Personalized Federated Learning under Mixture of Distributions

Yue Wu [*1] **Shuaicheng Zhang** [*2] **Wenchao Yu** [3] **Yanchi Liu** [3] **Quanquan Gu** [1] **Dawei Zhou** [2] **Haifeng Chen** [3] **Wei Cheng**[✉3]

Abstract

The recent trend towards Personalized Federated Learning (PFL) has garnered significant attention as it allows for the training of models that are tailored to each client while maintaining data privacy. However, current PFL techniques primarily focus on modeling the conditional distribution heterogeneity (i.e. concept shift), which can result in suboptimal performance when the distribution of input data across clients diverges (i.e. covariate shift). Additionally, these techniques often lack the ability to adapt to unseen data, further limiting their effectiveness in real-world scenarios. To address these limitations, we propose a novel approach, FedGMM, which utilizes Gaussian mixture models (GMM) to effectively fit the input data distributions across diverse clients. The model parameters are estimated by maximum likelihood estimation utilizing a federated Expectation-Maximization algorithm, which is solved in closed form and does not assume gradient similarity. Furthermore, FedGMM possesses an additional advantage of adapting to new clients with minimal overhead, and it also enables uncertainty quantification. Empirical evaluations on synthetic and benchmark datasets demonstrate the superior performance of our method in both PFL classification and novel sample detection.

1. Introduction

The sheer volume of data at our disposal today is often sequestered in isolated silos, making it challenging to access and utilize. Federated Learning (FL) presents a groundbreaking solution to this conundrum, enabling collaborative learning across distributed data sources without compromising the confidential nature of the original training data, while also being fully compliant with government regulations (Lim et al., 2020; Aledhari et al., 2020; Mothukuri et al., 2021). This method has drawn a lot of attention in recent years since it enables model training on diverse, decentralized data while protecting privacy and security. In many applications, the model needs to be adjusted for each device or user, notably the *cross-device* scenarios. These situations are the focus of Personalized Federated Learning (PFL), which tries to provide client-specific model parameters for a certain model architecture. In this scenario, each client aims to obtain a local model with a respectable test result on its own local data distribution (Wang et al., 2019).

In order to cater to the unique needs of individual clients and address the statistical diversity that exists among them, existing PFL studies frequently resort to an elegant amalgamation of federated learning and other sophisticated approaches, such as meta-learning (Sim et al., 2019), client clustering (Ghosh et al., 2020), multi-task learning (Marfoq et al., 2021), knowledge distillation (Zhu et al., 2021), and the lottery ticket hypothesis(Wang et al., 2022), to achieve the desired level of personalization. For example, clients can be assigned to many clusters, and clients in the same cluster are assumed to use the same model via clustered FL techniques (Ghosh et al., 2020). To train a global model as a meta-model and then fine-tune the parameters for each client, several researchers have embraced meta-learning based methodologies (Sim et al., 2019; Jiang et al., 2019). Wang et al. (Wang et al., 2022) suggested utilizing a routing hypernetwork to expertly curate and assemble modular blocks from a globally shared modular pool, in order to craft bespoke local networks through the application of the lottery ticket theory. A recent study (Marfoq et al., 2021) that leveraged the multi-task learning concept posited that each client's data distribution was a composite of M underlying distributions, and proposed the use of a linear mixture model to make tailored decisions based on the shared components among them. It optimizes the varying conditional distribution $\mathbb{P}_c(\mathbf{y}|\mathbf{x})$ under the assumption that the marginal distributions $\mathbb{P}_c(\mathbf{x}) = \mathbb{P}_{c'}(\mathbf{x})$ are the same for all clients (Assumption 2 in (Marfoq et al., 2021)).

While these approaches are adept at addressing the issue of

[*]Equal contribution [1]Department of Computer Science, University of California, Los Angeles, USA. [2]Department of Computer Science, Virginia Tech, Blacksburg, USA. [3]NEC Laboratories America, Princeton, USA. Correspondence to: Wei Cheng <weicheng@nec-labs.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

conditional distribution heterogeneity, commonly referred to as *concept shift*, within PFL, they fall short in addressing the more comprehensive issue of general statistical heterogeneity which encompasses other forms of variability, such as feature distribution skew (i.e., *covariate shift*) (Kairouz et al., 2021), that is each client has different input marginal distributions (i.e., $\mathbb{P}_c(\mathbf{x}) \neq \mathbb{P}_{c'}(\mathbf{x})$). For example, even with handwriting recognition, users may exhibit variations in stroke length, slant, and other nuances when writing the same phrases. In reality, data on each client may be deviated from being identically distributed, say, $\mathbb{P}_c \neq \mathbb{P}_{c'}$ for clients c and c'. That is, the joint distribution $\mathbb{P}_c(\mathbf{x}, \mathbf{y})$ (can be rewritten as $\mathbb{P}_c(\mathbf{y}|\mathbf{x})\mathbb{P}_c(\mathbf{x})$) may be different across clients. We refer to it as the "*joint distribution heterogeneity*" problem. Current approaches fall short of fully encapsulating the intricacies of the variations in the joint distribution among clients, owing to their tendency to impose a presumption of constancy on one term while adjusting the other (Marfoq et al., 2021; Zhu et al., 2021).

Besides, cross-device federated learning applications are often faced with a phenomenon known as *client drift*. This occurs when the learning model is deployed in a real-world online setting, and the distribution of inputs it encounters differs from the distribution it was trained on. As a result, the model's performance may be severely impacted. For instance, a PFL model trained on the historical medical records of a specific patient population may exhibit significant regional or demographic biases when tested on a new patient (Shukla & Marlin, 2019; Purushotham et al., 2017). To mitigate this, it is crucial to develop a cutting-edge PFL methodology that can easily adapt to new clients while incorporating the capability to perform uncertainty quantification. The key to achieving this lies in the ability to identify and account for any outliers that may deviate from the established training data distribution. Such a methodology would elevate PFL to a practical solution, enabling it to be deployed in a wide range of applications with confidence.

