Article

A Privacy-Enhanced Multi-Stage Dimensionality Reduction Vertical Federated Clustering Framework

Jun Wang * and XiangHua Chen

Posted Date: 9 July 2025

doi: 10.20944/preprints202507.0835.v1

Keywords: clustering; privacy protection; k-means; vertical federated learning; dimensionality reduction; PQ; MDS

Article

A Privacy-Enhanced Multi-Stage Dimensionality Reduction Vertical Federated Clustering Framework

Jun Wang * and XiangHua Chen

Nanjing University of Aeronautics and Astronautics; chenxianghua@nuaa.edu.cn
* Correspondence: wangjun0322@nuaa.edu.cn; Tel.: +86-1891-021-0673

Abstract

Federated Clustering (FL clustering) aims to discover latent knowledge in multi-source distributed data through clustering algorithms while preserving data privacy. Federated learning is categorized into horizontal and vertical federated learning based on data partitioning scenarios. Horizontal Federated Learning is applicable to scenarios with overlapping feature spaces but different sample IDs across parties. Vertical federated learning facilitates cross-institutional feature complementarity, particularly suited for scenarios with highly overlapping sample IDs yet significantly divergent features.As a classic clustering algorithm, k-means has seen extensive improvements and applications in horizontal federated learning. However, its application in vertical federated learning remains insufficiently explored, with room for enhancement in privacy protection and communication efficiency. Simultaneously, client feature imbalance may lead to biased clustering results.To improve communication efficiency, this paper introduces Product Quantization (PQ) to compress high-dimensional data into low-dimensional codes by generating local codebooks. Leveraging the inherent k-means algorithm within PQ, local training preserves data structures while overcoming privacy risks associated with traditional PQ methods that require server-side data reconstruction (which may leak data distributions).To enhance privacy without compromising performance, Multidimensional Scaling (MDS) maps codebook cluster centers into distance-preserving indices. Only these indices are uploaded to the server, eliminating the need for data reconstruction. The server executes k-means on the indices to minimize intra-group similarity and maximize inter-group divergence. This scheme retains original codebooks locally for strict privacy protection.The nested application of PQ and MDS significantly reduces communication volume and frequency while effectively alleviating clustering bias caused by client feature dimension imbalance. Validation on the MNIST dataset confirms that the approach maintains k-means clustering performance while meeting federated learning requirements for privacy and efficiency.

Keywords: clustering; privacy protection; k-means; vertical federated learning; dimensionality reduction; PQ; MDS

1. Introduction

Federated Learning (FL) is a distributed machine learning approach whose core concept involves multiple clients (e.g., mobile devices or enterprises) collaboratively training a shared global model without sharing local data[1]. Federated Learning Clustering integrates federated learning and clustering algorithms to mine latent knowledge from multi-source distributed data while ensuring data privacy protection[2]. Federated learning faces multiple challenges including data heterogeneity[3,4], communication bottlenecks[1,5], and privacy protection[6].

Federated learning is categorized into horizontal federated learning (Horizontal Federated Learning,HFL)[7] and vertical federated learning (Vertical Federated Learning, VFL)[8] based on data distribution patterns. In horizontal federated learning, participating parties share identical features but possess distinct sample IDs[9]. In vertical federated learning scenarios, the data distribution pattern entails different clients owning distinct feature sets while sharing identical sample IDs. Horizontal federated learning finds widespread application in edge computing and IoT collaboration[10,11]. Vertical federated learning's value lies in cross-institutional feature complementarity, particularly suitable for scenarios with highly overlapping sample IDs yet significantly divergent features, such as in healthcare[12], finance[13], and other industries.

Current research on federated clustering algorithms predominantly focuses on the horizontal federated domain, with relatively limited exploration in vertical federated settings. Compared to horizontal federated learning, vertical federated learning relies more heavily on high-frequency encrypted interactions and high-dimensional feature transmission [14], imposing greater demands on communication. Simultaneously, vertical federated learning requires multi-party collaboration to compute loss functions and gradients. Due to features being dispersed across different participants, it necessitates reliance on Secure Multi-Party Computation (MPC), Homomorphic Encryption (HE), and other technologies to achieve joint training under privacy protection , all incurring substantial computational costs[15].In 2023, Zitao Li et al. proposed a differentially private k-means clustering algorithm based on the Flajolet-Martin (FM) technique[16]. By aggregating differentially private cluster centers and membership information of local data on an untrusted central server, they constructed a weighted grid as a summary of the global dataset, ultimately generating global centers by executing the k-means algorithm. Subsequently, in the paper[17] , they introduced Flajolet-Martin (FM) sketches to encode local data and estimate cross-party marginal distributions under differential privacy constraints, thereby constructing a global Markov Random Field (MRF) model to generate high-quality synthetic data.Federico[18] proposed a vertical federated k-means algorithm based on homomorphic encryption and differential privacy protection, demonstrating its superior clustering performance (e.g., k-means loss and clustering accuracy) over traditional privacy-preserving K-Means algorithms while maintaining the same privacy level.Li [19]et al. proposed a vertical federated density peaks clustering algorithm based on a hybrid encryption framework. Building upon the merged distance matrix, they introduced a more effective clustering method under nonlinear mapping, enhancing Density Peaks Clustering (DPC) performance while addressing privacy protection issues in Vertical Federated Learning (VFL).

This study addresses the privacy protection issues of K-Means clustering in vertical federated learning scenarios by proposing a federated clustering framework based on lightweight multi-stage dimensionality reduction. It primarily resolves three major challenges: privacy constraints, communication bottlenecks, and feature imbalance problems.

