FedDelta: Incremental federated learning for heterogeneous data using dynamic leader election

Rafael García-Luque[a,*], Ernesto Pimentel[a], Francisco Durán[a], Ivan Iroslavov[a] and Emilio R. Carreira[a]

[a]ITIS Software, Universidad de Málaga, Málaga, 29071, Spain

ARTICLE INFO

Keywords:
Federated Learning
Ridge Regression
Peer-to-Peer Systems
Dynamic Leader
Closed-form Solution

ABSTRACT

Federated learning enables collaborative model training across multiple devices or organizations without sharing raw data, thereby addressing privacy and data ownership concerns. However, most existing federated approaches rely on centralized coordination and often struggle to maintain robustness and convergence stability in scenarios with heterogeneous or unbalanced data. In this work, we propose FedDelta, a federated multivariable linear regression method that achieves strong performance across varying data quantities and distributions, including scenarios with unbalanced data across participants. FedDelta employs a closed-form ridge regression solution and a decentralized communication scheme, in which participating peers dynamically assume coordination roles to aggregate model updates efficiently, eliminating the need for a central server. We evaluate FedDelta on real-world datasets and show that it achieves competitive accuracy and robustness compared to centralized and traditional federated methods. Our results highlight FedDelta's potential for privacy-preserving, scalable learning in resource-constrained and distributed environments.

1. Introduction

Federated learning (FL) has become a cornerstone for privacy-preserving machine learning, enabling multiple clients to collaboratively train models without sharing raw data [1]. This paradigm is especially critical in sensitive domains such as healthcare, where regulations and ethical concerns prohibit direct data centralization. For example, hospitals may wish to jointly develop predictive models for rare diseases, but each institution holds only a small, heterogeneous dataset, making traditional centralized or gradient-based FL approaches unreliable.

Despite its promise, existing FL methods face significant challenges in real-world deployments. Most notably, they struggle with statistical heterogeneity—the non-independent and identically distributed (non-IID) nature of client data [2]—which induces client drift [3], slows convergence, and degrades model performance. These difficulties are often exacerbated by the reliance on a central aggregation server, which introduces a single point of failure and communication bottlenecks, limiting scalability and robustness, especially in dynamic environments. Finally, most approaches utilize gradient descent (GD)-based optimization tailored for large datasets; in small-sample regimes, these methods suffer from high variance in gradient estimates, unstable convergence, and sensitivity to hyperparameters such as learning rate and batch size [4].

To address these limitations, we introduce FedDelta, a closed-form FL algorithm for multivariable linear regression. Unlike prior FL methods, FedDelta leverages the exact ridge regression solution, incrementally updating global model parameters via additive sufficient statistics rather than iterative gradient steps. This approach eliminates the instability of GD-based methods in small or heterogeneous datasets and enables precise, one-shot updates. FedDelta further reduces computational overhead by leveraging the Sherman-Morrison-Woodbury (SMW) identity and spectral decompositions to efficiently invert matrices, making it practical for high-dimensional settings.

FedDelta operates in a serverless architecture utilizing dynamic leader election, where the aggregation role rotates among clients. This mechanism removes the reliance on a fixed central server, preserving the exactness of global coordination while significantly enhancing fault tolerance.

Our empirical results demonstrate that FedDelta outperforms iterative baselines—namely, FedAvg [1], FedProx [5], and FedNova [6]—by calculating the exact analytical optimum rather than approximating it. This approach yields a 40% reduction in time-to-solution and a 5–10% improvement in accuracy, particularly in challenging scenarios with high statistical heterogeneity and unbalanced data distributions.

Our main contributions are: (i) We propose a closed-form, peer-to-peer FL algorithm for multivariable linear regression, FedDelta, tailored for heterogeneous and unbalanced data distributions where gradient-based FL methods are unreliable; (ii) *efficient matrix updates*, since FedDelta uses the SMW identity and spectral decompositions to incrementally update model parameters, significantly reducing computational costs; (iii) a *decentralized architecture*, with dynamic leader election, FedDelta eliminates central server dependencies, improving scalability and fault tolerance; and (iv) a *comprehensive evaluation*, by providing a theoretical analysis and extensive experiments, demonstrating that FedDelta outperforms standard FL baselines in computational efficiency, stability, and accuracy.

The remainder of this paper is organized as follows. Section 2 reviews related work and provides context to it.

*Corresponding author
rafagarcialuque@uma.es (R. García-Luque)
ORCID(s): 0009-0002-1216-0995 (R. García-Luque)

Section 3 introduces key concepts the work relies on. Section 4 formalizes the problem and system model. Section 5 details the FedDelta algorithm and analyzes its complexity. Section 6 presents empirical results. Sections 7 and 8 discuss findings and conclude the paper. The implementation of the algorithm and all other resources, including datasets used and some experimental evaluation, is available at https://github.com/scenic-uma/feddelta.

2. Related work

This section reviews existing aggregation methods in federated learning and discusses non-centralized architectures. It concludes highlighting how our approach advances the state of the art by combining closed-form aggregation with a decentralized, adaptive, and efficient framework.

Aggregation methods

FL relies on aggregating local updates from participating devices while preserving privacy. Traditionally, this aggregation is performed using the standard arithmetic mean aggregation method, FedAvg [1]. However, this technique is not robust against corrupted updates—whether due to adversarial attacks or failures in low-cost hardware. In fact, a single corrupted update in one round can degrade the global model for all devices [7]. Additionally, this method faces other challenges such as communication efficiency [8], non-IID data [9], and privacy protection [10].

These challenges become even more pronounced in real-world scenarios, where billions of clients may participate [11]. Consequently, the data collected by each client may vary significantly due to personal preferences [2] or geographical location [12], among others. This leads to inherent statistical heterogeneity in distributed data, causing variations in categories, domains, and dataset sizes among clients. This bias induces client drift [3], which causes local models to converge toward different minima, deviating from the global direction and resulting in noisy, unstable learning trends [13].

Several solutions have been proposed to address these issues. Some strategies limit local model drift through regularization techniques. E.g., FedProx [5] introduces a penalty term in local objectives to prevent divergence from the global model. FedNova [6] suggests aggregating normalized trained gradients instead of raw gradients to compute the global model. Scaffold [3] uses stochastic variance reduction [14] to align local and global updates. Similarly, FedDyn [15] aligns local and global stationary points at convergence, providing the same convergence guarantees as Scaffold.

Other approaches aim to reduce client drift by leveraging historical updates through momentum [16]. Specifically, FedAvgM [17] incorporates momentum in server-side aggregation, demonstrating effectiveness in realistic settings [12]. Another work introduces client-side momentum [18] to guide local updates in the direction of the global model. Although these methods are theoretically principled, they exhibit inherent instabilities in real-world cross-device FL scenarios [19]. Among these issues, they are often prone to parameter explosion [20] and highly sensitive to data partitioning schemes, which can significantly impact performance under extreme non-IID conditions [21].

As an alternative, closed-form solutions based on exact aggregation have emerged, offering substantial advantages in communication efficiency and convergence speed over gradient-based optimization methods. These methods do not suffer from statistical heterogeneity and remain invariant to the type of federated partitioning [19]. In this line, notable methods include Privacy-Preserving Distributed Ridge Regression (PPRR) by Chen et al. [22], which proposes a privacy-preserving centralized evaluation protocol using homomorphic encryption. In PPRR, each party submits encrypted versions of its local covariance matrix, and the evaluator computes the global ridge solution without accessing raw data. Similarly, the algorithm LOCO [23] distributes features among workers and uses random projections to preserve the covariance structure. Each node builds a local ridge model with reduced communication while maintaining approximation quality. Finally, Fed3R [19] proposes an FL algorithm based on a closed-form ridge regression classifier on top of pre-trained features, where each client computes local matrices and sends them to a central server. The server aggregates them to solve the global ridge equation in one shot. Fed3R is communication-efficient and robust to client heterogeneity.

Therefore, closed-form approaches have mainly focused on linear regression, ridge, and lasso, leveraging their analytical structure to improve communication efficiency and convergence speed. However, existing methods present significant limitations, as most are centralized, non-incremental, and poorly adaptable to dynamic conditions. These constraints highlight the need for FL systems that are decentralized, adaptive, and efficient, capable of directly addressing real-world challenges such as statistical heterogeneity, client asynchrony, and large-scale distributed environments.

Decentralized architectures

Adopting a P2P architecture can reduce the total volume of data transmitted across the network, lowering communication time and costs. Additionally, it mitigates the potential risk of a single point of failure [24]. Another key advantage is that the aggregation protocol can vary by client, allowing for personalized update strategies based on data similarity or statistical divergence [25].