In this study, we propose a Federated Gaussian Mixture Model (FedGMM) approach, which utilizes Gaussian mixture models to tackle the aforementioned issues. Our approach operates under the assumption that the joint distribution of data is a linear mixture of several base distributions. FedGMM builds up PFL by maximizing the log-likelihood of the observed data. To maximize the log-likelihood of the mixture model, we suggest a federated Expectation-Maximization (EM) algorithm for model parameter learning. The update rule for the Gaussian components has a closed-form solution and does not resort to gradient methods. To ensure convergence of the EM update rule, we incorporate our algorithm with the theoretical analysis of federated EM for GMMs. The Gaussian parameters inferred by the server offer a detailed global statistical descriptor of the data, and can be applied for various purposes, including density estimation and clustering, etc.

To sum up, our contributions are as follows:

- For the first time, this study explicitly addresses the challenging issue of joint distribution heterogeneity in PFL. Our approach serves as a novel solution to this problem, enabling the capability to perform uncertainty quantification. Furthermore, the proposed approach is designed to be highly flexible, allowing for easy inference of new clients, who did not participate in the training phase. This is achieved by learning their personalized mixture weights with a small computational overhead.
- Our method presents a highly adaptable framework that is independent of supervised discriminative learning models, making it easily adaptable to other learning models. The model parameters are learned in an end-to-end fashion via maximum likelihood estimation, specifically a federated Expectation-Maximization (EM) algorithm. Furthermore, we have theoretically analyzed the convergence bound of our log-likelihood function, providing a solid theoretical foundation for our approach. The federated learning process for the Gaussian mixture is a novel federated unsupervised learning approach, which may be of independent interest.
- In the experiments, we assessed our technique on both artificial and real-world datasets to validate its efficacy in simulating the mixture joint distribution of PFL data for classification, as well as its capacity to discover novel samples. The outcomes show that our technique performs significantly better than the state-of-the-art (SOTA) baselines.

2. Problem Formulation

Notations We use lowercase letters/words to denote scalars, lowercase bold letters/words to denote vectors, and uppercase bold letters to denote matrices. We use $\|\cdot\|$ to indicate the Euclidean norm. We also use the standard O and Ω notations. For a positive integer N, $[N] := \{1, 2, \ldots, N\}$.

We focus on the personalized federated classification task. Suppose there exist C clients. Each client $c \in [C]$ has its own dataset of size N_c, where a sample $\mathbf{s}_{c,i} = (\mathbf{x}_{c,i}, \mathbf{y}_{c,i})$ is assumed to be drawn from its distribution $\mathbb{P}_c(\mathbf{x}, \mathbf{y})$. The local data distribution $\mathbb{P}_c(\mathbf{x}, \mathbf{y})$ can be different. Therefore, it is natural to choose different hypotheses $h_c \in \mathcal{H}$ for each client c. Here, $\mathcal{H}$ can be some general and highly expressive function class like neural networks.

In this work, we use $h_c(\mathbf{x}, \mathbf{y})$ (sometimes denoted by $h_c(\mathbf{s})$) to represent the likelihood of the sample $\mathbf{s} = (\mathbf{x}, \mathbf{y})$. For classification tasks, the goal is naturally to achieve the ex-

pected maximum log-likelihood:

$$\forall c \in [C], \qquad \max_{h_c \in \mathcal{H}} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\mathbb{P}_c} \big[\log\big(h_c(\mathbf{x},\mathbf{y})\big)\big].$$

2.1. Mixture of Joint Distributions

To facilitate federated learning, it is necessary to pose assumptions on how the distributions of different clients are similar, such that the data from one client can be utilized to improve the learning of other clients. To this end, we adopt the simple but general assumption that the distribution of one client is a mixture of several base distributions:

$$\mathbb{P}_c(\mathbf{x},\mathbf{y}) = \sum_{m=1}^{M} \pi_c^*(m)\mathbb{P}^{(m)}(\mathbf{x},\mathbf{y}), \forall c \in [C]. \tag{1}$$

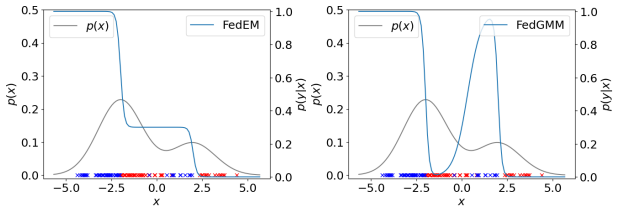
Here, $\mathbb{P}^{(m)}$ denotes the m-th base distribution that is shared across all clients, while $\pi_c^*(m)$ can differ for different client c. With this presumption, we may benefit from the fact that any client can gain knowledge from datasets collected from all other clients but eschew clear statistical assumptions about local data distributions, and the heterogeneous joint distribution can be accurately modeled as well. This assumption in a federated setting was first introduced by Marfoq et al. (2021) and was named FedEM. What differs is that Marfoq et al. (2021) additionally assumes that the marginal distributions of each base distribution $\mathbb{P}^{(m)}(\mathbf{x})$ are the same. This implies that every client has the same input distribution $\mathbb{P}_c(\mathbf{x}) = \mathbb{P}_{c'}(\mathbf{x})$, while the conditional distributions $\mathbb{P}_c(\mathbf{y}|\mathbf{x})$ are different across different clients, and admit a form of linear mixtures.

$$\mathbb{P}_c(\mathbf{y}|\mathbf{x}) = \sum_{m=1}^{M} \pi_c^*(m)\mathbb{P}^{(m)}(\mathbf{y}|\mathbf{x}). \tag{2}$$