Our main contributions are:

1) We innovatively propose a multi-stage dimensionality reduction framework applicable to vertical federated learning clustering based on Product Quantization (PQ) and Multidimensional Scaling (MDS) techniques. Locally, feature compression and codebook generation effectively reduce data volume, while PQ-quantized parameters inherently provide noise injection effects to enhance privacy. Innovatively, we introduce one-dimensional MDS embedding to map clustering centers in the codebook into distance-preserving indices. This achieves zero raw codebook upload, abandons data reconstruction on the server side, and fundamentally eliminates the risk of data distribution leakage.
2) The multi-stage dimensionality reduction mechanism significantly reduces transmitted data volume and communication frequency while ensuring clustering accuracy, thereby improving communication efficiency.
3) The combination of PQ dimensionality reduction and MDS embedding algorithms mitigates clustering bias caused by feature imbalance in vertical federated learning.
4) Extensive experiments on the MNIST dataset validate that our algorithm satisfies federated learning privacy requirements while preserving clustering accuracy.

The structure of our paper is arranged as follows: Section II discusses existing clustering algorithms, challenges in VFL, and theoretical analyses of various dimensionality reduction techniques;

Section III details our algorithmic design; Section IV presents the experimental process and results analysis conducted to validate our algorithm; Finally, Section V concludes the paper.

2. Related Work

2.1. k-means clustering algorithm:

k-means is an unsupervised learning algorithm that aims to partition n data objects into k clusters (k < n), such that similarity within the same cluster is maximized while similarity between different clusters is minimized[20].

The core principle involves iteratively optimizing cluster centroid positions to minimize the Sum of Squared Errors (SSE) within clusters:

$$\min_{C} \sum_{k=1}^{K} \sum_{x \in C_k} \|x - \mu_k\|$$

where μ_k is the centroid of cluster Ck

Algorithm 1 k-means algorithm steps:

```
Input: Dataset $X = \{x_1, x_2, \dots, x_n\}$,number of clusters K,maximum iterations T.
Output: Partitioning result of K clusters.

    Setps:
1:  Randomly select k samples as initial cluster centroids $\mu_1^{(0)}, \mu_2^{(0)}, \dots, \mu_3^{(0)}$,
2:  Iterate until convergence:
    1)  Assign each sample xi to the nearest cluster centroid:
            $C_k^t = \{x_i : \|x_i - \mu_k^{(t)}\|^2 \leq \|x_i - \mu_j^{(t)}\|^2, \forall j\}$
    2)  Update cluster centroids as the mean of cluster members:
            $\mu_k^{(t)} = \frac{1}{| C_k^{(t)} |} \sum_{x \in C_k^{(t)}} x$
        Termination conditions:
        i.   Centroid changes below threshold,
        ii.  Maximum iterations reached.
```

k-means exhibits linear time complexity growth[21], making it suitable for large-scale data while being simple, intuitive, and highly interpretable. However, it has several limitations: sensitivity to initial values[22], requirement of preset k-value, susceptibility to noise, and restrictions on cluster shapes. Its core challenges (k-value selection, sensitivity to initial centroids) can be mitigated through methods like the elbow method, silhouette coefficient, and k-means++.

2.2. Vertical Federated Learning

Horizontal federated learning involves clients sharing the same feature space but having different sample spaces. Vertical federated learning, in contrast, deals with datasets sharing the same sample space but possessing different feature spaces.

The vertical federated learning process comprises two stages: encrypted entity alignment (Private Set Intersection, PSI) (A Privacy-preserving Data Alignment Framework for Vertical Federated Learning) [23]and model training.

During the encrypted entity alignment stage, participants hash user IDs (e.g., using SHA-256) or encrypt them with RSA. They then perform secure intersection: exchanging encrypted IDs and comparing them through blinding and re-encryption techniques to avoid exposing non-overlapping samples. Ultimately, participants only learn the intersection ID list, with raw feature data remaining undisclosed[23].

In the model training phase, different participants own distinct feature spaces, while labels may be held by one party. To ensure privacy, encryption techniques (e.g., asymmetric encryption or homomorphic encryption) are typically employed during communication to transmit model updates or gradients, preventing data leakage. The central server or coordinator aggregates encrypted model updates from data providers securely[24].

Communication costs in vertical federated learning often become bottlenecks for training efficiency, particularly due to network heterogeneity among parties and the large volume of encrypted data. Coordinating communications incurs additional overhead. It is critical to minimize communication overhead without compromising model performance. As participant data dimensions increase, transmitting high-dimensional feature vectors significantly escalates communication burdens, requiring compression mechanisms to reduce bandwidth usage [25].

Privacy risks in vertical federated learning primarily stem from data distribution characteristics and multi-party data interactions. Sharing identical sample IDs across participants enables attackers to reconstruct user profiles through correlation analysis. During training, transmitting gradients or intermediate results may leak raw data distributions — for example, malicious participants could infer user scale or distribution during PSI. Model training requires collaborative computation of loss functions and gradients across parties. With features dispersed among participants, Secure Multi-Party Computation (MPC) and Homomorphic Encryption (HE)[24] are essential for privacy-preserving joint training, incurring substantial computational costs for security.

Two common approaches address privacy-security challenges: reducing communication volume (data quantity and transmission frequency) or transforming data. Reducing transmission volume and frequency simultaneously enhances communication efficiency.

Numerous approaches reduce communication volume, such as compression techniques including quantization, sparsification, model pruning, knowledge distillation, and embedding compression [26], which decrease transmitted model parameters. Quantization compresses gradients by reducing numerical precision, while sparsification transmits only critical gradient components while discarding insignificant ones. Communication strategies may also be optimized—for example, clients performing multiple local updates before server communication, or randomly selecting subsets of clients per round. Alternatively, constructing virtual datasets for unified server-side training reduces communication frequency.