As a result, several proposals have emerged. E.g., Gossip Learning [26] enables direct client-to-client communication within the network, allowing them to share model parameters using mechanisms such as peer-sampling services or random walks. Similarly, in [27], Lalitha et al. propose a distributed learning approach in which devices jointly aggregate model data with neighboring devices to estimate global model parameters. Another notable contribution is BrainTorrent [28], a framework that operates without a central server in a collaborative environment, enhancing the system's resilience to failures.

In the context of personalization and communication efficiency, in [29], Liu et al. introduced DFedPGP, a framework for decentralized FL. Unlike traditional undirected and symmetric topologies, DFedPGP uses a directed communication topology, allowing clients to select neighbors for update exchanges flexibly. The framework adopts partial model personalization by sharing only feature extraction layers as global parameters while keeping the linear classifier local, reducing communication costs and improving privacy. However, this approach relies on a strict separation of shared and private parameters, which may not be optimal for all model types or data distributions.

Other studies have focused on security and vulnerability aspects. For instance, in [30], Syros et al. introduced backdoor attacks in P2P-FL, exploiting graph structural properties to place malicious nodes strategically. Similarly, DEFEAT [31] proposes a decentralized FL framework to defend against gradient attacks, which allows malicious servers to reconstruct sensitive client data from model gradients.

Beyond secure exchange, a fundamental aspect of decentralized architectures is determining who coordinates the aggregation of local updates. Approaches such as [32] combine FL with permissioned blockchain to enable privacy-preserving collaborative learning among distributed nodes. A key feature of Swarm Learning is dynamic leader election through blockchain to coordinate parameter aggregation. In the same way, in [33], Behera et al. propose a leader selection based on a distributed consensus mechanism, such as the Raft algorithm or lightweight election variants, where candidate nodes nominate themselves and others vote based on criteria like latency, computational capacity, or availability. Leadership is temporary and periodically renewed, ensuring fault tolerance and load balancing while preventing any single node from becoming a bottleneck.

Our approach: decentralized, adaptive, and efficient FL

Unlike previous methods, our solution combines the strengths of closed-form aggregation with a robust, serverless architecture. While most existing closed-form approaches are centralized and require a trusted server, our method eliminates the single point of failure by distributing aggregation responsibilities among clients. This design enables dynamic leader election and supports asynchronous participation, making it robust to client dropouts and network variability.

Furthermore, our approach is inherently adaptive: it accommodates heterogeneous data distributions and client capabilities without sacrificing convergence speed or communication efficiency. By leveraging analytical aggregation, we avoid the instability and parameter explosion issues common in gradient-based methods under extreme non-IID conditions. In contrast to prior work, our protocol is incremental and scalable, supporting large-scale deployments and dynamic environments where clients may join or leave at any time.

In summary, our solution advances the state of the art by providing a decentralized, adaptive, and communication-efficient FL framework that directly addresses the limitations of both centralized closed-form and traditional gradient-based methods.

3. Preliminaries

In this section, we first introduce the basic concepts and notation related to FL, followed by a review of the ridge regression model. Then, we discuss how ridge regression can be reformulated using sufficient statistics to enable efficient and privacy-preserving computation in distributed environments.

3.1. Federated learning

In centralized FL systems, a set of K nodes collaborate with a server during each iteration of the training process. Each node $i \in \{1, ..., K\}$ has access to its own local dataset D_i, which remains private. Based on this private data, each node independently trains a local model M_i.

Once local training is complete, the models obtained by each node, $M_1, M_2, \dots, M_K$, are sent to the server instead of sharing the local datasets. Cryptographic techniques are commonly used to protect sensitive information contained in the models before uploading them, notably Secure Multi-Party Computation (SMPC) [34], Differential Privacy (DP) [35], and Homomorphic Encryption (HE) [36]. However, we focus here on exactness, incrementality, and robustness.

The server applies an aggregation algorithm such as FedAvg, FedDyn, etc. (see Section 2), denoted as $\mathrm{Agg}(\cdot)$, to combine the nodes' local models into a global model $M = \mathrm{Agg}(M_1, M_2, ..., M_K)$. This operation completes one round, after which the global model M is distributed to the nodes to continue local training in the next iteration. The total number of rounds is typically determined based on model performance, stopping the process once a desired criterion is met.

Based on this, the objective of FL is to minimize the global loss function $L(W)$, which, as in [37], can be expressed as:

$$L(W) = \sum_{i=1}^{K} f_i L_i(W) \tag{1}$$

where $W \in \mathbb{R}^{d\times 1}$ represents the model parameters and $L_i(W)$ is the local loss function for node i. The coefficients f_i reflect the relative importance of each node in the aggregation, subject to the constraint $\sum_{i=1}^{K} f_i = 1$. This relative importance f_i can be determined by the aggregation algorithm employed.

As already explained, in FedDelta, any of the participants may assume the role of model aggregator, or leader, in each of the iterations of the training process.

3.2. Ridge regression model

Consider a supervised and centralized learning model with a dataset composed of n samples, each with d features, represented by the matrix $X \in \mathbb{R}^{n\times d}$. The associated target values are collected in the matrix $Y \in \mathbb{R}^{n\times 1}$. However, depending on the nature of the dataset, these features can often lead to overfitting [38], or multicollinearity. This phenomenon can distort the estimation of regression coefficients, inflate their standard errors, reduce statistical power, and degrade the predictability of the model's predictions [39]. To address these challenges, as in [40], a regularization term, controlled by a Tikhonov hyperparameter $\lambda \in \mathbb{R}^+$, is included in the loss function:

$$L(W) = \|Y - XW\|^2 + \lambda\|W\|^2 \tag{2}$$

where W, as above, represents the vector of model parameters.

According to the theory of least-squares, the closed-form for the optimal model parameter W^* which minimizes $L(W)$ can be expressed as:

$$W^* = (X^\top X + \lambda I_d)^{-1} X^\top Y \tag{3}$$

where I_d denotes the $d \times d$ identity matrix. The regularization parameter must satisfy $\lambda > 0$, which ensures that $X^\top X + \lambda I_d$ is positive definite (PD) and thus always invertible.

3.3. Reformulation using sufficient statistics

The closed-form ridge estimator in Equation (3) assumes that all data are available at once and therefore cannot be used directly when each node receives them sequentially. Recursive ridge regression methods provide this capability by exploiting the linear structure of Equation (3) to incorporate incoming data without recomputing the solution from scratch [41].

The SMW formula or Cholesky updates are commonly employed to address this and efficiently update the inverse of a matrix incrementally under low-rank modifications [42]. Despite this, these approaches require direct access to each sequential data batch to compute the updated matrices and, therefore, are not suitable for privacy-constrained settings in decentralized environments [43]. To overcome these limitations, we adopt the Fed3R approach [19], which avoids the need for raw data access by reformulating ridge regression through additive sufficient statistics. Consequently, Equation (3) is rewritten as:

$$W^* = (A + \lambda I_d)^{-1} B, \tag{4}$$

where $A \in \mathbb{R}^{d\times d}$ is the covariance matrix of the input data, and $B \in \mathbb{R}^{d\times 1}$ is the global cross-covariance matrix between inputs and targets. These matrices are obtained by summing the local contributions from each client:

$$A = \sum_{i=1}^{K} X_i^\top X_i, \tag{5}$$

$$B = \sum_{i=1}^{K} X_i^\top Y_i, \tag{6}$$

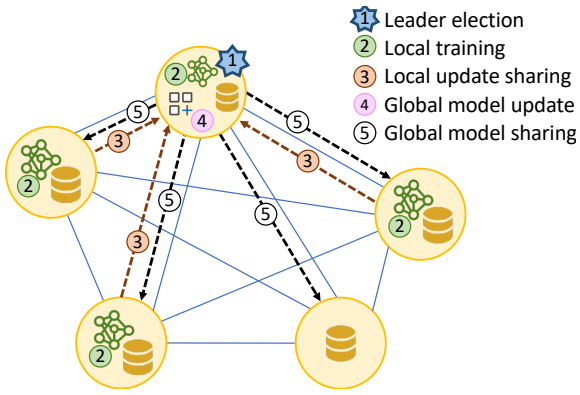


Figure 1: Training process in the proposed P2P-FL system model.

where X_i and Y_i are the local input and target data matrices at each node i, which are stacked to form the global matrices X and Y, respectively. If a node's data lacks features, local statistics are padded with zeros to match the global dimension.

