted in bold (all standard deviations are ≤ 0.02).

Dataset	NoOpt MD	NoOpt SD	Opt MD	Opt SD
adult	0.8403	0.8403	0.8408	**0.8411**
australian	0.8093	0.8070	0.8125	**0.8145**
breast-cancer	0.9653	0.9640	**0.9656**	0.9609
ijcnn1	0.9379	0.9373	0.9418	0.9408
kr-vs-kp	0.9835	**0.9855**	**0.9855**	0.9851
malware	0.9802	0.9807	0.9838	**0.9843**
skin-nonskin	**0.9995**	**0.9995**	**0.9995**	**0.9995**
unsw15	0.9995	**0.9999**	0.9977	0.9580
uwb	0.9278	0.9238	0.9253	**0.9293**

2) PERFORMANCE IN THE NON-I.I.D. REGIME

We tested SVF with covariate shift non-i.i.d. distributions. Due to time constraints, we restricted this experimentation on 5 datasets of the 9 tested in other experiments. The experiments (see Table 5), do not show loss in accuracy. On the contrary, on several datasets, the performances are even better than the ones we obtain in the i.i.d. scenario. This is quite an unexpected result, observed also in [34], that would require further investigation in future works. Nonetheless, we believe that the results demonstrate that SVF is capable of managing federated scenarios with non-i.i.d. data distributions (at least for the covariate-shift case).

TABLE 5. Comparison of all methods in the non-i.i.d. setting. Best values are highlighted in bold (all standard deviations are ≤ 0.02).

Dataset	NoOpt MD	NoOpt SD	Opt MD	Opt SD
australian	0.8125	0.8200	0.8142	**0.8249**
breast-cancer	**0.9704**	0.9642	0.9619	0.9635
kr-vs-kp	0.9840	**0.9873**	0.9854	0.9854
malware	0.9823	0.9746	**0.9825**	0.9602
uwb	0.9273	0.9263	**0.9298**	0.9248

3) RANDOM DISPLACEMENTS VS MARGIN DISPLACEMENTS

In Figure 2, we showed that SVF with random displacements may incur in learning problems, as there is no assurance that the displacement vectors will not significantly alter the final hypothesis. However, this technique performs remarkably well in practice, achieving performance very close to that obtained using margin displacements (Table 6). Overall, we can observe, on average, a slightly loss in accuracy in the non-i.i.d. setting.

TABLE 6. Comparison of OptMD and Random in the non-i.i.d. setting with the RBF kernel. Best values are highlighted in bold (all standard deviations are ≤ 0.02).

Dataset	OptMD	Random
australian	**0.8180**	0.8128
breast-cancer	0.9646	**0.9691**
kr-vs-kp	**0.9941**	0.9938
malware	**0.9818**	0.9817
uwb	0.9554	**0.9559**

4) SINGLE δ VS MULTIPLE δ

The comparison between single and multiple $\boldsymbol{\delta}$ can only be conducted with linear kernel SVMs. We do not observe any significant differences in accuracy between the two techniques (Table 4). Nonetheless, in the general case, `OptMD` should be preferred because it provides increased privacy and can be applied with any type of kernel function. The only downside is its higher computational complexity due to the optimization process.

D. DISCUSSION

Overall, SVF has demonstrated performance comparable to its centralized SVM counterpart, highlighting its viability for real-world federated applications. Moreover, SVF achieves results on par with its main competitor, Fed-KSVM.

From a performance perspective, no particular SVF technique stands out as universally superior. Therefore, the choice of method should be guided by other factors, such as privacy and computational complexity. For instance, SVF with random displacement may be preferable for enhanced privacy, as it can be interpreted as a form of differential privacy. Similarly, SVF with multiple δ values provides stronger privacy guarantees compared to a single δ. On the other hand, when computational efficiency is a concern, the `NoOpt` variant of SVF is the recommended choice.

It is worth mentioning that, while some concerns may arise regarding the effect of combining support vectors learned from clients in highly skewed scenarios, the experimental results presented suggests that the presented technique is quite robust with respect to these issues.

A key advantage of SVF over Fed-KSVM is its flexibility in supporting any kernel of the form $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, making it applicable in scenarios where custom kernels are more suitable than the standard RBF kernel.