Data processing methods involve homomorphic encryption, secure multi-party computation, differential privacy, random masking, etc. Current solutions face limitations: fully homomorphic encryption offers theoretical security but suffers from slow speeds; partially homomorphic variants are more practical yet provide compromised protection [27]. Differential privacy sacrifices model accuracy through noise injection [28].

Vertical federated learning faces an additional challenge: when participants (clients) exhibit significant disparities in feature quantities (e.g., one party's feature dimension dA $\gg$dB), this substantially impacts model performance, training efficiency, and privacy security. The high-dimensional feature party carries greater information volume, potentially causing the model to over-rely on its features and drowning out the contributions of the low-dimensional party. In secure aggregation, the party with fewer features is vulnerable to statistical correlation attacks by the high-dimensional party. The computational/communication overhead of the high-dimensional party A far exceeds that of B, becoming a system bottleneck.Existing solutions to this problem include mapping features from different clients into a unified space and employing dual regularization (local consistency + complementary consistency) to reduce distribution discrepancies[29]. Other methods prioritize adjusting client weights based on feature importance; for instance, the FedFSA algorithm dynamically weights the loss function according to feature quality[30]. Constraining the feature norms of local model outputs can also prevent high-dimensional clients from dominating updates and enhance generalization capability[30].Embedding compression serves as another effective strategy, using PCA or sparse coding to reduce dimensions of high-dimensional embedding vectors (e.g., from 100-dimensional to 20-dimensional)[31]. Adopting asynchronous updating mechanisms can alleviate this issue by allowing low-dimensional clients to submit updates more frequently, reducing waiting time[32].

This research focuses on implementing the K-Means clustering algorithm within vertical federated learning. The core challenge lies in collaboratively computing the distances between sample points (to centroids) and determining cluster assignments for samples, as well as updating centroids, all while ensuring participant data remains local (privacy protection).We propose using the Product Quantization (PQ) process instead of local model training. This serves a dual purpose: firstly, it leverages PQ's inherent column-wise k-means clustering to replace the original global-dimension k-means clustering; secondly, it utilizes PQ to compress data dimensionality and transmitted parameters.The objective of clustering algorithms is to partition a dataset into groups (clusters) based on inherent similarities, achieving high intra-cluster similarity and low inter-cluster similarity. Given that clustering relies on distance features, as long as the index can accurately represent the distance relationships among the cluster centers within the codebook, the codebook itself can remain local. Only the index needs uploading to the server for subsequent clustering, thereby minimizing transmitted data volume.Consequently, we designed a PQ and MDS-based multi-stage dimensionality reduction framework specifically for vertical federated scenarios, achieving the dual objectives of privacy preservation and communication efficiency.Adopting a multi-stage dimensionality reduction strategy (as opposed to a single-step direct reduction) is crucial for another key reason: significant disparities in feature dimensions across clients. Direct global dimensionality reduction could disproportionately favor clients with higher feature dimensions, leading to inaccurate results. Our framework effectively balances the impact of features with varying dimensions and avoids such bias through secondary dimensionality reduction in subspace.

2.3. PQ Quantization Technique

PQ quantization technique (Product Quantization)[33]is a quantization technique used for compressing high-dimensional vector data and approximate nearest neighbor search. Its fundamental principle decomposes the high-dimensional vector space into the Cartesian product of multiple low-dimensional subspaces, then quantizes each subspace separately. In PQ quantization, the original vector is divided into multiple subvectors. Each subvector undergoes clustering via the K-means algorithm to generate a codebook, and the original vector is replaced by its nearest centroid, thereby achieving efficient compression.

Current applications of quantization techniques in federated learning primarily focus on communication efficiency optimization, storage optimization, computational acceleration, personalized training, and enhanced privacy protection. For instance[34] indicates that quantization can reduce communication costs by two orders of magnitude. Quantized compressed models require less device storage, accelerate computation, and suit edge deployment. Product Quantization (PQ) can mitigate data heterogeneity issues in federated learning (Optimized Product Quantization), such as clients generating personalized codebooks based on local data for server usage. PQ-quantized parameters inherently provide noise injection effects. For example, the medical federated learning framework FL-TTE adds Gaussian noise (ξ=1.0) to quantized hazard ratio parameters, increasing model bias by only 0.5% while meeting privacy requirements (Gradient Obfuscation Gives a False Sense of Security in Federated Learning)[35].

However, traditional PQ techniques primarily emphasize data compression, and a critical step involves uploading the codebook to the server for data reconstruction. Although reconstructed data exhibits reduced accuracy compared to raw data, it still risks exposing the original data distribution. This research focuses on implementing clustering algorithms in vertical federated learning—where clustering is a classification problem rather than regression. Consequently, the server requires no data reconstruction. Capitalizing on clustering's reliance on distance features, we innovatively re-encode the codebook indices to ensure they represent distance relationships among cluster centers within the codebook. This enables retaining the codebook locally while uploading only indices to the server for clustering, maximally preserving local data and reducing computational overhead on the server side.

The core process of Product Quantization (PQ):

1) **Space Decomposition:** Decompose a vector of dimension D into Msubspaces, each with dimension D/M(requiring Dmod M=0).
   Mathematical Expression: The original vector v$\in R^d$ s partitioned into M subvectors vi$\in R^{d/M}$,with each subvector quantized independently
2) **Subspace Quantization:** Perform k-means clustering on each subspace to generate a codebook containing k cluster centers.
3) **Encoding Representation:** The original vector is represented by a combination of indices corresponding to the nearest cluster centers of its subvectors. For example, a 128-dimensional vector can be compressed into eight 8-bit indices (requiring only 64 bits of storage, achieving a compression ratio of up to 97%).