In this approach, matrices A and B are transmitted to the round leader, enabling efficient and privacy-preserving updates while avoiding full recomputations (see [44]).

4. Problem formulation

In this section, we present the problem formulation for our proposed FL system. We begin by defining the system model, followed by a description of how global statistics are constructed incrementally. We then discuss the properties of the update matrices used in our approach and conclude with a detailed explanation of the incremental update mechanism for the inverse of the regularized covariance matrix.

4.1. System model

Consider the FL scenario illustrated in Figure 1, where a group of K individual nodes participate in a training process that takes place over R rounds. At each aggregation round $r \in \{1, ..., R\}$, one of the nodes assumes the role of leader (Step 1) to coordinate the training process. Some nodes may not engage in some training round (Step 2) of the training process if they do not have sufficient local data or due to other constraints. Nodes that do participate send their local contributions to the leader (Step 3), who aggregates these contributions (Step 4), and then broadcasts the global model updates to all participating nodes (Step 5).

Each node $i \in \{1, ..., K\}$ collects n_i^r new samples at round r, forming a local dataset $D_i^r = \{(X_i^r, Y_i^r)\}$ with d features, where $X_i^r \in \mathbb{R}^{n_i^r\times d}$ and $Y_i^r \in \mathbb{R}^{n_i^r\times 1}$ are the input and target matrices, respectively. The local dataset D_i^r remains private and inaccessible to any other node $j \neq i$. The data collected over rounds, represented as batches $\{X_i^1, X_i^2, ..., X_i^r\}$ are independent and non-IID, reflecting the heterogeneity encountered in real-world scenarios.

Therefore, the goal is to estimate the global regression model $W \in \mathbb{R}^{d\times 1}$, whose closed-form solution is presented

in Equation (4). However, this solution is not computed directly from all data at once, but rather constructed incrementally in each communication round r using the terms A and B, which summarize the relevant information collected by each node, as expressed in Equations (5) and (6).

This strategy captures the joint information distributed across all nodes while preserving privacy by avoiding the exchange of raw data. Therefore, the global model in Equation (2) for a decentralized system can be computed analogously in a dynamic system by minimizing the following regularized empirical risk:

$$L(W) = \sum_{r=1}^{R} \sum_{i=1}^{K} \|Y_i^r - X_i^r W\|^2 + \lambda \|W\|^2 \tag{7}$$

where $\lambda > 0$ is a regularization parameter.

4.2. Incremental construction of global statistics

In our approach, the global covariance matrices A and B (see Section 3.3) are defined by aggregating local contributions at each update round r, instead of computing them in a single step:

$$A = \sum_{r=1}^{R} \sum_{i=1}^{K} A_i^r \tag{8} \qquad B = \sum_{r=1}^{R} \sum_{i=1}^{K} B_i^r \tag{9}$$

where $A_i^r = X_i^{r\top} X_i^r$ and $B_i^r = X_i^{r\top} Y_i^r$ compute the local statistics at each node i and round r, from its private data batch $D_i^r = (X_i^r, Y_i^r)$. Since it only relies on additive statistics, in addition to preserving privacy, the method remains applicable even under non-IID data distributions, as it does not assume any specific structure on how data is partitioned across nodes or rounds.

We denote A^r the global matrix accumulated at round r, and similarly B^r. Then, the full statistics satisfy:

$$A^r = \sum_{i=1}^{K} X_i^{r\top} X_i^r \tag{10} \qquad B^r = \sum_{i=1}^{K} X_i^{r\top} Y_i^r \tag{11}$$

While transmitting sufficient statistics ($X^T X$, $X^T Y$) avoids sharing raw data directly, we assume a regime where $n \gg d$ to prevent feature reconstruction. As already pointed out in Section 3.1, for stricter privacy guarantees, FedDelta is compatible with techniques like SMPC, DP or HE.

4.3. Properties of the update matrices

At each update round r, the global matrix A^r is updated with the local contributions A_i^r (see Equation (8)) from each node i (that participates in that round). This matrix is symmetric and positive semidefinite (PSD), since it is the product of a matrix and its transposed. Consequently, since A^r is the sum of PSD matrices, it is also PSD.

To ensure numerical stability and invertibility of the system matrix, we work with the regularized version

$$P^r = \sum_{k=1}^{r} A^k + \lambda^r I_d \tag{12}$$

where $\lambda^r > 0$ is a regularization parameter, which could be different for each round r. The matrix P^r aggregates all the global contributions A^k obtained up to the current round r. Since each A^k is PSD and $\lambda^r I_d$ is symmetric positive definite (SPD), their sum P^r is SPD as well.

Given the incremental nature of the system, the regularization is expressed as

$$P^r = P^{r-1} + A^r + \Delta\lambda^r I_d \tag{13}$$

where $r > 1$ and $\Delta\lambda^r = \lambda^r - \lambda^{r-1}$ captures any changes in the regularization parameter over time. Of course, if the regularization parameter is fixed, then $\Delta\lambda^r = 0$, since there is no change in that round. Accordingly, the update matrix at round r is defined as

$$U^r = A^r + \Delta\lambda^r I_d \tag{14}$$

Since I_d is symmetric, $\Delta\lambda^r I_d$ is also symmetric, and thus this regularization term ensures that P^r remains SPD, which guarantees numerical stability and invertibility of the matrix throughout the incremental updates.

4.4. Incremental update of the inverse

To enable incremental updates, we leverage the fact that the change from the regularized version P at a given round to P' at next round can be expressed as an update matrix U (see Equation (14)), defined as the aggregated node contribution A and the regularization adjustment term $\Delta\lambda I_d$. That is,

$$P' = P + U \tag{15}$$

Lemma 1 (Symmetric Spectral Decomposition)**.** *Given a matrix U computed in a given round using Equation (14), there exists an orthogonal matrix $Q \in \mathbb{R}^{d\times d}$ and a diagonal matrix $\Lambda \in \mathbb{R}^{d\times d}$ such that:*

$$U = Q\Lambda Q^\top \tag{16}$$

where $\Lambda = \mathrm{diag}(\sigma_1, \ldots, \sigma_d)$ whose entries σ_i are the eigenvalues of the matrix U.

Proof. The proof is directly derived from the spectral theorem, because U is symmetric, since it is obtained as a sum of symmetric matrices A and $\Delta\lambda I_d$, for each round. □

The low-rank update structure in Equation (16) enables efficient computation of the inverse of P' from the inverse of P by applying SMW identity to Equation (15):

$$P'^{-1} = P^{-1} - P^{-1} Q S^{-1} Q^\top P^{-1} \tag{17}$$

where $S = \Lambda^{-1} + Q^\top P^{-1} Q$, and Λ^{-1} corresponds to the diagonal matrix $\mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_d)$, where divisions by zero (if any $\sigma_i = 0$) are handled using the Moore-Penrose pseudoinverse. Since the update matrix U is $d \times d$, it is a full-size matrix, but its effective rank, denoted as $\mathrm{erank}(U)$, depends on the contributions by each node, which determine its low-rank structure. This allows for inverting only a $\mathrm{erank}(U) \times \mathrm{erank}(U)$ matrix, leading to significant computational savings when $\mathrm{erank}(U) \ll d$.

5. FedDelta: an incremental approach

This section presents the algorithmic details of FedDelta. First a specification of the algorithm itself, and then an analysis of its computational complexity.

5.1. Algorithm overview

FedDelta is structured around three main stages: initialization, local updates, and aggregation. These steps are carried out by two distinct roles in the system: the leader (Alg. 1), which handles model aggregation and broadcasting, and the participants (Alg. 2), which compute local statistics and update their models accordingly. In this architecture, all nodes perform the participant role, with one node additionally taking on the temporary leader role. This rotation ensures that while the computational exactness of centralized aggregation is preserved, the structural dependency on any single node is removed, thus making the approach more robust.

Roles and leader management

FedDelta uses a probabilistic Gossip protocol for leader election and model coordination [26]. The Gossip model implements an asynchronous push–pull model, with logarithmic convergence, resilience to node churn, and minimal per-node communication overhead.

At the beginning of each training round, a leader selection stage takes place in which each eligible node generates a bid value and participates in a fixed number of gossip rounds (G). During this process, nodes communicate with a subset of randomly selected peers (fanout), exchanging their known bid values. This epidemic-style information dissemination ensures rapid convergence of the election state across the network. After G rounds, the node with the highest bid value is promoted to leader status, with random selection used for tie-breaking among the most capable nodes.