This assumption simplifies what the clients must learn: the mixture weights $\pi_c^*(\cdot)$ and the conditional distribution $\mathbb{P}^{(m)}(\mathbf{y}|\mathbf{x})$. In other words, the training objective will degenerate to minimizing the cross entropy for classification, rather than to maximizing the likelihood of $\{(\mathbf{x}_{c,i},\mathbf{y}_{c,i})\}_{i\in[N_c]}$. In contrast, if we allow $\mathbb{P}^{(m)}(\mathbf{x})$ to be different, then the conditional probability will appear in the following form:

$$\mathbb{P}_c(\mathbf{y}|\mathbf{x}) = \frac{\sum_{m=1}^{M} \pi_c^*(m)\mathbb{P}^{(m)}(\mathbf{y}|\mathbf{x})\mathbb{P}^{(m)}(\mathbf{x})}{\sum_{m=1}^{M} \pi_c^*(m)\mathbb{P}^{(m)}(\mathbf{x})}. \tag{3}$$

It is clear that aside from learning the conditional distribution $\mathbb{P}^{(m)}(\mathbf{y}|\mathbf{x})$, to faithfully characterize the conditional probability, we also need to learn the base input distribution $\mathbb{P}^{(m)}(\mathbf{x})$. Figure 1 shows that when $\mathbb{P}^{(m)}(\mathbf{x})$ are indeed different, there will be a fundamental gap between the classification errors.



(a) Same $\mathbb{P}_c(\mathbf{x})$ (b) Different $\mathbb{P}_c(\mathbf{x})$

Figure 1. An illustrative example: data are drawn from a mixture of two distributions: $\mathbb{P}^{(1)}(x) = \mathcal{N}(x;-2,1.5)$, $y = f^{(1)}(x) = \mathbb{1}\{x < -2\}$ and $\mathbb{P}^{(2)}(x) = \mathcal{N}(x;2,1.5)$, $y = f^{(2)}(x) = \mathbb{1}\{x < 2\}$. Figure (a) shows how an algorithm that assumes $\mathbb{P}^{(1)} = \mathbb{P}^{(2)}$ fails to predict the label correctly. Figure (b) shows that once the input distribution is considered, the model can fully capture the data distribution.

3. Proposed Method

3.1. Motivation

It is widely known that the likelihood maximization problem under a linear mixture structure can be solved by the Expectation-Maximization (EM) technique. Consider the following learning objective: $\forall c \in [C]$,

$$\max_{\pi_c,\boldsymbol{\theta},\boldsymbol{\phi}} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\mathbb{P}_c} \Big[\log\Big(\sum_{m=1}^{M} \pi_c(m)P_{\boldsymbol{\phi}_m}(\mathbf{x})P_{\boldsymbol{\theta}_m}(\mathbf{y}|\mathbf{x})\Big)\Big].$$

Similar to Marfoq et al. (2021), this kind of problem can be solved by optimizing the parameters $\boldsymbol{\phi}_m$ and $\boldsymbol{\theta}_m$ separately via gradient methods. The difficulty in learning $\mathbb{P}^{(m)}(\mathbf{x})$ lies in that most modern density estimation models (such as auto-regressive models, normalizing flows, etc) are either very large, rendering it impractical for edge devices, or taking extremely long training time.

To learn the input distribution $\mathbb{P}_c(x)$ efficiently, we resort to Gaussian mixture models (GMM); for the conditional distribution $\mathbb{P}_c(y|x)$, we follow the same idea as Marfoq et al. (2021), to use light-weighted, parameterized supervised learning models.

3.2. Models

Formally, we define our model as:

- All clients share the GMM parameters $\{\boldsymbol{\mu}_{m_1}, \boldsymbol{\Sigma}_{m_1}\}$ for any $m_1 \in [M_1]$.
- All clients share the supervised learning parameters $\boldsymbol{\theta}_{m_2}$ for $m_2 \in [M_2]$.
- Each client c keeps its own personalized learner weights $\pi_c(m_1,m_2)$, which satisfies $\sum_{m_1,m_2} \pi_c(m_1,m_2) = 1$.

Note that M_1 is the number of Gaussian components, and M_2 is the number of learners. Under our definition of the models above, for client c, its hypothesis is defined as:

$$h_c(\mathbf{x}, \mathbf{y}) := \sum_{m_1, m_2} \pi_c(m_1, m_2) \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{m_1}, \boldsymbol{\Sigma}_{m_1}) P_{\boldsymbol{\theta}_{m_2}}(\mathbf{y}|\mathbf{x}),$$

where $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the probability density of multi-variate Gaussian distribution[1], and $P_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ is some supervised-learning model parameterized by $\boldsymbol{\theta}$.

Under this formulation, our optimization target becomes $\forall c \in \mathcal{C}$ (we omit M_1 or M_2 when clear):

$$\max_{\pi_c, \boldsymbol{\theta}} \mathop{\mathbb{E}}_{(\mathbf{x}, \mathbf{y}) \sim \mathbb{P}_c} \Big[\log \Big(\sum_{m_1, m_2} \pi_c(m_1, m_2) \mathcal{N}(x; \boldsymbol{\mu}_{m_1}, \boldsymbol{\Sigma}_{m_1}) P_{\boldsymbol{\theta}_{m_2}}(\mathbf{y}|\mathbf{x}) \Big) \Big].$$

3.3. The Centralized EM Algorithm

To reduce notation clutter, we use $m = (m_1, m_2)$ and $\boldsymbol{\Theta}_m = (\boldsymbol{\mu}_{m_1}, \boldsymbol{\Sigma}_{m_1}, \boldsymbol{\theta}_{m_2})$. We denote our model as $P_{\pi_c, \boldsymbol{\Theta}}(\mathbf{x}, \mathbf{y}) = \sum_m \pi_c(m) P_{\boldsymbol{\Theta}_m}(\mathbf{x}, \mathbf{y})$. Under this simplified notation, we can derive the EM algorithm as follows. Here we first provide a brief derivation of the centralized EM algorithm. Later on, we will extend it to the client-server EM algorithm in a federated setting.