The advantages of PQ quantization are: The codebook in PQ is a collection of multi-stage cluster centers, decomposing high-dimensional problems into low-dimensional subproblems through space partitioning; by utilizing Cartesian product combinations, it covers an enormous search space with minimal storage, achieving high compression ratios and rapid approximate computations through index mapping. This characteristic enables PQ to significantly reduce storage and computational overhead while maintaining acceptable accuracy loss[36].

The loss in PQ quantization stems from the destruction of correlations due to subspace partitioning and information truncation from discrete quantization[37]. Mathematically, it manifests as:

1) Reconstruction Error:
   Each subspace generates K centroids via K-means ($C = \{c_{m,k}\}_{k=1}^{K}$). Subvector $V_m$is quantized to its nearest centroid c_m^*, with reconstruction error:

$$\mathcal{L}_{\text{recon}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{m=1}^{M} \|v_m^{(i)} - c_m^*\|_2^2$$

   This error decreases as K increases.
2) Subspace Partitioning Error:
   A D-dimensional vector is partitioned into M subspaces (each of dimension d=D/M). For an original vector v$\in R^D$, partitioned as $[v_1, v_2, \ldots, v_M]$. Independent quantization across subspaces destroys inter-dimensional correlations, and the lower bound of the reconstruction error for $v^\wedge = [q_1(v_1), q_2(v_2), \ldots, q_M(v_M)]$ is:

$$\|v - \hat{v}\|2^2 \geq \sum m = 1^M \|v_m - q_m(v_m)\|_2^2$$

   When subspace partitioning does not align with the principal components of the data distribution (e.g., without PCA preprocessing), the error increases significantly.

2.4. MDS Mapping Method:

To ensure a mapping relationship between indices and cluster centers in the codebook, multiple mathematical methods can map point sets in n-dimensional space to one-dimensional values while preserving original distance relationships as much as possible. Examples include PCA projection[38], one-dimensional MDS embedding[39], random projections, and kernel PCA. PCA is simpler but preserves variance rather than pairwise distances. The one-dimensional MDS embedding method suits small datasets by directly optimizing distance preservation. We adopt the one-dimensional MDS embedding algorithm here.

Classical one-dimensional MDS embedding maps points in high-dimensional space to a one-dimensional line, ensuring that Euclidean distances (absolute differences) between projected points approximate the original distances as closely as possible. It essentially solves for the optimal linear projection direction through eigenvalue decomposition.

Core ideas:

Given m n-dimensional points x1,x2,...,xm, the goal is to find a mapping:

$$f : \mathbb{R}^n \rightarrow \mathbb{R}, f(x_I) = z_i$$

the one-dimensional distance satisfies:

$$|z_i - z_j| \approx \|x_i - x_j\|, \forall i, j$$

Optimization Objective: Minimize the stress function (Stress).

$$Stress = \sqrt{\frac{\sum_{i<j}(\|x_i - x_j\| - \|z_i - z_j\|)^2}{\sum_{i<j}\|x_i - x_j\|^2}}$$

Algorithm 2 MDS One-Dimensional Embedding Algorithm Steps:

```
Input: m×ndata matrixX,distance metric (default: Euclidean distance).
Output: One-dimensional coordinate vector $z = (z_1, z_2, \ldots, z_m) \in R_m$

    Setps:
1:  Compute distance matrix
        $D_{ij} = \|x_i - x_j\|, \quad D \in \mathbb{R}^{m\times n}$
2:  Square the distance matrix
        $D^{(2)} = \left[D_{ij}^2\right]$
3:  Double-centering
        $B = -\frac{1}{2} J D^{(2)} J$
      Where: J is the centering matrix: $J = I_m - \frac{1}{m}\mathbf{1}\mathbf{1}^T$,1 is an all-ones column vector,B geometrically
    represents the inner product matrix,$B = ZZ^T$ (Z is centered coordinates)
4:  Eigen decomposition
        $B = V \wedge V^T, \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_m)$
    Sort eigenvalues:$\lambda_1 > \lambda_2 > ... > \lambda_m$
5:  Select the largest eigenvalue
      Take $\lambda_{max} = \lambda_1$ ,and its corresponding eigenvector$v_1 = [v_{11}, v_{12}, ..., V_{1m}]^T$
6:  Compute one-dimensional coordinates
        $z = \sqrt{\Lambda_{max}} v_1$
```

Loss Analysis:

During one-dimensional embedding, the loss is primarily caused by distance distortion, which can be strictly quantified using the stress function

$$Stress = \sqrt{\frac{\sum_{i<j}(\|x_i - x_j\| - \|z_i - z_j\|)^2}{\sum_{i<j}\|x_i - x_j\|^2}}$$

According to the Johnson-Lindenstrauss lemma, when embedding any n points from high-dimensional space to one dimension, distance distortion inevitably exists:

$$\exists i, j \text{ s.t. } \frac{|d_{ij} - \hat{d}ij|}{dij} \geq \epsilon > 0$$

where ε is determined by the intrinsic dimensionality of the data.

Theoretical Lower Bound of Loss:For any dataset, the minimum achievable stress for one-dimensional embedding satisfies:

$$Stress_{min} \geq \sqrt{1 - \frac{\lambda_1}{\sum_{k=1}^{n} \lambda_k}}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$are the eigenvalues of the double-centered matrix B.

3. Method

This research comprehensively addresses factors such as feature imbalance, communication efficiency, and privacy preservation, designing a multi-stage dimensionality reduction clustering framework for vertical federated learning scenarios.