Once leadership is established, the elected node sends a request to all participating nodes (Alg. 1, L5) to coordinate the FL process by aggregating their local model updates (L9–29). The system has a timeout for all nodes to communicate. Once that time expires, it must rely on a quorum-based update mechanism, where a subset of nodes S such that $|S| \geq m$, with m being the minimum number of nodes required to contribute their updates, has answered so that the leader can compute and distribute the global model. This threshold ensures the statistical validity of the aggregated model while providing flexibility in handling node availability. If no quorum is reached, the leader informs all nodes that the round is being skipped (Alg. 1, L32). In this way, nodes that participated in the attempt may save their data for future iterations.

When a node completes its local training (Alg. 2, L3–6), it transmits its local model update to the leader. Once the required quorum m is reached, the leader processes the accumulated updates to generate a global model update, which is then distributed back to all participating nodes (Alg. 1, L28, and Alg. 2, L11). This process is repeated for each training round, with new leader elections occurring at

Algorithm 1 LEADERROUTINE

1: $A_{\text{Global}} \leftarrow$ zero matrix of size $d \times d$
2: $B_{\text{Global}} \leftarrow$ zero matrix of size $d \times 1$
3: $r = 1$
4: **while** true **do**
5: Send *requests* for updates to all nodes
6: Wait for updates until *timeout* expires
7: Let S be the subset of nodes that responded
8: **if** $|S| \geq m$ **then**
9: Let $\{A_i, B_i\}_{i \in S}$ be the updates received
10: Compute reg. term λ and store the prev. one λ_{prev}
11: $A \leftarrow$ zero matrix of size $d \times d$
12: $B \leftarrow$ zero matrix of size $d \times 1$
13: **for** each node $i \leftarrow 1 \in S$ **do**
14: $A \leftarrow A + A_i$
15: $B \leftarrow B + B_i$
16: **end for**
17: $A_{\text{Global}} \leftarrow A_{\text{Global}} + A$
18: $B_{\text{Global}} \leftarrow B_{\text{Global}} + B$
19: **if** first round **then**
20: $P \leftarrow A_{\text{Global}} + \lambda I_d$
21: Compute and store P^{-1}
22: **else**
23: $\Delta\lambda \leftarrow \lambda - \lambda_{\text{prev}}$
24: $U \leftarrow A + \Delta\lambda I_d$
25: $Q, \Lambda \leftarrow$ SPECTRALDECOMPOSITION(U)
26: $P^{-1} \leftarrow \text{SMW}\left(P^{-1}, Q, \Lambda\right)$ ▷ *Store locally*
27: **end if**
28: Broadcast P^{-1} and B_{Global} to K participants
29: $W \leftarrow P^{-1} B_{\text{Global}}$
30: $r \leftarrow r + 1$
31: **else**
32: Notify all nodes in K that the round is being skipped
33: **end if**
34: **end while**

Algorithm 2 PARTICIPANTROUTINE

1: **while** true **do**
2: Wait for request from leader
3: **if** enough new data available **then**
4: Prepare a new data batch $(X_{\text{new}}, Y_{\text{new}})$
5: $A = X_{\text{new}}^{\top} X_{\text{new}}$
6: $B = X_{\text{new}}^{\top} Y_{\text{new}}$
7: Send A, B to leader
8: **end if**
9: Wait for notification from leader
10: **if** round not skipped (by leader) **then**
11: Get P^{-1} and B_{Global}
12: $W = P^{-1} B_{\text{Global}}$
13: **end if**
14: **end while**

the start of each round to maintain system dynamicity and fault tolerance.

Local computation and global aggregation

Model updates proceed in each training round. Nodes may send updates when new data becomes available. Nodes that skip a round, or have been unavailable for multiple rounds, still receive the current global parameters to synchronize, even if they did not contribute their local statistics in that round, as illustrated in Figure 1.

When so requested (Alg. 2, L2), and if it has at least the minimum number of samples, each node i computes its local statistics (A, B), as specified in Alg. 2, L5–6. These matrices are additive across data partitions and communication rounds, enabling the global quantities to be formed by simple accumulation (Alg. 1, L14–15).

As already explained in previous sections, the leader incorporates a ridge regularization term at each round (Alg. 1, L10). This regularization is captured by the matrix P^r (Equation (13)), which incorporates the update matrix U^r (Equation (14)). This matrix consists of two components: A^r, the aggregated data-driven update, and the term $\Delta\lambda^r$, which allows for dynamic regularization. The computation of the global model is specified in Alg. 1, L19–27. Once P^r is computed, the leader inverts it, and then derives the global model for round r by solving:

$$W^r = P^{r^{-1}} \sum_{k=1}^{r} B^k \tag{18}$$

where the coefficient vector W^r represents the global model at round r, and $\sum_{k=1}^{r} B^k$ denotes the global matrix accumulated up to the current round r.

For rounds other than the first one, the matrix from the previous round, denoted as P^{r-1}, has already been computed and stored by the leader. This allows the use of SMW to update $P^{(r-1)^{-1}}$ without recomputing the inverse $P^{r^{-1}}$ from scratch. To apply this identity, the matrix update U^r is expressed in low-rank form, as detailed in Equation (16). This recursive inverse update is specified in Alg. 1, L24–26. After $P^{r^{-1}}$ is computed, the leader derives the global model for round r by solving Equation (18).

Once computed, the leader distributes the accumulated global matrix B_{Global} and $P^{r^{-1}}$ to all available nodes (Alg. 1, L28). By distributing these matrices, not only all nodes have an updated model, they are all ready to become leaders in the next round.

Stopping criterion

The procedure is specified as an infinite loop, since it is intended for distributed systems which indefinitely get new data. However, an alternative stopping criteria could be established. E.g., we could fix the number of rounds or we could use a converge criterium, e.g., stopping when the mean squared error (MSE) on a fixed validation set stops improving, or gets below some threshold for some given number of rounds.

5.2. Complexity analysis

In this section, we formally define the iteration complexity required by the nodes during the learning process across multiple rounds of FL, given the algorithms specified in Section 4.1. To contextualize the performance of our approach, we compare FedDelta with a centralized learning system and the three baseline FL aggregation methods based on Mini-Batch Gradient Descent (MGD) for multivariable linear regression, namely FedNova, FedProx, and FedAvg. The results of this analysis are summarized in Table 1.

FedAvg, FedProx, and FedNova perform E local iterations over minibatches of size b_i for each node, with a local complexity $\mathcal{O}(Eb_id)$. Once the local update is completed, each of the K nodes sends its model $W_i \in \mathbb{R}^{d\times 1}$ to the leader node, which aggregates the K local models through a weighted average, with a computational cost of $\mathcal{O}(Kd)$. The leader then distributes the global model to each node, which will use it as the base for the next training round. This distribution has a communication cost of $\mathcal{O}(Kd)$.

Although FedAvg, FedProx, and FedNova share the same order of complexity for both local computation and global aggregation, they differ in how they perform the update and combine the models: FedAvg applies a weighted average of the local models based on the number of samples at each node; FedProx introduces a proximal term that stabilizes the updates in non-IID environments, slightly increasing the cost per local iteration without affecting the overall complexity; and FedNova normalizes the local updates by the effective number of steps, which introduces an additional overhead in the aggregation phase, although it still maintains a complexity of $\mathcal{O}(Kd)$. Thus, while these methods have differences in their internal structure, they all share the same asymptotic complexity per round.

According to the FedDelta algorithm, each node i, with n_i samples, computes its local matrices A_i and B_i, with operation costs of $\mathcal{O}(n_id^2)$ and $\mathcal{O}(n_id)$, respectively, that is, the final local cost per node is bounded by $\mathcal{O}(n_id^2)$. Subsequently, the leader node aggregates the local contributions for that round by summing K matrices, which has a complexity of $\mathcal{O}(Kd^2)$. Next, the regularization term (a direct sum) is added, and the inverse of the global matrix P^r is computed. For the initial round ($r = 1$), this operation is done from scratch with a cost of $\mathcal{O}(d^3)$, whereas for subsequent rounds ($r > 1$), the leader follows the incremental procedure described in Section 5.1, which involves symmetric spectral decomposition of updates with a cost of $\mathcal{O}(d^3)$, followed by an update of the inverse using the SMW formula with a cost of $\mathcal{O}(\text{erank}(U)\, d^2)$, where $\text{erank}(U)$ is the effective rank of the update matrix, and typically $\text{erank}(U) \ll d$.