Denote $q_{\mathbf{s}}(\cdot)$ as a probability distribution over $[M]$, where $\mathbf{s} = (\mathbf{x}, \mathbf{y})$. Also, for each sample, we assume it is drawn by first sampling the latent random variable $z \sim \pi_c(\cdot)$ and then sampling $(\mathbf{x}, \mathbf{y}) \sim P_{\boldsymbol{\Theta}_z}(\mathbf{x}, \mathbf{y})$.

To derive the centralized EM algorithm, we can establish the following lower bound of the likelihood for a sample $(\mathbf{x}, \mathbf{y})$:

$$\begin{aligned}
&\log \left(P_{\pi_c, \boldsymbol{\Theta}}(\mathbf{x}, \mathbf{y}) \right) \\
&\geq \sum_{m \in [M]} q_{\mathbf{s}}(m) \log \Big(\frac{P_{\pi_c, \boldsymbol{\Theta}}(z = m, \mathbf{x}, \mathbf{y})}{q_{\mathbf{s}}(m)} \Big) \\
&= \sum_{m \in [M]} q_{\mathbf{s}}(m) \Big[\log \Big(\frac{P_{\pi_c, \boldsymbol{\Theta}}(z = m)}{q_{\mathbf{s}}(m)} \Big) + \log \left(P_{\pi_c, \boldsymbol{\Theta}}(\mathbf{x}, \mathbf{y} | z = m) \right) \Big] \\
&= \sum_{m \in [M]} q_{\mathbf{s}}(m) \Big[\log \Big(\frac{\pi_c(m)}{q_{\mathbf{s}}(m)} \Big) + \log \left(P_{\boldsymbol{\Theta}_m}(\mathbf{x}, \mathbf{y}) \right) \Big] \qquad (4) \\
&= \sum_{m \in [M]} q_{\mathbf{s}}(m) \Big[\log \Big(\frac{P_{\pi_c, \boldsymbol{\Theta}}(z = m | \mathbf{x}, \mathbf{y})}{q_{\mathbf{s}}(m)} \Big) + \log \left(\mathbb{P}_{\pi_c, \boldsymbol{\Theta}}(\mathbf{x}, \mathbf{y}) \right) \Big], \qquad (5)
\end{aligned}$$

where the first inequality is due to Jensen's inequality. Equation (4) comes from the first equation (the line directly above (4)); Equation (5) comes from the same line by decomposing $P_{\pi_c, \boldsymbol{\Theta}}(z = m)$ into the conditional probability.

The EM algorithm will try to maximize Equation (4) and (5) alternatively, to ensure the lower bound of the likelihood (also called evidence lower bound) is maximized. This leads to the following update form:

- **E-Step:** Fix π_c and $\boldsymbol{\Theta}$, maximize Equation (5) via $q_{\mathbf{s}}(m)$ for each $\mathbf{s} = (\mathbf{x}, \mathbf{y})$, we see the optimal solution will be

$$\begin{aligned}
q_{\mathbf{s}}(m) &= P_{\pi_c, \boldsymbol{\Theta}}(z = m | \mathbf{x}, \mathbf{y}) \\
&\propto \mathbb{P}_{\pi_c, \boldsymbol{\Theta}}(z = m, \mathbf{x}, \mathbf{y}) = \pi_c(m) \mathbb{P}_{\boldsymbol{\Theta}_m}(\mathbf{x}, \mathbf{y}).
\end{aligned}$$

- **M-Step:** Fix $q_{\mathbf{s}}(\cdot|\mathbf{x}, \mathbf{y})$, maximize Equation (4) via π_c and $\boldsymbol{\Theta}$, we see the optimal solution will be

$$\begin{aligned}
\pi_c(m) &= \frac{1}{N_c} \sum_{i=1}^{N_c} q_{\mathbf{s}_i}(m), \\
\boldsymbol{\Theta}_m &= \arg\max_{\boldsymbol{\Theta}} \sum_{i=1}^{N_c} q_{\mathbf{s}_i}(m) \log(\mathbb{P}_{\boldsymbol{\Theta}}(\mathbf{x}_i, \mathbf{y}_i)).
\end{aligned}$$

Now we substitute $m = (m_1, m_2)$ and $\boldsymbol{\Theta}_m = \{\boldsymbol{\mu}_{m_1}, \boldsymbol{\Sigma}_{m_1}, \boldsymbol{\theta}_{m_2}\}$. We can index the base component $\mathbb{P}_{m_1, m_2}(\mathbf{x}, \mathbf{y}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{m_1}, \boldsymbol{\Sigma}_{m_1}) \cdot \mathbb{P}_{\boldsymbol{\theta}_{m_2}}(\mathbf{y}|\mathbf{x})$. Substituting the specific model into the EM update rules proposed before, we can write the update rule at step t as:

- **E-Step:** For each client $c \in [C]$, for each $i \in [N_c]$,

$$\begin{aligned}
q^{(t)}_{\mathbf{s}_{c,i}}(m_1, m_2) \propto \pi_c^{(t-1)}(m_1, m_2) &\mathcal{N}(\mathbf{x}_{c,i}; \boldsymbol{\mu}_{m_1}^{(t-1)}, \boldsymbol{\Sigma}_{m_1}^{(t-1)}) \\
&\cdot \mathbb{P}_{\boldsymbol{\theta}_{m_2}^{(t-1)}}(\mathbf{y}_{c,i}|\mathbf{x}_{c,i}). \qquad \text{(E)}
\end{aligned}$$