This framework is implemented based on PQ quantization and MDS techniques. After privacy-preserving alignment across multiple clients, the product quantization technique replaces traditional local model training. Algorithmically, PQ fundamentally relies on k-means clustering principles, compressing high-dimensional features into low-dimensional codes to significantly reduce communication overhead.Regarding privacy protection design: directly uploading codebooks (cluster center sets) risks exposing original data distributions. We observe clustering effectiveness heavily depends on relative distances between cluster centers. Thus, we employ the MDS one-dimensional embedding algorithm, $f : \mathbb{R}^n \to \mathbb{R}, f(x_I) = z_i$ ensuring:

$$|z_i - z_j| \approx \|x_i - x_j\|, \forall i, j$$

The server executes a one-shot clustering algorithm on distance-preserving indices to obtain the clustering structure, achieving maximized intra-group similarity and Maximized inter-group divergence

This approach preserves raw codebook privacy locally while maintaining distance relationships essential for clustering through indices, ensuring algorithmic performance and further enhancing communication efficiency.

The adoption of multi-stage dimensionality reduction instead of direct dimensionality reduction is partially due to significant disparities in feature counts across clients. Direct reduction would bias clustering results toward certain features under such imbalances.

Figure 1 shows the process of the algorithm.

3.1. Algorithm Design

Suppose there is a federated learning framework with m clients. The global dataset is D with global dimension dim, distributed across clients. D^g denotes the dataset of the g-th client, where data points in D^g are $dim^g - dimensional$ vectors. There are $\sum dim^g = dim$. Define sub_dim as the subspace dimensionality in PQ quantization, and sub_k as the number of cluster centers per subspace in PQ quantization. $Codebook^g$ represents the codebook of the g-th client, and $code^g{}_i$ denotes the cluster center set of the i-th subspace for the g-th client,$code^g{}_i$ is the data index set of the g-th client, and B is the global data index set.

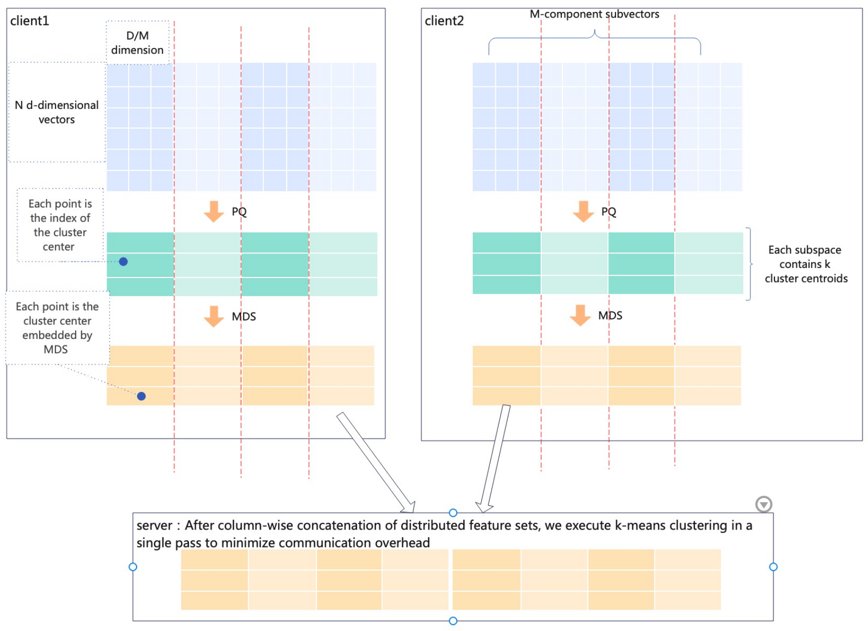
client1
D/M dimension
N d-dimensional vectors
PQ
Each point is the index of the cluster center
MDS
Each point is the cluster center embedded by MDS
client2
M-component subvectors
PQ
Each subspace contains k cluster centroids
MDS
server：After column-wise concatenation of distributed feature sets, we execute k-means clustering in a single pass to minimize communication overhead

Figure 1

Algorithm Mainly Consists of the Following Steps:

1. Encrypted Entity Alignment
   Select common samples: Extract common samples from each party's dataset.
2. Local Initialization and Training of PQ Quantizer:
   1) Pad dimensions: Determine whether the local dimension dim is divisible by sub_dim. If not divisible, pad dimensions.
      First, calculate the number of dimensions p to pad:

$$dim' = sub\_dim' \times sub\_k$$

$$p = dim' - dim$$

      Then, pad the original vector by appending p zeros to its end.

$$v' = [v_1, v_2, \ldots, v_{\text{dim}}, \underbrace{0, 0, \ldots, 0}_{p}]$$

   2) Generate subspace codebooks based on training data.
3. Secondary mapping:
   Perform secondary mapping on cluster centers in subspace codebooks using MDS one-dimensional embedding algorithm, then use the normalized mapped values as codebook indices.
4. Data transmission:
   Codebooks are stored locally. Transfer the indices to the server side.
5. Server-side global cluster center aggregation:
   Execute k-means algorithm on the indices uploaded by clients at the server side to obtain abstract global cluster centers.
   When using abstract cluster centers, apply the same mapping operation to local data, then compute distances to global abstract cluster centers to determine true cluster assignments.
   The algorithm requires only one round of communication.
   Table 1 Symbol Explanation:

Table 1. Symbol Explanation:

Symbol	Description
D	Description
sub_dim	Subspace data dimensionality
sub_k	k-value for the g-th client
D^g	Dataset of the g-th client
$codebook^g$	Codebook of the g-th client
${code^g}_i$	Cluster center set of the i-th subspace for the g-th client
B^g	Data index set of the g-th client
B	Global data index set

Algorithm Pseudocode:

Algorithm 3 Algorithm:

Input: Distributed dataset$D = D_1, D_2, \ldots, D_m$, where D_g is the data of the g-th client. Number of clusters k. Maximum iterations T.
Output: Global cluster centers $C = c_1, c_2, \ldots, c_k$.

```
Setps:
1:  RSA Private Set Intersection
    1)  Server generates RSA key pair and broadcasts public key
    2)  Clients blind their own elements
    3)  Server signs blinded elements
    4)  Clients unblind received signatures
    5)  Clients send unblinded signatures to server
    6)  Server computes intersection signatures and sends to clients
    7)  Clients map signatures back to original elements
2:  Model Training:
    1)  Server distributes parameters: sub_dim (subspace dimensionality), ks (number of subspace
        cluster centers)
    2)  At local client (g-th client):
          If dim not divisible by sub_dim, pad dimensions with zeros.
          codebook=pq.train(sub_dim,ks)
          for code in codebook:
            MDS dimensionality reduction:code = mds(code)
            Normalization:code = normalize(code)
            Add to code_list:code_list.append(code)
          Convert data to code indices: data_codes = pq.encode(data,code_list)
          Upload encrypted data indices to server
    3)  Server aggregation:   Collect data indices from all clients.
          Global index set $B = \Phi$
          for client in client_list:      Column-wise merge:$B = B \cup B^g$
          In abstract index set B:
          Initialize cluster centers by randomly selecting k data points as initial global cluster
        centers.
          Perform k-means clustering on sample set using initialized global centers to obtain final
        cluster centers C: $c_1, c_2, \ldots, c_k$
```

3.2. Privacy Enhancement

The privacy protection mechanism of this algorithm derives from two aspects:For one thing PQ quantization technology reduces dimensionality and processes raw data, preventing its direct upload to the server.for another, Locally retained PQ codebooks avoid leakage of original data distributions.

Additionally, to defend against attacks like membership inference analysis, differential privacy noise protection can be incorporated. Differential privacy provides a rigorous framework ensuring analytical results never reveal individual information. By adding noise to data or intermediate results, attackers cannot determine any individual's presence in the dataset. According to differential privacy serial/parallel theorems [40]:

1. Serial Composition:For a given dataset D,assume there exists random algorithms $M_1, M_2, \ldots, M_n$, with privacy budgets$\epsilon_1, \epsilon_2, \ldots, \epsilon_n$respectively. The composition algorithm $M(M_1(D), M_2(D), \ldots, M_n(D))$provides $(\sum_{i=1}^{n} \epsilon_i$ -DP protection. That is, for the same dataset, applying a series of differentially private algorithms sequentially provides protection equivalent to the sum of privacy budgets.
2. Parallel Composition:For disjoint datasets$D_1, D_2, \ldots, D_n$,assume there exists random algorithms $M_1, M_2, \ldots, M_n$,with privacy budgets $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ respectively,respectively $M(M_1(D_1), M_2(D_2), \ldots, M_n(D_n))$ provides $(max\epsilon_i)$-DP privacy budgets. That is,for disjoint datasets, applying different differentially private algorithms separately in parallel provides privacy protection equivalent to the maximum privacy budget among the composed algorithms.

Adding differential privacy noise to raw data or codebooks can effectively enhance privacy protection.

4. Experiments and Results

4.1. Experimental Settings

4.1.1. Dataset

The MNIST dataset is adopted, with columns split according to client feature ratios, and different subsets are distributed to different clients.

4.1.2. Parameter Settings

- Total number of clients:2
- Total number of client features: 784
- Client feature ratios: (1:1),(1:6),(1:13)
- PQ quantization subspace dimensions: 1,2,4,8
- Number of PQ quantization cluster centers per subspace: 10,64,128,256

4.1.3. Evaluation Metrics

The evaluation employs NMI and ARI metrics.

4.2. Performance Analysis

Centralized data clustering serves as the baseline for validation.

4.2.1. Comparison between proposed method and centralized kmeans

This experiment involves two clients, each with 392 data dimensions at a 1:1 ratio. As shown in Table 2, the proposed algorithm achieves slightly higher average NMI and ARI values under various subspace dimensions and cluster center counts than centralized k-means, indicating effective extraction of original data features.When the subspace dimension is 1, equivalent to performing kmeans on each data column and replacing values with cluster centers, PQ loss is minimized. Data confirms superior performance in this scenario. Performance improves with fewer cluster centers per subspace, partly because the sparse MNIST dataset can be effectively represented by fewer clusters.

Table 2. Algorithm Performance Comparison

Dispersion Coefficient	Hash Dimension	Proposed Method		Without MDS Algorithm Using Codebook Indices at Server		Without MDS Algorithm, Restored Data at Server		Centralized Algorithm	
		NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
1	10	0.53403	0.42542	0.50209	0.41110	0.51871	0.40490	0.49581	0.36387
2	10	0.49353	0.37211	0.47981	0.38815	0.49406	0.38168		
4	10	0.51018	0.38628	0.44081	0.33740	0.52128	0.40873		
8	10	0.52476	0.43289	0.42186	0.31696	0.52207	0.40520		
1	64	0.49707	0.36712	0.48437	0.37222	0.51585	0.40136		
2	64	0.53773	0.42763	0.44766	0.33290	0.49617	0.36369		
4	64	0.50124	0.37786	0.42137	0.32132	0.49500	0.38318		
8	64	0.48329	0.39052	0.41128	0.34143	0.49390	0.36302		
1	128	0.48375	0.36266	0.50666	0.41889	0.49081	0.36068		
2	128	0.49855	0.38596	0.46257	0.35915	0.49043	0.36039		
4	128	0.48026	0.35906	0.42605	0.34344	0.49403	0.38208		
8	128	0.47704	0.37444	0.40413	0.31093	0.48159	0.37014		
1	256	0.52029	0.40768	0.49181	0.40905	0.49049	0.36069		
2	256	0.48873	0.36059	0.49902	0.39875	0.48150	0.35965		
4	256	0.49801	0.39486	0.43289	0.35194	0.49628	0.36436		
8	256	0.46309	0.36877	0.41015	0.34877	0.48375	0.36400		

4.2.2. Impact of MDS on Algorithm Performance

To evaluate MDS's impact, we compare against two alternative implementations without MDS: one clusters codebook indices at the server, and another reconstructs data using PQ quantization at the server. Privacy comparison: direct index clustering > proposed method > data reconstruction.