The coefficient vector W is computed using Equation (18) by both the leader and the other nodes, with an additional cost of $\mathcal{O}(d^2)$. Finally, the leader sends the global matrices B_{Global} (which corresponds to the partial sum $\sum_{k=1}^{r} B^k$ in Equation (9)) and $P^{r^{-1}}$ (see Equation (17)) to all K nodes. The communication cost for this step is $\mathcal{O}(Kd^2)$, which is dominated by the transmission of the $d \times d$ matrix $P^{r^{-1}}$ (see Table 1).

Table 1
Summary of the computational and communication complexity per round.

Stage	FedDelta	FedNova	FedAvg	FedProx	Centralized
Local computation	$\mathcal{O}(n_i d^2 + \text{erank}(U)d^2 + d^3)$	$\mathcal{O}(Eb_i d)$	$\mathcal{O}(Eb_i d)$	$\mathcal{O}(Eb_i d)$	$\mathcal{O}(nd^2 + d^3)$
Global aggregation	$\mathcal{O}(Kd^2)$	$\mathcal{O}(Kd)$	$\mathcal{O}(Kd)$	$\mathcal{O}(Kd)$	$\mathcal{O}(nd^2)$
Comm. distribution	$\mathcal{O}(Kd^2)$	$\mathcal{O}(Kd)$	$\mathcal{O}(Kd)$	$\mathcal{O}(Kd)$	$\mathcal{O}(nd)$
Total per round	$\mathcal{O}\left(nd^2+Kd^2+\text{erank}(U)d^2+d^3\right)$	$\mathcal{O}\left(E(b+K)d\right)$	$\mathcal{O}\left(E(b+K)d\right)$	$\mathcal{O}\left(E(b+K)d\right)$	$\mathcal{O}(nd+nd^2+d^3)$

On the other hand, the centralized approach concentrates all information and computation in a single node, with a total complexity of $\mathcal{O}(nd^2 + d^3)$, derived from the main matrix operations: the computation of $X^\top X$ with a cost of $\mathcal{O}(nd^2)$, $X^\top Y$ with $\mathcal{O}(nd)$, and the inversion of $X^\top X$ with $\mathcal{O}(d^3)$. This can lead to a bottleneck in scenarios with large volumes of data. In this context, FedDelta provides an efficient balance by delegating the heavy computation to the aggregation step and leveraging incremental updates to minimize costs without sacrificing accuracy.

As highlighted in Table 1, methods based on MGD exhibit linear local complexity, making them suitable for devices with limited computational resources. However, their performance depends on several factors, such as the total number of local iterations E, communication frequency, and the convergence behavior of their iterative updates. In contrast, FedDelta shifts more computational load to each client by calculating exact accumulated statistics, but enables a deterministic and precise update of the global model per round, with more efficient initial inversion and subsequent updates. This structure ensures exact equivalence with the centralized model, distributes the computational load more evenly among nodes, reduces the overall cost by leveraging the incremental update framework (provided that the updates are of low rank), and avoids exposing raw data by transmitting only sufficient statistics.

6. Experiments and results

In this section, we present the experimental evaluation of our proposed FedDelta algorithm against three state-of-the-art federated learning methods (FedAvg, FedNova, and FedProx) and a centralized baseline. First, the used datasets, experimental settings and scenarios are presented, and then the results are discussed. The source code of the algorithm, along with the datasets and the experimental evaluation—including additional experiments and results omitted here for space reasons—is available at https://github.com/scenic-uma/feddelta.

6.1. Data sets

The experiments discussed in the following sections utilize four publicly available datasets, each exhibiting distinct characteristics in terms of size, dimensionality, and multicollinearity. Table 2 summarizes their main statistics, including number of samples (n), number of features (d), and key multicollinearity metrics, namely absolute Pearson correlation ($\overline{|\rho|}$), mean variance inflation factor (VIF), and total correlation:

Table 2
Summary of datasets.

Dataset	n	d	$\overline{\lvert\rho\rvert}$	VIF	Tot. Corr.
discharges	667,332	1,342	0.15	15.20	19.20
discharges-fs	2,346,760	14	0.68	2.32	1.17
insurance	1,338	6	0.04	4.81	10.28
housing	20,433	12	0.30	35.46	20.03
covid-19	505,211	60	0.10	28.07	43.34

- The *covid-19* dataset[1] includes data on confirmed COVID19 cases and deaths, hospitalizations and intensive care unit admissions, as well as information on geographical and temporal coverage. It also provides detailed country-by-country source data, vaccination information, and other related variables.
- Both *discharges* and *discharges-fs* are derived from the Statewide Planning and Research Cooperative System (SPARCS) Inpatient De-identified dataset[2], which provides detailed discharge information on patient characteristics, diagnoses, treatments, services, charges, and basic record-level details for each discharge. The original dataset contains 34 features and 2,346,760 samples:
  - *discharges-fs* is generated by applying a feature selection process, showing a very high average absolute Pearson correlation but uniformly low VIF values (mean and max VIF $\approx$ 2.32), yielding a moderate condition number ($\kappa \approx 3.0\times10^7$) and low total correlation.
  - *discharges* is the result of applying label encoding [45]. Samples with missing values for some attributes were removed, significantly reducing the dataset size.
- The *insurance* (Medical Insurance Cost) dataset[3] includes age, gender, BMI (Body Mass Index), number of children or dependents, whether the beneficiary is a smoker, their region of residence in the US, and the medical costs billed by the insurance provider.

[1] The *covid-19* dataset is available at https://docs.owid.io/projects/etl/api/covid/#legacy-data-work.

[2] The *discharges* dataset is available at https://health.data.ny.gov/Health/Hospital-Inpatient-Discharges-SPARCS-De-Identified/82xm-y6g8/about_data.

[3] The *insurance* dataset is available at https://www.kaggle.com/datasets/mirichoi0218/insurance/data.

- The *housing* (California Housing Prices) dataset[4] contains information about houses in a California district and summary statistics based on 1990 census data.

6.2. Experimental setup

Experiments were run on a workstation with an Intel Core i5-10300H CPU, 16 GB RAM, and an NVIDIA GeForce GTX 1650 GPU, under Microsoft Windows 11 Pro (build 26100). We used Python 3.10 with scikit-learn 1.0 as ML backend, along with numpy $\geq$ 1.21, pandas $\geq$ 1.3, and matplotlib $\geq$ 3.4. We set `n_jobs=-1` to leverage all available logical processors.

All experiments were conducted with $K = 20$ nodes. As detailed in the following section, each scenario employs a distinct data distribution across these nodes to simulate various heterogeneity and imbalance patterns. We set the number of global training rounds to 30, which is sufficient for FedAvg, FedNova, and FedProx to reach convergence, although scenarios with larger sample sizes often stabilize in fewer rounds. At the beginning of each round, a leader is dynamically elected using a lightweight gossip protocol: each node submits a bid based on its computational resources, and three push-pull gossip iterations (fan-out of 3) are performed to disseminate and compare bids. The node with the highest bid is selected as the leader for that round, ensuring both fault tolerance and optimal use of available resources. A fixed regularization parameter $\lambda = 0.01$ is employed for all algorithms. To ensure fair runtime comparisons, all MGD-based methods are trained for 200 local epochs per client with a batch size of 32, thereby standardizing the computational workload across algorithms.

6.3. Experimental scenarios

The experimental scenarios are designed to systematically assess the influence of four key factors on both convergence (measured by R^2) and execution time for all evaluated methods. Specifically, we address the following research questions:

Q1 How does the total sample size $n_{\text{tot}} = \sum_k n_k$ affect convergence and runtime?

Q2 For a fixed n_{tot}, how does the distribution of samples across clients ($\{n_k\}$)—i.e., client-level imbalance—impact performance?

Q3 What is the effect of intrinsic feature heterogeneity (multicollinearity) on each method?

Q4 How does the number of features d influence convergence and computational cost?

To answer these questions, we evaluate FedDelta, Centralized, FedAvg, FedNova, and FedProx in four different scenarios. Each of the scenarios is described by a tuple $(N_s : N_l)$, where on each round N_s is the number of samples assigned to the majority of clients (90% of clients, 18 in these experiments, since K is 20); and N_l is the number of samples assigned to a minority of clients (10% of clients, 2 in these experiments). The four scenarios on which the behavior of the different algorithms is compared are then described as:

- Balanced small (20 : 20): all clients have a small, similar number of samples.
- Balanced large (4000 : 4000): all clients have a large, similar number of samples.
- Moderate imbalance (20 : 3000): most clients have few samples, a minority have many.
- Extreme imbalance (30 : 10000): strong skew between majority and minority clients.