- **M-Step:** For each client $c \in [C]$, $m_1 \in [M_1]$, $m_2 \in [M_2]$,

$$\begin{aligned}
\pi_c^{(t)}(m_1, m_2) &= \frac{1}{N_c} \sum_{i=1}^{N_c} q^{(t)}_{\mathbf{s}_{c,i}}(m_1, m_2), \qquad \text{(M)} \\
\boldsymbol{\mu}^{(t)}_{m_1, c} &= \frac{\sum_{i=1}^{N_c} \sum_{m_2} q^{(t)}_{\mathbf{s}_{c,i}}(m_1, m_2) \mathbf{x}_{c,i}}{\sum_{i=1}^{N_c} \sum_{m_2} q^{(t)}_{\mathbf{s}_{c,i}}(m_1, m_2)}, \\
\boldsymbol{\Sigma}^{(t)}_{m_1} &= \frac{\sum_{i=1}^{N_c} \sum_{m_2} q^{(t)}_{\mathbf{s}_{c,i}}(m_1, m_2) (\mathbf{x}_{c,i} - \boldsymbol{\mu}^{(t)}_{m_1, c})(\mathbf{x}_{c,i} - \boldsymbol{\mu}^{(t)}_{m_1, c})^\top}{\sum_{i=1}^{N_c} \sum_{m_2} q^{(t)}_{\mathbf{s}_{c,i}}(m_1, m_2)}, \\
\boldsymbol{\theta}^{(t)}_{m_2, c} &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N_c} \sum_{m_1} q^{(t)}_{\mathbf{s}_{c,i}}(m_1, m_2) \log(\mathbb{P}_{\boldsymbol{\theta}}(\mathbf{y}_i|\mathbf{x}_i)).
\end{aligned}$$

The update rule for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ in the M-step is obtained by explicitly solving the optimization problem. Notice that for $\boldsymbol{\theta}_{m_2, c}$, the maximization objective is equivalent to the (weighted) cross-entropy loss for classification.

3.4. The Client-Server EM Algorithm

Federated learning restricts that each client can only access their own data. In this section, we describe how to extend the centralized EM algorithm to the federated client-server setting. Equation (E) and (M) describes how the client should maintain their personalized weights $\pi_c^{(t)}$, their own estimation of the shared GMM bases $(\boldsymbol{\mu}^{(t)}_{m_1, c}, \boldsymbol{\Sigma}^{(t)}_{m_1, c})$ and the base learners $\boldsymbol{\theta}^{(t)}_{m_2, c}$. When a central server is present,

[1] The probability density of multi-variate Gaussian is defined as: $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) := \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\big(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \big)$.

each client shall send their own parameters to the server and the server will aggregate the parameters and broadcast the aggregated parameter back to all clients. The detailed federated algorithm 1 is included in Appendix A.

More specifically, at each round, (1) the central server broadcasts the aggregated base models to all clients; (2) each client locally updates the parameter of the base models and the mixture weights according to Equation (E) and (M); (3) the clients send the updated components $(\boldsymbol{\mu}_{m_1,c}^{(t)}, \boldsymbol{\Sigma}_{m_1,c}^{(t)}), \boldsymbol{\theta}_{m_2,c}^{(t)}$ and the summed response $\gamma_c^{(t)}(m_1, m_2) = \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2)$ back to the server; 4) the server aggregates the updates as follows:

$$\boldsymbol{\mu}_{m_1}^{(t)} = \frac{\sum_{c\in[C]}\sum_{m_2\in[M_2]}\gamma_c^{(t)}(m_1,m_2)\boldsymbol{\mu}_{m_1,c}^{(t)}}{\sum_{c\in[C]}\sum_{m_2\in[M_2]}\gamma_c^{(t)}(m_1,m_2)},$$

$$\boldsymbol{\Sigma}_{m_1}^{(t)} = \frac{\sum_{c\in[C]}\sum_{m_2\in[M_2]}\gamma_c^{(t)}(m_1,m_2)\boldsymbol{\Sigma}_{m_1,c}^{(t)}}{\sum_{c\in[C]}\sum_{m_2\in[M_2]}\gamma_c^{(t)}(m_1,m_2)},$$

$$\boldsymbol{\theta}_{m_2}^{(t)} = \frac{\sum_{c\in[C]}\sum_{m_1\in[M_1]}\gamma_c^{(t)}(m_1,m_2)\boldsymbol{\theta}_{m_2,c}^{(t)}}{\sum_{c\in[C]}\sum_{m_1\in[M_1]}\gamma_c^{(t)}(m_1,m_2)}.$$

3.5. Theoretical Guarantees

Since most federated learning algorithms are gradient-based, their convergence analyses usually assume the gradients of different clients are similar. For small steps of updates, the averaged updated parameters can still enjoy a decrease in the training loss. This is not the case for our GMM updates, because the M-step uses the closed-form solution for each client and then aggregates them, which means the widely-adopted gradient-similarity assumption will not help.

What we present in the following is an analysis of purely federated Gaussian Mixture Models. The convergence guarantee for the gradient-updated parameter $\boldsymbol{\theta}$ will have identical assumptions and proof as in Marfoq et al. (2021). We choose to omit the convergence result for $\boldsymbol{\theta}$. When leaving $\boldsymbol{\theta}$ out, we obtain a pure unsupervised likelihood maximization algorithm 2 in Appendix A. The centralized version of it is exactly the classical EM algorithm for GMM. The federated learning process for the Gaussian mixture is a novel federated unsupervised learning approach, which may be of independent interest.

To show the convergence of the proposed client-server EM algorithm, we consider the case that $\boldsymbol{\Sigma}_m$ is fixed to $\mathbf{I}$, and only $\boldsymbol{\mu}$ is updated and aggregated. This assumption is widely adopted in previous works regarding the convergence of EM algorithms for GMM. It is also well known that if the covariance matrix $\boldsymbol{\Sigma}_m$ is not restricted, GMM can assign one component $\mathcal{N}(\cdot; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ to one single data point $\mathbf{x}$ such that $\boldsymbol{\mu}_m = \mathbf{x}$ and $\boldsymbol{\Sigma}_m \to \mathbf{0}$, so that the likelihood goes to positive infinity. Assuming $\boldsymbol{\Sigma}_m = \mathbf{I}$ prevents this kind of unwanted divergence.