As Table 2 shows, performance comparison: proposed method ≥ data reconstruction   direct index clustering. When subspace dimension=1, direct index clustering performs comparably to other methods. However, its performance declines sharply as PQ subspace dimension increases, indicating insufficient capture of data distance features. The proposed method better captures data distance features, yielding slightly superior performance over data reconstruction.

4.2.3. Impact of Client Feature Quantity on Performance

Adjusting feature ratios among clients (1:1, 1:6, 1:13) shows algorithm performance remains relatively stable within a narrow range, confirming client feature ratios do not affect performance.

Table 3. Experimental Results : Impact of Client Feature Quantity on Performance

Number of Clients	Feature Ratio Among Clients	Subspace Dimension	Cluster Centers per Subspace	Proposed Method	
				NMI	ARI
2	1:1	1	10	0.53403	0.42542
2	1:1	2	10	0.49353	0.37211
2	1:1	4	10	0.51018	0.0.38628
2	1:1	8	10	0.52476	0.43289
2	1:6	1	10	0.51950	0.40631
2	1:6	2	10	0.51160	0.39005
2	1:6	4	10	0.51162	0.38794
2	1:6	8	10	0.52514	0.42123
2	1:13	1	10	0.49672	0.36139
2	1:13	2	10	0.49207	0.36924
2	1:13	4	10	0.51211	0.38790
2	1:13	8	10	0.53030	0.44559

5. Conclusion

Based on the characteristics of clustering algorithms, this study transforms the core contradiction of federated clustering—"data utility vs. privacy preservation vs. communication efficiency"—into a verifiable distance-preserving optimization problem, providing a secure clustering implementation framework for vertical federated learning. The proposed algorithm can address communication challenges in vertical federated learning while preserving privacy.

Addressing three core challenges of K-Means clustering in Vertical Federated Learning—inadequate privacy protection, excessive communication overhead, and feature dimension imbalance—this paper innovatively proposes a multi-level dimensionality reduction federated clustering framework integrating Product Quantization (PQ) and Multidimensional Scaling (MDS). By compressing original high-dimensional features into low-dimensional codes via PQ and further reducing dimensionality through one-dimensional MDS embedding on codebooks, communication efficiency is significantly enhanced. Privacy protection is achieved through: 1) dimensionality reduction lowering data precision and transmission volume, and 2) mapping sensitive codebooks to secure indices via MDS embedding, transmitting only distance-preserving indices to the server. Original codebooks and feature data remain exclusively on local clients. Experimental results demonstrate that the algorithm ensures stable clustering performance while enhancing communication efficiency and privacy. The multi-stage reduction framework does not degrade performance and even slightly outperforms centralized algorithms.

Beyond clustering applications, this multi-stage dimensionality reduction framework can extend to other machine learning algorithms in vertical federated settings. Future work may explore optimizing and integrating diverse privacy mechanisms, such as combining Secure Multi-Party Computation (SMPC) , homomorphic encryption or implementing multi-tiered privacy policies.

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial intelligence and statistics. PMLR, 2017, pp. 1273–1282.
2. Stallmann, M.; Wilbik, A. Towards Federated Clustering: A Federated Fuzzy-Means Algorithm (FFCM). *arXiv preprint arXiv:2201.07316* **2022**.
3. Chen, M.; Shlezinger, N.; Poor, H.V.; Eldar, Y.C.; Cui, S. Communication-efficient federated learning. *Proceedings of the National Academy of Sciences* **2021**, *118*, e2024789118.
4. Zhu, H.; Xu, J.; Liu, S.; Jin, Y. Federated learning on non-IID data: A survey. *Neurocomputing* **2021**, *465*, 371–390.
5. Zhou, X.; Yang, G. Communication-efficient and privacy-preserving large-scale federated learning counteracting heterogeneity. *Information Sciences* **2024**, *661*, 120167.
6. Mohammadi, N.; Bai, J.; Fan, Q.; Song, Y.; Yi, Y.; Liu, L. Differential privacy meets federated learning under communication constraints. *IEEE Internet of Things Journal* **2021**, *9*, 22204–22219.
7. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning* **2021**, *14*, 1–210.
8. Wei, K.; Li, J.; Ma, C.; Ding, M.; Wei, S.; Wu, F.; Chen, G.; Ranbaduge, T. Vertical federated learning: Challenges, methodologies and experiments. *arXiv preprint arXiv:2202.04309* **2022**.
9. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* **2019**, *10*, 1–19.
10. Li, J.; Wei, H.; Liu, J.; Liu, W. FSLEdge: An energy-aware edge intelligence framework based on Federated Split Learning for Industrial Internet of Things. *Expert Systems with Applications* **2024**, *255*, 124564.
11. Khan, L.U.; Pandey, S.R.; Tran, N.H.; Saad, W.; Han, Z.; Nguyen, M.N.; Hong, C.S. Federated learning for edge networks: Resource optimization and incentive mechanism. *IEEE Communications Magazine* **2020**, *58*, 88–93.
12. Rieke, N.; Hancox, J.; Li, W.; Milletari, F.; Roth, H.R.; Albarqouni, S.; Bakas, S.; Galtier, M.N.; Landman, B.A.; Maier-Hein, K.; et al. The future of digital health with federated learning. *NPJ digital medicine* **2020**, *3*, 119.