These scenarios address Q1 (by contrasting "small" vs. "large") and Q2 (by contrasting balanced vs. skewed at fixed n_{tot}). By correlating the heterogeneity metrics shown in Table 2 by the different datasets with R^2 and execution time, we directly quantify each method's sensitivity to intrinsic multicollinearity (Q3). We analyze how convergence and runtime scale with total sample count and feature dimensionality (addressing Questions Q1 and Q4) by comparing FedDelta's invariant cost to the linear scaling of gradient methods (FedAvg, FedNova, FedProx) and the cubic growth of the Centralized method.

Table 3
Welch's t-test for Centralized and FedDelta.

Dataset	$\overline{R^2}_C$	$\overline{R^2}_\Delta$	ΔR^2	p_{R^2}	T_C/T_Δ	p_T
discharges-fs	0.72	0.72	-2.30×10^{-4}	0.58	85.2×	0.19
discharges	0.86	0.86	-3.13×10^{-3}	0.06	55.0×	0.04
covid-19	0.97	0.97	< 0.001	1.00	47.8×	0.01
housing	0.95	0.95	< 0.001	0.77	2.07×	0.12
insurance	0.76	0.76	< 0.001	0.94	0.67×	0.06

6.4. Overall comparison

Table 3 reports a per-dataset, pairwise comparison between Centralized and FedDelta (Welch's t-tests on R^2 and runtime). FedDelta attains virtually identical accuracy to the centralized solver on all datasets ($|\Delta R^2| \leq 3 \times 10^{-3}$; all $p > 0.05$) while delivering large speedups where scale is high (T_C/T_Δ).

Fig. 2 complements this by situating FedDelta among all federated baselines across the four partition patterns. Each figure displays boxplots of R^2 and execution time (in seconds) for the four representative data partitions for the *covid-19* dataset, highlighting the comparative performance of FedDelta, Centralized, FedAvg, FedNova, and FedProx. Results for the other datasets can be found in the supplementary material available at https://scenic.uma.es/feddelta/. Across partitions, FedDelta tracks Centralized in R^2 while requiring substantially less wall-clock time; in contrast, FedAvg/FedNova/FedProx plateau at much lower R^2 values and converge more slowly. Centralized

[4] The *housing* dataset is available at https://www.kaggle.com/datasets/camnugent/california-housing-prices/data.

Table 4
Mean and standard deviation of the R^2 score across datasets using the Friedman base comparison.

Dataset	FedDelta	FedAvg	FedNova	FedProx	Cent.	FT
covid-19	$0.98_{<0.01}$	$0.94_{<0.01}$	$0.94_{<0.01}$	$0.94_{<0.01}$	$0.98_{<0.01}$	+
discharges-fs	$0.86_{<0.01}$	$0.22_{<0.01}$	$0.22_{<0.01}$	$0.22_{<0.01}$	$0.86_{<0.01}$	+
housing	$0.95_{<0.01}$	$-0.57_{0.04}$	$-0.57_{0.04}$	$-0.58_{0.04}$	$0.95_{<0.01}$	+
insurance	$0.76_{<0.01}$	$0.17_{0.01}$	$0.17_{0.01}$	$0.17_{0.01}$	$0.76_{<0.01}$	+

Table 5
Mean and standard deviation of execution time (in seconds) across datasets using the Friedman base comparison.

Dataset	FedDelta	FedAvg	FedNova	FedProx	Cent.	FT
covid-19	$1.06_{0.04}$	$94.69_{1.34}$	$94.08_{1.33}$	$103.09_{1.52}$	$40.32_{0.86}$	+
discharges-fs	$1.03_{0.02}$	$94.62_{1.14}$	$94.14_{1.19}$	$103.00_{1.31}$	$40.75_{0.93}$	+
housing	$0.61_{0.03}$	$88.58_{2.86}$	$87.86_{2.82}$	$96.47_{3.12}$	$1.83_{0.13}$	+
insurance	$1.01_{0.22}$	$83.96_{9.73}$	$70.73_{7.78}$	$77.36_{8.48}$	$0.22_{0.23}$	+

remains faster than gradient methods in some scenarios (e.g., *covid-19* under (20:20) and (20:3000)), but FedDelta consistently achieves the largest efficiency gains. Taken together, these results show that FedDelta preserves centralized accuracy—FedDelta consistently matches the centralized R^2—with markedly better efficiency under both balanced and skewed client distributions. Moreover, FedDelta consistently reduces runtime by one to two orders of magnitude, and outperforms gradient-based methods in both accuracy and speed across all heterogeneity patterns.

6.5. Statistical analysis

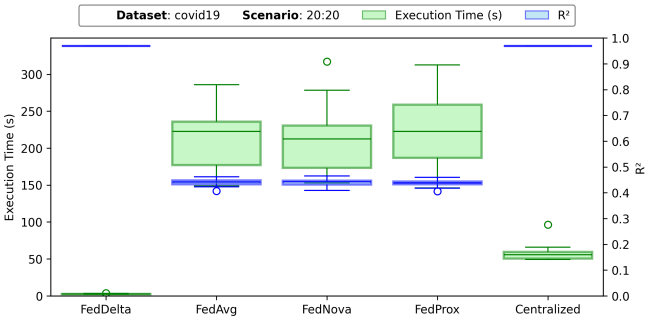
We conducted statistical analysis using SAES [46] following a three-step pipeline: (1) normality assessment using Shapiro-Wilk test to determine whether to use parametric (ANOVA and t-test) or non-parametric tests, (2) since data violated normality assumptions, we applied Friedman's test (FT) for omnibus comparisons, and (3) Wilcoxon signed-rank tests for pairwise comparisons when significance was detected ($\alpha < 0.05$).

Tables 4 and 5 show, respectively, the FT results for R^2 and execution time scores. Dark and light gray shades represent the best and second-best values, respectively. A '+' in the FT column indicates a significant difference for the raw dataset. Table 4 shows that FedDelta consistently achieves performance comparable to the Centralized approach and significantly outperforms all other considered FL methods across all datasets. Table 5 demonstrates that FedDelta achieves exceptional computational efficiency, with execution times comparable to Centralized while substantially outperforming other federated methods by orders of magnitude.

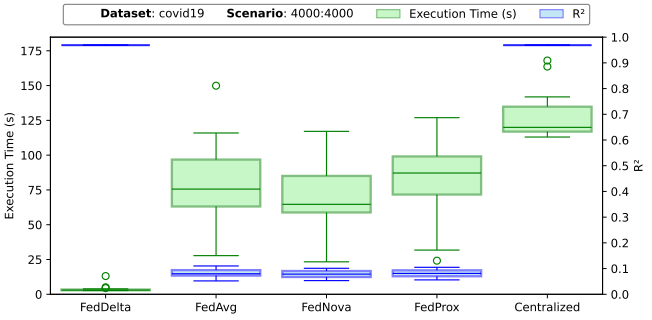
Each subtable in Table 6 presents the results of the Wilcoxon test, with each symbol summarizes the statistical relationship between the row and column algorithms across the *covid-19* and *discharges-fs* datasets, which were selected for pairwise comparison due to their distinctive behavior in the FT. A '++' symbol indicates that the row

Figure 2: Exec. time and R^2 comparison for *covid-19*.

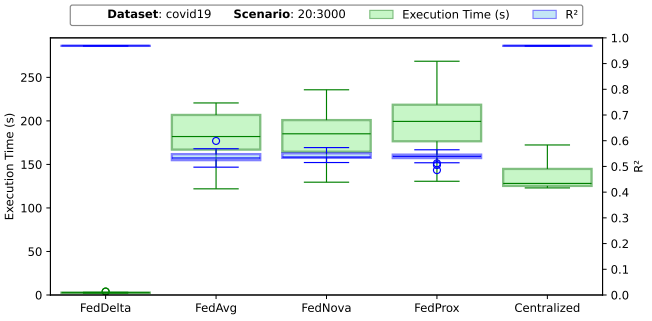
(a) Scenario (20:20).



(b) Scenario (4000:4000).



(c) Scenario (20:3000).



(d) Scenario (30:10000).

algorithm significantly outperforms the column algorithm in both datasets, while a single '+' indicates superiority in one of the two. Similarly, '–' means the column algorithm significantly outperforms the row algorithm in both datasets, '==' indicates no significant difference, and '=+' suggests a non-significant trend favoring the row algorithm. These results

Table 6
Wilcoxon test results. Each subtable reports pairwise comparisons between FedDelta and baseline methods for execution time and R^2 under different data partitions. Subtables are labeled (a)–(h).