It is important to note that SVF is not positioned as a competitor to deep neural network-based approaches but rather as an alternative for specific scenarios. For example, when an organization already has a well-performing SVM-based model in place, SVF allows for a seamless transition to a federated setting without the need for a complete system overhaul. This decision may be driven by cost considerations or technical factors, such as the reliance on years of feature engineering. In such cases, replacing the existing model with a neural network could result in reduced data efficiency and diminished interpretability for human experts.

V. CONCLUSION

This paper introduces a novel federated SVM approach, SVF, which enables practitioners to harness the complete potential of kernel functions without approximations or explicit mapping computation. Our experiments demonstrate competitive performance, matching centralized results in most cases.

While Federated Learning has gained significant traction in the research community, most existing approaches are based on neural networks. However, leveraging SVMs in FL presents a compelling alternative, particularly in domains where tabular data is prevalent, and high-quality features have been meticulously engineered over years of expert work. In such cases, replacing an effective SVM with a neural network may not always be desirable, as it could require more training data and obscure domain-specific insights captured by the existing model.

FL is especially valuable in scenarios where data scarcity necessitates collaboration to train robust models. Therefore, transitioning to a more data-hungry architecture is only justified if the additional information gained from federated partners sufficiently compensates for the loss of data efficiency.

Given these considerations, we believe the proposed approach could have a significant impact, enabling the development of superior models in data-constrained environments where tabular data dominates. We believe that the theory and methods introduced in this paper are conductive to several promising directions of future research. From a theoretical perspective, we aim to: (a) strengthen the formal guarantees on data privacy and (b) rigorously analyze the robustness of our techniques in highly non-i.i.d. settings. Regarding improvements to the proposed FL method, we plan to explore schemes where the server takes an active role in training an SVM model using the received displaced support vectors.

APPENDIX

PROOF THEOREM 1

Theorem 1: Let f and $\tilde{f}$ be two SVM models with Mercer's kernel κ and regularization parameter λ, trained on datasets $D = \{(\mathbf{x}, y)\}$ and $D' = \{(\mathbf{x}+\boldsymbol{\delta}, y) : (\mathbf{x}, y) \in D\}$, respectively. Assume that the data points $(\mathbf{x}, y)$ are drawn i.i.d. from the distribution P_1 over the surface $\mathcal{S}$ of the unit sphere, and that $\boldsymbol{\delta}$ is a random perturbation vector independent from $\mathbf{x}$, drawn i.i.d. from a distribution P with $\|\boldsymbol{\delta}\| \leq \bar{\delta}$, such that $\mathbf{x}+\boldsymbol{\delta} \in \mathcal{S}$. Further, assume the following conditions hold:

- *The SVM loss function and its derivative are Lipschitz continuous with Lipschitz constants ℓ and ℓ', respectively.*
- *The probability density function of P_1 is Lipschitz continuous with Lipschitz constant L.*

Then, the infinity norm of the difference between the decision functions of f and $\tilde{f}$ is bounded by:

$$\|f - \tilde{f}\|_\infty \leq 2 \cdot \frac{\ell}{\ell'} \cdot L \cdot \bar{\delta} \cdot \frac{\pi^{\frac{d}{2}}}{\Gamma\left(\frac{d}{2}\right)}.$$

Proof:

By a special case of Theorem 2.7 in [38], we have:

$$\|f - \tilde{f}\|_\infty \leq 2 \cdot \frac{\ell}{\ell'} \cdot \|P_1 - P_2\|_{tv},$$

where $\|\cdot\|_{tv}$ is the total variation norm and P_2 is the distribution of the perturbed examples $\mathbf{x}+\boldsymbol{\delta}$. By the convolutional formula for the sum of independent random variables, we have that:

$$p_2(\mathbf{x}) = \int p_1(\mathbf{x}-\boldsymbol{\delta}) \cdot p(\boldsymbol{\delta})\, d\boldsymbol{\delta} = \mathbb{E}_{\boldsymbol{\delta}}[p_1(\mathbf{x}-\boldsymbol{\delta})],$$

where p_1, p_2 and p are the probability density functions of P_1, P_2 and P distributions, respectively. The total variation distance can be computed as [39]:

$$\|P_1 - P_2\|_{tv} = \frac{1}{2} \int_{\mathcal{S}} |p_1(\mathbf{x}) - p_2(\mathbf{x})|\, d\mu(\mathbf{x}),$$

where μ is the measure of the sample space. This yields:

$$\begin{aligned}
\|P_1 - P_2\|_{tv} &= \frac{1}{2} \int_{\mathcal{S}} |p_1(\mathbf{x}) - \mathbb{E}_{\boldsymbol{\delta}}[p_1(\mathbf{x}-\boldsymbol{\delta})]|\, d\mu(\mathbf{x}) \\
&= \frac{1}{2} \int_{\mathcal{S}} |\mathbb{E}_{\boldsymbol{\delta}}[p_1(\mathbf{x}) - p_1(\mathbf{x}-\boldsymbol{\delta})]|\, d\mu(\mathbf{x}) \\
&\leq \frac{1}{2} \int_{\mathcal{S}} \mathbb{E}_{\boldsymbol{\delta}}[|p_1(\mathbf{x}) - p_1(\mathbf{x}-\boldsymbol{\delta})|]\, d\mu(\mathbf{x}) \\
&\leq \frac{1}{2} \int_{\mathcal{S}} \mathbb{E}_{\boldsymbol{\delta}}[L \cdot \|\boldsymbol{\delta}\|]\, d\mu(\mathbf{x}) \\
&\qquad \text{(by Lip. cont. of } P_1\text{)} \\
&\leq \frac{L}{2} \cdot \bar{\delta} \cdot \mu(\mathcal{S}).
\end{aligned}$$

Since the data points are normalized to lie on the surface of a d-dimensional hypersphere, the measure $\mu(\mathcal{S})$ is the surface area of the hypersphere, given by $\frac{2\pi^{d/2}}{\Gamma(d/2)}$. Thus,

$$\|P_1 - P_2\|_{tv} \leq \frac{L}{2} \cdot \bar{\delta} \cdot \frac{2\pi^{d/2}}{\Gamma(d/2)} = L \cdot \bar{\delta} \cdot \frac{\pi^{d/2}}{\Gamma(d/2)}.$$

Substituting this back into our bound on $\|f - \tilde{f}\|_\infty$, we get:

$$\|f - \tilde{f}\|_\infty \leq 2 \cdot \frac{\ell}{\ell'} \cdot \|P_1 - P_2\|_{tv} \leq 2 \cdot \frac{\ell}{\ell'} \cdot L \cdot \bar{\delta} \cdot \frac{\pi^{d/2}}{\Gamma(d/2)},$$

which completes the proof. □

APPENDIX
PROOF THEOREM 2

Theorem 2: Let $\mathbf{w}^\star = \sum_{\mathbf{x}_i \in S} \alpha_i y_i \phi(\mathbf{x}_i)$ *denote the optimal hyperplane acquired by training a kernel SVM on a dataset D. Let* D' *be a dataset obtained from D, by moving all the support vectors* $\mathbf{x}_i \in S \subseteq D$ *according to a displacement vector* $\boldsymbol{\delta}$ *s.t.* $(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_i + \boldsymbol{\delta})) \perp \mathbf{w}^\star$. *Then,* $\mathbf{w}^\star$ *is an optimal hyperplane for a kernel SVM trained on* D'.

Proof: Without loss of generality, we provide a proof for the case of hard-margin SVM. The same considerations apply to the soft-margin SVM case. In the following, we will demonstrate that the subset of all support vectors S remains unchanged when we apply $\boldsymbol{\delta} \perp \mathbf{w}^\star$. By definition, S contains all the points with functional margin equal to 1:

$$y_i(\mathbf{w}^\star \cdot \phi(\mathbf{x}_i) + b) = 1 \qquad \forall \mathbf{x}_i \in S$$

Now, by adding $\boldsymbol{\delta} \perp \mathbf{w}^\star$ to all the support vectors we have:

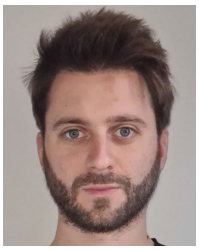
$$\begin{aligned} & y_i(\mathbf{w}^\star \cdot \phi(\mathbf{x}_i + \boldsymbol{\delta}) + b) = \\ & = y_i(\mathbf{w}^\star \cdot [\phi(\mathbf{x}_i) + \phi(\mathbf{x}_i + \boldsymbol{\delta}) - \phi(\mathbf{x}_i)] + b) \\ & = y_i(\mathbf{w}^\star \cdot \phi(\mathbf{x}_i) + \underbrace{\mathbf{w}^\star \cdot (\phi(\mathbf{x}_i + \boldsymbol{\delta}) - \phi(\mathbf{x}_i))}_{=0} + b) \\ & = y_i(\mathbf{w}^\star \cdot \phi(\mathbf{x}_i) + b) = 1. \end{aligned}$$

Thus, the displaced support vectors are still support vectors. At the same time, no example that was not already in S becomes a new support vector, since they are not moved and their functional margin is by definition greater than 1. Given that, as said in (2), the optimal hyperplane $\mathbf{w}^\star$ is a linear combination of support vectors, and S does not change, $\mathbf{w}^\star$ can be obtained from the pseudo-support vectors $\widetilde{\mathbf{x}}_i = \mathbf{x}_i + \boldsymbol{\delta}$. □

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, Jan. 2016, pp. 1273–1282.
[2] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3347–3366, Apr. 2023.
[3] S. Tavara, A. Schliep, and D. Basu, "Federated learning of oligonucleotide drug molecule thermodynamics with differentially private ADMM-based SVM," in *Proc. Int. Workshops Mach. Learn. Princ. Pract. Knowl. Discovery Databases*. Cham, Switzerland: Springer, Jan. 2021, pp. 459–467.
[4] I. Hegedűs, G. Danner, and M. Jelasity, "Decentralized learning works: An empirical comparison of gossip learning and federated learning," *J. Parallel Distrib. Comput.*, vol. 148, pp. 109–124, Feb. 2021.
[5] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.
[6] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jun. 2021.
[7] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowl.-Based Syst.*, vol. 216, Jan. 2021, Art. no. 106775.
[8] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, H. Daumé and A. Singh, Eds., Oct. 2019, pp. 5132–5143.
[9] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Represent.*, Jan. 2020. [Online]. Available: https://www.proceedings.com/75296.html
[10] T. Li, A. Kumar Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.
[11] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
[12] M. A. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Schölkopf, "Support vector machines," *IEEE Intell. Syst. Appl.*, vol. 13, no. 4, pp. 18–28, Jul. 1998.
[13] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[14] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *Ann. Statist.*, vol. 36, no. 3, pp. 1171–1220, May 2008.
[15] E. Bakopoulou, B. Tillman, and A. Markopoulou, "FedPacket: A federated learning approach to mobile packet classification," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3609–3628, Oct. 2022.
[16] N. Ge, G. Li, L. Zhang, and Y. Liu, "Failure prediction in production line based on federated learning: An empirical study," *J. Intell. Manuf.*, vol. 33, no. 8, pp. 2277–2294, Dec. 2022.
[17] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 20, Dec. 2007, pp. 1177–1184.
[18] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA, Jan. 2020, pp. 7611–7624.
[19] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Los Alamitos, CA, USA, Jun. 2021, pp. 10708–10717.
[20] F. D'Ascenzo et al., "Machine learning-based prediction of adverse events following an acute coronary syndrome (PRAISE): A modelling study of pooled datasets," *Lancet*, vol. 397, no. 10270, pp. 199–207, Jan. 2021.
[21] M. Polato and F. Aiolli, "Boolean kernels for rule based interpretation of support vector machines," *Neurocomputing*, vol. 342, pp. 113–124, May 2019.
[22] R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke, "Explainable machine learning for scientific insights and discoveries," *IEEE Access*, vol. 8, pp. 42200–42216, 2020.
[23] P. Whig, S. Kouser, A. B. Bhatia, R. R. Nadikattu, and P. Sharma, "Explainable machine learning in healthcare," in *Explainable Machine Learning for Multimedia Based Healthcare Applications*. Cham, Switzerland: Springer, 2023, pp. 77–98.
[24] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, "Federated forest," *IEEE Trans. Big Data*, vol. 8, no. 3, pp. 843–854, Jun. 2022.
[25] A.-C. Hauschild, M. Lemanczyk, J. Matschinske, T. Frisch, O. Zolotareva, A. Holzinger, J. Baumbach, and D. Heider, "Federated random forests can improve local performance of predictive models for various healthcare applications," *Bioinformatics*, vol. 38, no. 8, pp. 2278–2286, Apr. 2022.
[26] S. Kalloori and S. Klingler, "Cross-silo federated learning based decision trees," in *Proc. 37th ACM/SIGAPP Symp. Appl. Comput.*, New York, NY, USA, Apr. 2022, pp. 1117–1124.
[27] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, Apr. 2020, pp. 4642–4649.
[28] M. Polato, R. Esposito, and M. Aldinucci, "Boosting the federation: Cross-silo federated learning without gradient descent," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–10.
[29] Y. H. Pang, H. Zhang, J. D. Deng, L. Peng, and F. Teng, "Rule-based collaborative learning with heterogeneous local learning models," in *Proc. 26th Pacific-Asia Conf. Adv. Knowl. Discovery Data Mining*, Chengdu, China. Cham, Switzerland: Springer, Jan. 2022, pp. 639–651.
[30] F. Hartmann, S. Suh, A. Komarzewski, T. D. Smith, and I. Segall, "Federated learning for ranking browser history suggestions," in *Proc. Int. Workshop Federated Learn. User Privacy Data Confidentiality Conjunct NeurIPS (FL-NeurIPS)*, Jan. 2019.
[31] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 1, May 1993, pp. 586–591.
[32] M. Polato, A. Gallinaro, and F. Aiolli, "Privacy-preserving kernel computation for vertically partitioned data," in *Proc. ESANN*, 2021, pp. 11–16.
[33] S. Boyd, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[34] X. Zhou and X. Wang, "A memory-efficient federated kernel support vector machine for edge devices," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 12, pp. 17359–17371, Dec. 2024.
[35] M. Jaggi, V. Smith, M. Takáč, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, vol. 2, Cambridge, MA, USA, Dec. 2014, pp. 3068–3076.
[36] H.-X. Li, J.-L. Yang, G. Zhang, and B. Fan, "Probabilistic support vector machines for classification of noise affected data," *Inf. Sci.*, vol. 221, pp. 60–71, Feb. 2013.
[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Jul. 2011.
[38] A. Christmann, D. Xiang, and D.-X. Zhou, "Total stability of kernel methods," *Neurocomputing*, vol. 289, pp. 101–118, May 2018.
[39] J. Duchi, "Lecture notes on statistics and information theory," Dept. Statist., Stanford Univ., Stanford, CA, USA, Tech. Rep., Dec. 2023.