13. Wu, Z.; Hou, J.; He, B. Vertibench: Advancing feature distribution diversity in vertical federated learning benchmarks. *arXiv preprint arXiv:2307.02040* **2023**.
14. Khan, A.; ten Thij, M.; Wilbik, A. Communication-efficient vertical federated learning. *Algorithms* **2022**, *15*, 273.
15. Cheng, K.; Fan, T.; Jin, Y.; Liu, Y.; Chen, T.; Papadopoulos, D.; Yang, Q. Secureboost: A lossless federated learning framework. *IEEE intelligent systems* **2021**, *36*, 87–98.
16. Li, Z.; Wang, T.; Li, N. Differentially private vertical federated clustering. *arXiv preprint arXiv:2208.01700* **2022**.
17. Zhao, F.; Li, Z.; Ren, X.; Ding, B.; Yang, S.; Li, Y. VertiMRF: Differentially Private Vertical Federated Data Synthesis. In Proceedings of the Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2024, pp. 4431–4442.
18. Mazzone, F.; Brown, T.; Kerschbaum, F.; Wilson, K.H.; Everts, M.; Hahn, F.; Peter, A. Privacy-Preserving Vertical K-Means Clustering. *arXiv preprint arXiv:2504.07578* **2025**.
19. Li, C.; Ding, S.; Xu, X.; Guo, L.; Ding, L.; Wu, X. Vertical Federated Density Peaks Clustering under Nonlinear Mapping. *IEEE Transactions on Knowledge and Data Engineering* **2024**.
20. Ahmed, M.; Seraj, R.; Islam, S.M.S. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics* **2020**, *9*, 1295.
21. Han, J.; Kamber, M.; Mining, D. Concepts and techniques. *Morgan kaufmann* **2006**, *340*, 94104–103205.
22. Mary, S.S.; Selvi, T. A study of K-means and cure clustering algorithms. *Int. J. Eng. Res. Technol* **2014**, *3*, 1985–1987.
23. GAO, Y.; XIE, Y.; DENG, H.; ZHU, Z.; ZHANG, Y. A Privacy-preserving Data Alignment Framework for Vertical Federated Learning. *J. Electron. Inf. Technol.* **2024**, *46*, 3419–3427.
24. Yang, L.; Chai, D.; Zhang, J.; Jin, Y.; Wang, L.; Liu, H.; Tian, H.; Xu, Q.; Chen, K. A survey on vertical federated learning: From a layered perspective. *arXiv preprint arXiv:2304.01829* **2023**.
25. Liu, Y.; Kang, Y.; Zou, T.; Pu, Y.; He, Y.; Ye, X.; Ouyang, Y.; Zhang, Y.Q.; Yang, Q. Vertical federated learning: Concepts, advances, and challenges. *IEEE Transactions on Knowledge and Data Engineering* **2024**, *36*, 3615–3634.
26. Zhao, Z.; Mao, Y.; Liu, Y.; Song, L.; Ouyang, Y.; Chen, X.; Ding, W. Towards efficient communications in federated learning: A contemporary survey. *Journal of the Franklin Institute* **2023**, *360*, 8669–8703.
27. Yang, H.; Liu, H.; Yuan, X.; Wu, K.; Ni, W.; Zhang, J.A.; Liu, R.P. Synergizing Intelligence and Privacy: A Review of Integrating Internet of Things, Large Language Models, and Federated Learning in Advanced Networked Systems. *Applied Sciences* **2025**, *15*, 6587.
28. Zhang, C.; Li, S. State-of-the-art approaches to enhancing privacy preservation of machine learning datasets: A survey. *arXiv preprint arXiv:2404.16847* **2024**.
29. Qi, Z.; Meng, L.; Li, Z.; Hu, H.; Meng, X. Cross-Silo Feature Space Alignment for Federated Learning on Clients with Imbalanced Data **2025**.
30. Hu, K.; Xiang, L.; Tang, P.; Qiu, W. Feature norm regularized federated learning: utilizing data disparities for model performance gains. In Proceedings of the Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, 2024, pp. 4136–4146.
31. Aramian, A. Managing Feature Diversity: Evaluating Global ModelReliability in FederatedLearning for Intrusion Detection Systems in IoT, 2024.
32. Johnson, A. A Survey of Recent Advances for Tackling Data Heterogeneity in Federated Learning **2025**.
33. Jegou, H.; Douze, M.; Schmid, C. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* **2010**, *33*, 117–128.
34. Konečnỳ, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* **2016**.
35. Yue, K.; Jin, R.; Wong, C.W.; Baron, D.; Dai, H. Gradient obfuscation gives a false sense of security in federated learning. In Proceedings of the 32nd USENIX Security Symposium (USENIX Security 23), 2023, pp. 6381–6398.
36. Ge, T.; He, K.; Ke, Q.; Sun, J. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence* **2013**, *36*, 744–755.
37. Xiao, S.; Liu, Z.; Shao, Y.; Lian, D.; Xie, X. Matching-oriented product quantization for ad-hoc retrieval. *arXiv preprint arXiv:2104.07858* **2021**.
38. Deisenroth, M.P.; Faisal, A.A.; Ong, C.S. *Mathematics for machine learning*; Cambridge University Press, 2020.

39. Izenman, A.J. Linear Dimensionality Reduction. *Springer New York* **2013**.
40. Vadhan, S.; Zhang, W. Concurrent Composition Theorems for Differential Privacy **2022**.