(a) Exec. time for (20:20) data partition.

	FedAvg	FedNova	FedProx	Cent
FedDelta	++	++	++	++
FedAvg		==	=+	−−
FedNova			=+	−−
FedProx				−−

(b) R^2 for (20:20) data partition.

	FedAvg	FedNova	FedProx	Cent
FedDelta	++	++	++	−−
FedAvg		==	= +	−−
FedNova			= +	−−
FedProx				−−

(c) Exec. time for (4000:4000) data partition.

	FedAvg	FedNova	FedProx	Cent
FedDelta	++	++	++	++
FedAvg		==	==	++
FedNova			= +	++
FedProx				++

(d) R^2 for (4000:4000) data partition.

	FedAvg	FedNova	FedProx	Cent
FedDelta	++	++	++	−−
FedAvg		==	==	−−
FedNova			==	−−
FedProx				−−

(e) Exec. time for (20:3000) data partition.

	FedAvg	FedNova	FedProx	Cent
FedDelta	++	++	++	++
FedAvg		==	++	+−
FedNova			++	+−
FedProx				+−

(f) R^2 for (20:3000) data partition.

	FedAvg	FedNova	FedProx	Cent
FedDelta	++	++	++	−−
FedAvg		==	==	−−
FedNova			==	−−
FedProx				−−

(g) Exec. time for (30:10000) data partition.

	FedAvg	FedNova	FedProx	Cent
FedDelta	++	++	++	++
FedAvg		==	= +	+−
FedNova			= +	+−
FedProx				+−

(h) R^2 for (30:10000) data partition.

	FedAvg	FedNova	FedProx	Cent
FedDelta	++	++	++	−−
FedAvg		==	==	−−
FedNova			==	−−
FedProx				−−

show that FedDelta significantly outperforms all federated baselines, with competitive performance against Centralized (superior in three datasets, inferior in two, equivalent in one).

6.6. Convergence analysis

FedDelta's closed-form update via the SMW identity achieves the centralized ridge optimum in a single aggregation step, with no dependency on learning-rate schedules or local epochs. In contrast, gradient-based methods (FedAvg, FedNova, FedProx) require many local passes and multiple communication rounds yet systematically fall short of the centralized solution:

- On *discharges* and *discharges-fs*, FedAvg, FedNova and FedProx plateau at $R^2 \approx 0.04$ despite 1,000 local epochs, whereas FedDelta and Centralized both reach $R^2 \approx 0.72$-0.86 (Welch's t, $p > 0.05$ for ΔR^2).
- On *covid-19*, gradient-based methods stall below $R^2 \approx 0.36$; FedDelta matches Centralized at $R^2 \approx 0.97$ (no significant difference, $p \approx 1.0$).
- On *housing* and *insurance*, the same pattern holds: FedDelta converges exactly to the centralized R^2, while gradient solvers remain far below the optimum.

The absence of drift, oscillation or sensitivity to data imbalance or feature heterogeneity confirms FedDelta's robustness and exactness across all datasets.

6.7. Execution time analysis

In wall-clock time, FedDelta outperforms every alternative by one to two orders of magnitude:

- *discharges-fs*: Centralized 550s vs. FedDelta 5s ($T_C/T_\Delta \approx 110\times$, $p \ll 0.01$); FedAvg 90s, FedNova 100s, FedProx 105s.
- *discharges*: Centralized 300s vs. FedDelta 5s ($\approx 60\times$); gradients 80–95s.
- *covid-19*: Centralized 450s vs. FedDelta 6s ($\approx 75\times$); gradients 200–300s.
- *housing* and *insurance*: FedDelta completes in $\mathcal{O}(d^2)$ seconds ($\approx$ 2-6s), Centralized in $\mathcal{O}(d^3)$ ($\approx$ 7-50s), and gradient methods in $\mathcal{O}(n_{\text{tot}} \times \#\text{rounds})$ ($\approx$ 20-30s).

The dramatic speedup derives from FedDelta's constant-round, statistic-based aggregation cost versus the cubic inversion cost of Centralized and the linear-iterative cost of gradient schemes.

It is concluded that across all five real-world datasets—regardless of scale, imbalance or feature conditioning—FedDelta matches the exact convergence of the centralized

solver (no significant ΔR^2) and achieves 10-100× reductions in runtime, demonstrating unequivocal superiority over federated gradient methods and the centralized baseline.

7. Discussion

The experimental results demonstrate that our proposed approach achieves competitive performance compared to baseline methods. The improvements observed in accuracy and convergence speed suggest that our method effectively leverages the available data and model architecture.

Several factors may affect the validity of our findings:

- The implementation of baseline methods and hyperparameter tuning may introduce bias. We mitigated this by using publicly available code and performing grid search for key parameters.
- Even though the datasets used were carefully selected, trying to include datasets with a variety of characteristics, the experiments were conducted on a limited set of datasets and scenarios. The generalizability of our results to other domains or larger-scale settings remains to be validated.
- The choice of evaluation metrics may not capture all aspects of model performance, such as fairness or robustness.

The experimentation is quite illustrative, showing FedDelta's good behavior across diverse datasets and scenarios. The major gains can be seen in the *covid-19* and *discharges-fs* datasets. The *covid-19* dataset presents a "poorly conditioned" case, which stresses the numerical robustness and efficiency of FedDelta's SMW updates versus direct matrix inversion and gradient-based federated methods. On the other hand, the *discharges-fs* is a "well-conditioned yet correlated" scenario that verifies that FedDelta matches the centralized ridge solver under stable inversion.

8. Conclusion

In this paper, we have presented FedDelta, a novel federated learning approach specifically designed for multivariable linear regression. Our method leverages the closed-form solution of ridge regression and reformulates it using sufficient statistics, enabling efficient, privacy-preserving model updates without requiring raw data sharing. By employing the SMW formula, we achieve efficient incremental updates of the model parameters, significantly reducing computational overhead. Extensive experiments across various datasets demonstrate that FedDelta outperforms traditional gradient-based FL methods in both accuracy and time-to-solution speed, particularly in heterogeneous settings. Our results highlight the potential of FedDelta to address the challenges of FL in decentralized environments, making it a promising solution for real-world applications where data privacy and efficiency are paramount. Future work will explore extending FedDelta to other types of models and its application in more complex federated learning scenarios.

CRediT authorship contribution statement

Rafael García-Luque: Writing – original draft, Writing – review & editing, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization; **Ernesto Pimentel**: Supervision, Resources, Writing - Original Draft and Writing - Review & Editing; **Francisco Durán**: Supervision, Resources, Writing - Original Draft and Writing - Review & Editing; **Ivan Iroslavov**: Writing - Original Draft, Software, Methodology, Validation; and, **Emilio R. Carreira**: Writing - Original Draft, Software, Methodology, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been partly supported by project *Including people in smart city applications* (PID2021-125527NB-I00), funded by the *Spanish Ministry of Science and Innovation*.

Data availability

The datasets are available in their own public repositories. The implementation of the algorithm and all other resources, including links to the datasets used and some experimental evaluation is available at https://github.com/scenic-uma/feddelta.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. Agüera y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Procs. of the 20th Intl. Conf. on Artificial Intelligence and Statistics, AISTATS 2017, volume 54, 2017, pp. 1273–1282.

[2] A. Z. Tan, H. Yu, L. Cui, Q. Yang, Towards personalized federated learning, IEEE Transactions on Neural Networks and Learning Systems 34 (2023) 9587–9603. doi:10.1109/TNNLS.2022.3160699.

[3] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, A. T. Suresh, Scaffold: stochastic controlled averaging for federated learning, in: Procs. of the 37th Intl. Conf. on Machine Learning, ICML'20, 2020, p. 12.

[4] L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Review 60 (2018) 223–311. doi:10.1137/16M1080173.

[5] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, in: I. Dhillon, D. Papailiopoulos, V. Sze (Eds.), Proceedings of Machine Learning and Systems, volume 2, 2020, pp. 429–450. URL: https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf.

[6] J. Wang, Q. Liu, H. Liang, G. Joshi, H. V. Poor, Tackling the objective inconsistency problem in heterogeneous federated optimization, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, 2020, p. 13.

[7] K. Pillutla, S. M. Kakade, Z. Harchaoui, Robust aggregation for federated learning, IEEE Transactions on Signal Processing 70 (2022) 1142–1154. doi:10.1109/TSP.2022.3153135.