Theorem 1. Denote $F(\mu_{1:M}, \pi_{1:C})$ as the log-likelihood function, then we have

$$\frac{1}{T}\sum_{t=1}^{T}|F(\mu_{1:M}^{(t)}, \pi_{1:C}^{(t)}) - F(\mu_{1:M}^{(t-1)}, \pi_{1:C}^{(t-1)})| = O(T^{-1}).$$

Theorem 1 implies that the log-likelihood will finally converge to a maximum. The idea of the proof (details included in Appendix B) relies on the use of first-order surrogates of F to establish that each M-step will always increase the log-likelihood.

4. Experiments

4.1. Datasets

Synthetic dataset. The synthetic dataset can be seen as a d-dimensional extension of Figure 1. More specifically, assume there are M Gaussian components $\mathbb{P}^{(m)}(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_m, \mathbf{I}_d)$, with a corresponding labeling function $F^{(m)}(\mathbf{x}) = \mathbb{1}\{(\mathbf{x}-\boldsymbol{\mu}_m)^\top \mathbf{v}_m > 0\}$, where $\boldsymbol{\mu}_m$ and $\mathbf{v}_m$ are specified beforehand. For each client c, the data generation is as follows: 1). sample π_c from the Dirichlet distribution $\mathrm{Dir}(\alpha)$ with $\alpha = 0.4$ to serve as the heterogeneous mixture weight; 2). for each sample $i \in [N_c]$, first generate $z_i \sim \pi_c(\cdot)$; 3). then draw $\mathbf{x}_i \sim \mathbb{P}^{(z_i)}(\mathbf{x}_i) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{z_i}, \mathbf{I}_d)$ and $y_i = F^{(z_i)}(\mathbf{x}_i)$. For the experiments, we set $M = 3$ and $d = 32$. We generate $C = 300$ clients and each client has around $N_c = 3000$ samples.

Real datasets. We also use three federated benchmark datasets spanning different machine learning tasks to evaluate the proposed approach: image classification on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), handwriting character recognition on FEMNIST (Caldas et al., 2018a). We preprocessed all the datasets in the same manner as previously in (Marfoq et al., 2021) to build the testbed. To simulate the joint distribution heterogeneity, we sample 50% of image data (denoted as $\mathcal{D}_2$, $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$) to perform a two-step approach for prepossessing image data: 1) we simulate heterogeneity of $\mathbb{P}_c(\mathbf{x})$ by transforming sampled images with 90-degree rotation, horizontal flip and inverse (Shorten & Khoshgoftaar, 2019) (denoted as $T(\cdot)$); 2) we introduce heterogeneity in $\mathbb{P}_c(\mathbf{y}|\mathbf{x})$ by applying a randomly generated permutation (denoted as P_A) to the labels of the transformed image data. Formally, the new dataset, denoted as $\widehat{\mathcal{D}}$, is defined as follows: $\widehat{\mathcal{D}} = \mathcal{D}_1 \cup \{(T(\mathbf{x}), P_A(\mathbf{y}))|(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_2\}$. In this way, we can obtain data from different joint distributions. We create the federated setting of CIFAR-10 by distributing samples with the same label across the clients according to a symmetric Dirichlet distribution with parameter 0.4, as in (Marfoq et al., 2021). CIFAR-100 data are distributed following (Marfoq et al., 2021). For all tasks, we randomly split each local dataset into training (60%), valida-

tion (20%), and test (20%) sets. In Table 1, we summarize the datasets, tasks, number of clients, the total number of samples, and backbone discriminative architectures.

4.2. Baseline Methods

To demonstrate the efficiency of our method, we compare the proposed FedGMM with the following baselines:

- Local: a personalized model trained only on the local dataset at each client;
- FedAvg (McMahan et al., 2017): a generic FL method that trains a unique global model for all clients;
- FedProx (Li et al., 2020): a re-parametrization of FedAvg to tackle statistical heterogeneity in FL;
- FedAvg+ (Jiang et al., 2019): a modification of FedAvg with two stages of training and local tuning;
- Clustered FL (Sattler et al., 2020): a framework exploiting geometric properties of the FL loss surface which groups the client population into clusters using conditional distributions;
- pFedMe (T Dinh et al., 2020): a bi-level optimization PFL that decouples the optimization of personalized models from learning the global model;
- FedEM (Marfoq et al., 2021): a federated multi-task learning approach assuming that local data distributions are mixtures of underlying distributions.

4.3. Implementation Details

To properly initialize each base component of the GMM, we employ a Resnet18 (He et al., 2016) encoder that has been pre-trained on the ImageNet dataset to encode input images and generate embeddings of dimension 512. Recognizing that high dimensionality can lead to increased computational complexity and reduced effectiveness of GMM, we utilize PCA (Jolliffe, 1986) to project the encoded embeddings into a lower-dimensional space of 48. For the sake of fairness in comparison, it is important to note that the Resnet18 encoder and PCA are exclusively employed for preprocessing inputs of the GMM component, while the inputs for the supervised backbone are raw images.

For each method, we follow (Marfoq et al., 2021) to tune the learning rate via grid search. In our experiments, the number of local epochs of each method is set to 1, the total communication round is set to 200, and the batch size is set to 128, as in (Marfoq et al., 2021). For a fair comparison, we adopt the same supervised backbone architecture for all baselines. More implementation[2] details are included in Appendix C.1.

4.4. Classification

The results are shown in Table 2. The evolution of average test accuracy over time for each experiment is shown in the Appendix. From the table, we observe that FedAvg surpasses Local, which indicates that federated training improves performance because of taking advantage of knowledge from other clients. However, personalized methods such as FedAvg+, ClusteredFL, and pFedMe perform worse than FedAvg because they only locally adjust the global model on each client. This strategy is not sufficient to capture the diversity of the joint distribution and cannot handle sample-specific personalization when samples come from different marginal distributions have varying labeling functions. ClusteredFL also fails to outperform FedAvg on all datasets, highlighting the importance of knowledge sharing between clusters for training good personalized models. FedEM, on the other hand, performs better than other PFL baselines on most datasets by effectively modeling the heterogeneity of conditional distributions. As shown in the table, FedGMM outperforms all baselines, achieving 26.1% and 9.8% improvement on CIFAR-100 and Synthetic dataset respectively compared to the leading baselines. This is a result of its ability to construct personalized models based on the joint data distribution, effectively capturing the heterogeneity of each sample across different clients.