MIRKO POLATO received the M.Sc. and Ph.D. degrees in brain, mind, and computer science from the University of Padova, Italy, in 2013 and 2018, respectively. He is an Assistant Professor with the Department of Computer Science, University of Turin. He has authored around 50 research products, including international peer-reviewed conferences and journal articles. His research mainly focuses on federated learning and interpretable machine learning. He served as a program committee member for several international conferences and a referee for several international journals. More information about him can be found at https://makgyver.github.io

ROBERTO ESPOSITO is an Associate Professor with the Machine Learning Group, Computer Science Department, University of Turin. He has more than 20 years of research experience in the field of machine learning. He has published more than 70 research products, including articles in prestigious journals (JMLR, *Information Systems*, *Neurocomputing*, *Nature Methods*, and IEEE Access) and conferences (ICML and IJCAI). His current research interests include non-functional aspects of machine learning (e.g., fairness and explainability) and the theory of neural networks and federated learning.

LORENZO SCIANDRA received the B.Sc. and M.Sc. degrees in computer science from the University of Turin, in 2020 and 2023, respectively, where he is currently pursuing the Ph.D. degree in modeling and data science with the Department of Mathematics. His primary research interests include the intersection of classical and combinatorial optimization with deep and machine learning methodologies. More information about him can be found at https://www.lorenzosciandra.com/