[8] O. Shahid, S. Pouriyeh, R. M. Parizi, Q. Z. Sheng, G. Srivastava, L. Zhao, Communication efficiency in federated learning: Achievements and challenges, CoRR abs/2107.10996 (2021). URL: https://arxiv.org/abs/2107.10996.

[9] Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-iid data silos: An experimental study, CoRR abs/2102.02079 (2021). URL: https://arxiv.org/abs/2102.02079. arXiv:2102.02079.

[10] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, H. Vincent Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Transactions on Information Forensics and Security 15 (2020) 3454–3469. doi:10.1109/TIFS.2020.2988575.

[11] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawit, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. Theertha Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, S. Zhao, Advances and Open Problems in Federated Learning, Now Foundations and Trends, 2021.

[12] T.-M. H. Hsu, H. Qi, M. Brown, Federated visual classification with real-world data distribution, in: 16th European Conf. on Computer Vision, ECCV 2020, Springer, 2020, pp. 76–92. doi:10.1007/978-3-030-58607-2_5.

[13] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of fedavg on non-iid data, in: 8th Intl. Conf. on Learning Representations, ICLR 2020, 2020, pp. 1–26.

[14] S. J. Reddi, A. Hefny, S. Sra, B. Póczós, A. Smola, Stochastic variance reduction for nonconvex optimization, in: Procs. of the 33rd Intl. Conf. on Intl. Conf. on Machine Learning - Volume 48, ICML'16, 2016, pp. 314–323.

[15] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, V. Saligrama, Federated learning based on dynamic regularization, CoRR abs/2111.04263 (2021). URL: https://arxiv.org/abs/2111.04263.

[16] J. Wang, V. Tantia, N. Ballas, M. G. Rabbat, Slowmo: Improving communication-efficient distributed sgd with slow momentum, ArXiv abs/1910.00643 (2019). URL: https://api.semanticscholar.org/CorpusID:203626531.

[17] T. H. Hsu, H. Qi, M. Brown, Measuring the effects of non-identical data distribution for federated visual classification, CoRR abs/1909.06335 (2019). URL: http://arxiv.org/abs/1909.06335.

[18] G. Kim, J. Kim, B. Han, Communication-efficient federated learning with acceleration of global momentum, CoRR abs/2201.03172 (2022). URL: https://arxiv.org/abs/2201.03172. arXiv:2201.03172.

[19] E. Fanì, R. Camoriano, B. Caputo, M. Ciccone, Accelerating heterogeneous federated learning with closed-form classifiers (Fed3R), in: 41st Intl. Conf. on Machine Learning (ICML), volume 235, 2024, pp. 13029–13048.

[20] F. Varno, M. Saghayi, L. Rafiee Sevyeri, S. Gupta, S. Matwin, M. Havaei, AdaBest: Minimizing client drift in federated learning via adaptive bias estimation, in: European Conf. on Computer Vision, ECCV 2022, Springer, 2022, pp. 710–726.

[21] D. M. Jimenez-Gutierrez, M. Hassanzadeh, A. Anagnostopoulos, I. Chatzigiannakis, A. Vitaletti, A thorough assessment of the non-iid data impact in federated learning, 2025. URL: https://arxiv.org/abs/2503.17070. arXiv:2503.17070.

[22] Y. R. Chen, A. Rezapour, W.-G. Tzeng, Privacy-preserving ridge regression on distributed data, Information Sciences 451-452 (2018) 34–49.

[23] C. Heinze, B. McWilliams, N. Meinshausen, G. Krummenacher, LOCO: Distributing ridge regression with random projections, 2015. URL: https://arxiv.org/abs/1406.3469. arXiv:1406.3469.

[24] M. Ye, X. Fang, B. Du, P. C. Yuen, D. Tao, Heterogeneous federated learning: State-of-the-art and research challenges, ACM Comput. Surv. 56 (2023). doi:10.1145/3625558.

[25] Z. Li, J. Lu, S. Luo, D. Zhu, Y. Shao, Y. Li, Z. Zhang, Y. Wang, C. Wu, Towards effective clustered federated learning: A peer-to-peer framework with adaptive neighbor matching, IEEE Transactions on Big Data 10 (2024) 812–826. doi:10.1109/TBDATA.2022.3222971.

[26] M. Blot, D. Picard, N. Thome, M. Cord, Distributed optimization for deep learning with gossip exchange, Neurocomputing 330 (2019) 287–296. doi:https://doi.org/10.1016/j.neucom.2018.11.002.

[27] A. Lalitha, O. C. Kilinc, T. Javidi, F. Koushanfar, Peer-to-peer federated learning on graphs, CoRR abs/1901.11173 (2019). URL: http://arxiv.org/abs/1901.11173.

[28] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, C. Wachinger, Braintorrent: A peer-to-peer environment for decentralized federated learning, CoRR abs/1905.06731 (2019). URL: http://arxiv.org/abs/1905.06731.

[29] Y. Liu, Y. Shi, B. Wu, Q. Li, X. Wang, L. Shen, Decentralized Directed Collaboration for Personalized Federated Learning , in: 2024 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2024, pp. 23168–23178. doi:10.1109/CVPR52733.2024.02186.

[30] G. Syros, G. Yar, S. Boboila, C. Nita-Rotaru, A. Oprea, Backdoor attacks in peer-to-peer federated learning, ACM Trans. Priv. Secur. 28 (2024). doi:10.1145/3691633.

[31] G. Lu, Z. Xiong, R. Li, N. Mohammad, Y. Li, W. Li, Defeat: A decentralized federated learning against gradient attacks, High-Confidence Computing 3 (2023) 100128. doi:https://doi.org/10.1016/j.hcc.2023.100128.

[32] S. Warnat-Herresthal, H. Schultze, K. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, S. Ktena, F. Tran, M. Bitzer, S. Ossowski, N. Casadei, C. Herr, D. Petersheim, U. Behrends, F. Kern, T. Velavan, Swarm learning for decentralized and confidential clinical machine learning, Nature 594 (2021) 265 – 270. doi:10.1038/s41586-021-03583-3.

[33] M. R. Behera, S. Upadhyay, S. Shetty, R. den Otter, Federated learning using peer-to-peer network for decentralized orchestration of model weights, 2021. URL: https://api.semanticscholar.org/CorpusID:235579336.

[34] Y. Lindell, Secure multiparty computation, Commun. ACM 64 (2020) 86–96. doi:10.1145/3387108.

[35] C. Dwork, Differential privacy: A survey of results, in: M. Agrawal, D. Du, Z. Duan, A. Li (Eds.), Theory and Applications of Models of Computation, Springer, 2008, pp. 1–19.

[36] C. Gentry, Fully homomorphic encryption using ideal lattices, in: 41st Annual ACM Symposium on Theory of Computing, STOC'09, ACM, 2009, pp. 169–178. doi:10.1145/1536414.1536440.

[37] B. Liu, N. Lv, Y. Guo, Y. Li, Recent advances on federated learning: A systematic survey, Neurocomputing 597 (2024) 128019. doi:https://doi.org/10.1016/j.neucom.2024.128019.

[38] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer, 2006.

[39] N. Draper, H. Smith, Applied Regression Analysis, 2nd edition, John Wiley & Sons, 1981.

[40] Y. Nesterov, Lectures on Convex Optimization, 2nd ed., Springer, 2018.

[41] H. V. Poor, Linear estimation (t.kailauth, a. h. sayed, and b. hassibi; 2000) [book review], IEEE Transactions on Information Theory 51 (2005) 2236–2240. doi:10.1109/TIT.2005.848662.

[42] W. W. Hager, Updating the inverse of a matrix, SIAM Review 31 (1989) 221–239. doi:10.1137/1031049.

[43] E. Fanì, R. Camoriano, B. Caputo, M. Ciccone, Resource-efficient personalization in federated learning with closed-form classifiers, IEEE Access 13 (2025) 61928–61957. doi:10.1109/ACCESS.2025.3556587.

[44] A. Björck, Numerical Methods for Least Squares Problems, Society for Industrial and Applied Mathematics, 1996. doi:10.1137/1.9781611971484.

[45] C. D. P. Kedar Potdar, Taher S. Pardawala, A comparative study of categorical variable encoding techniques for neural network classifiers, Intl. J. of Computer Applications 175 (2017) 7–9. doi:10.5120/ijca2017915495.

[46] E. R. Carreira, A. J. Nebro, SAES: A Python library for statistical evaluation of stochastic artificial ingelligence algorithms (1.3.6), Zenodo, 2025. doi:10.5281/zenodo.15683420.