Besides, to see how the simulation results would change if we deviate from Gaussian assumptions, we conducted the following synthetic experiments. We use two settings to conduct the comparison. Setting 1 considers non-Gaussian input distribution. Setting 2 is also a synthetic setting, where some of the clients completely differ from others. Specifically, Setting 1 is the same as our Gaussian synthetic setting, but the data-generating distribution is different. Here, we adopt two different distributions, i.e., Laplace and Beta distributions. Other distributes would be similar. First, we generate 3 d-dimensional ($d = 32$) components based on the selected distribution type. Each component is determined either by the mean vector μ for Laplace distribution or the vectors α and β for Beta distribution. Then, we generate data from these components using multivariate distribution. We use Dirichlet distribution to distribute data to each client. Totally, we have 30 clients. For Setting 2, some clients sampled data from Gaussian, the others from a different distribution (i.e., Laplace or Beta distribution). Similarly, we also use 30 clients for simulation. The first 20 clients' data are sampled from Gaussian, and the data of the last 10 clients are sampled from selected distribution, i.e., Laplace or Beta distribution. We use Dirichlet distribution to distribute data to each client. The results are summarized in Table. 3. From the table, we can observe that under both settings, our method can still perform well since our

[2]https://github.com/zshuai8/FedGMM_ICML2023

Table 1. Datasets and models.

Dataset	Task	Number of clients	Number of samples	Backbone Supervised Model
Synthetic	Binary Classification	300	$\sim 1,000,000$	Linear sigmoid function
CIFAR-10	Image classification	80	60,000	MobileNet-v2
CIFAR-100	Image classification	100	60,000	MobileNet-v2
FEMNIST	Handwritten character recognition	539	120,772	2-layer CNN + 2-layer FFN

Table 2. Average test accuracy (%) across clients.

Dataset	Local	FedAvg	FedProx	FedAvg+	ClusteredFL	pFedMe	FedEM	FedGMM(Ours)
Synthetic	57.52	53.21	52.70	53.41	53.12	53.91	65.61	**72.02**
CIFAR10	19.96	45.53	37.0	34.33	38.81	23.51	49.12	**52.96**
CIFAR100	13.36	17.71	7.95	11.51	12.46	9.92	17.28	**22.33**
FEMNIST	62.39	75.08	32.84	57.99	75.04	39.45	75.56	**79.49**

model considers the cluster and mixture structure of the data distribution.

Table 3. Effectiveness on non-Gaussian distribution data (accuracy (%)).

	Setting 1		Setting 2	
	Beta	Laplace	Beta/partial	Laplace/partial
FedGMM(Ours)	**72.12**	**89.06**	**80.54**	**84.79**
FedEM	71.77	83.94	74.22	81.79
FedAVG	56.24	82.45	56.13	70.15
FedAVG+Local	56.6	82.53	57.7	70.36
fedProx	55.64	75.64	55.9	71.16
ClusteredFL	56.23	82.45	56.1	70.14
Local	58.46	83.68	67.18	74.69

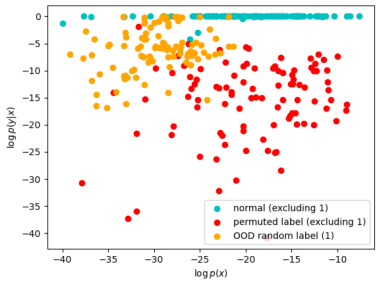


Figure 2. FedGMM to detect marginal distribution and conditional distribution outliers.

4.5. Novel Sample Detection

In our algorithm, the server meticulously maintains comprehensive, global statistics of all data points within the federated learning ecosystem, such as the GMM parameters[3] and the supervised learning components. Thus, for a new sample, the learned model is able to quickly infer its marginal distribution [4], conditional distribution (Eq. 3) and the joint distribution (Eq. 1). As such, a by-product of the model is that it can be used to detect out-of-distribution samples. We begin by using a typical leave-one-out method for out-of-distribution detection to demonstrate the effectiveness of our model in identifying various types of outliers. Specifically, we train our model using the MNIST dataset, with 50 clients each contributing 500 sampled images. In the training, we exclude images of number 1 and test on normal samples together with two types of outliers. The first category of outliers consists of images from the same marginal distribution $\mathbb{P}(\mathbf{x})$, namely $\{0, 2, 3,..., 9\}$, but their labels have been altered by applying a random permutation. The second category of outliers are images of digit 1 that are not present in the training data. We plot all the sample points with respect to their $\log \mathbb{P}(\mathbf{x})$ and $\log \mathbb{P}(\mathbf{y}|\mathbf{x})$ values inferred by our model in Figure 2. Here, the dots in cyan color are the normal ones. The orange points denote unseen input '1', and the red dots are outliers with the same marginal distribution but altered labels. We can observe that by modeling the conditional probability, the y-axis can separate red dots from the normal ones. Our density estimation model can separate the second type of outlier from other numbers as well.

To evaluate the performance of our OOD detection approach quantitatively, we trained each model using the following settings: we construct a federated setting using MNIST data, similar to the one described in Sec. 4.1. Details are included in Appendix C.3. Basically, we create two sets of test samples drawn from the training distribution. The first set (as in-domain) remains unchanged. As the second (out-of-domain) set, we simulate the heterogeneity of $\mathbb{P}_c(\mathbf{x})$ by transforming sampled images with a scale factor of 0.5, 90-degree rotation, and horizontal flip (Shorten & Khoshgoftaar, 2019). With the test samples, we want to investigate if a model can distinguish between known and novel samples.

For comparison purposes, since none of the baselines are able to detect novel samples, we adapt them as follows. Similar to the idea in (Liu et al., 2020), we use the prediction output logits with softmax to represent the classifier's con-

[3]We can aggregate the global parameter $\pi = \sum_{c\in[C]} \frac{1}{N_c}\gamma_c$.

[4]The marginal distribution can be calculated by $\mathbb{P}(\mathbf{x}) = \sum_{m_2\in[M2]}\sum_{m_1\in[M1]} \pi(m_1, m_2)\cdot\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{m_1},\boldsymbol{\Sigma}_{m_1})$.

Table 4. Comparison between FedGMM and the applicable baselines on novel sample/client detection.

Model	AUROC	AP	Max-F1
Local	50.74	60.14	66.67
FedAvg	66.55	68.05	66.67
FedProx	75.23	76.24	71.90
FedAvg+	66.65	68.09	66.67
ClusteredFL	50.74	60.14	66.67
pFedMe	73.32	77.91	68.30
FedEM	86.04	90.02	80.25
FedGMM	**99.21**	**99.60**	**99.49**

fidence in different categories. The highest value among different categories is treated as the in-domain likelihood. This means the sharper the sample's prediction distribution, the more certain the classifier is that the sample is in-domain. Since the personalized baseline approaches do not have a global model, we selected the highest confidence value among different clients for a given new sample. It's worth noting that we did not include the Bayesian method in (Kotelevskii et al., 2022) as the baseline because the method can only perform novel detection at the client level, whereas here, we are conducting it at the sample level. Following (Cheng & Vasconcelos, 2021; Vaze et al., 2022; Sharma et al., 2021), we report Area Under ROC (AUROC), Average Precision (AP), and Max-F1 for evaluation.

Table 4 summarizes the results. We observe that FedGMM outperforms all baseline's overall evaluation metrics, indicating the superiority of our model in modeling joint distribution. Our approach models each sample with a mixture distribution of different components, as described in Sec. 3, which fits the mixture data well hence allowing to detect novel samples that are close to the boundary. Similar to (Liu et al., 2020), in Figure 6 in Appendix C.3, we visualize the normalized likelihood histogram of known and novel samples for FedGMM, FedEM, and FedAvg. The figures indicate the likelihoods of FedGMM are more distinguishable for known and novel samples than for the baselines.

4.6. Generalization to Unseen Clients

As previously discussed, FedGMM is flexible, enabling easy inference of new clients who did not participate in the training phase. This is accomplished by learning their personalized mixture weights. Specifically, we only need to update q, π, and γ in lines 6, 8, and 10 of Algorithm 1 in Appendix A. All other parameters remain fixed during the update process. This adaptation incurs minimal computational costs.

To validate the effectiveness of our approach for generalization to unseen client data, we use the data with the same training setting as in the previous classification task (refer to Sec. 4.4). We use 80% of clients to train the model and 20% to test for unseen data adaptation, as per the setting in (Marfoq et al., 2021). We split samples into 50% for adaptation and 50% testing and adapt the mixture weights in our approach and the mixture weights of conditional distributions in FedEM using the adaptation samples from unseen clients. Aside from FedAvg+ and FedEM, it is uncertain how the other PFL algorithms can be adapted to unseen clients. As FedAvg has a global model, we can still use it for test on the new data. As shown in Table 5, our approach obtains a minimal decrease in accuracy, as it has the ability to adapt to new joint distributions, whereas FedEM only adapts to conditional distributions. Our approach and FedEM both surpass FedAvg+ as it is unable to adapt to new data distributions, leading to subpar performance when there is a change in the distribution. Our approach's ability to model the joint distribution with a mixture model allows for easy generalization to unseen client data, making it a practical and effective solution in cases of client drift. More results are included in Appendix C.4.

Table 5. Average test accuracy of new clients unseen at training.

Model	FedAvg	FedAvg+	FedEM	FedGMM
FEMNIST	74.50	51.00	72.00	**78.51**
CIFAR10	44.51	32.25	47.51	**50.25**
CIFAR100	11.50	7.75	16.50	**21.25**

4.7. Parameter Sensitivity

We also analyzed the hyperparameters of FedGMM in this section. Basically, FedGMM only has two hyper-parameters, i.e., M_1 and M_2. Different choices of the number of mixture components do not significantly impact the model's classification performance. However, the clustering quality may vary depending on the number of components used. We present the accuracy with respect to the number of GMM cluster components and supervised learning model components in Figure 3. The figure shows that our algorithm is not very sensitive to hyperparameters and that selecting a component number close to the ground-truth component number of the distribution can improve the clustering quality and boost the classification performance. In our setting, we have two ground-truth clusters, and labeling functions, thus the setting of M_1=2 and M_2=2 gets the best performance.

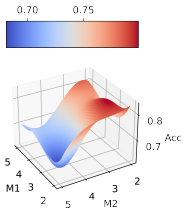


Figure 3. Parameter sensitivity analysis with respect to the number of GMM clusters, number of classifiers, and performance.

5. Additional Related Work

There has been significant advancement in the creation of new techniques to address various FL difficulties in recent years (Wang et al., 2020; Kairouz et al., 2021; Li et al., 2020; Yu et al., 2022). Research in this field focuses on how to do model aggregation, how to achieve personalization (Achituve et al., 2021; Chen et al., 2022), how to attack/defense the federated learning system (Lam et al., 2021), and efficiency aspects including communication efficiency (Liu et al., 2021; Amiri et al., 2020; Shahid et al., 2020; Hou et al., 2022; Hyeon-Woo et al., 2022), hardware efficiency (Cheng et al., 2021) and algorithm efficiency (Balakrishnan et al., 2022; Xu et al., 2022). In this section, we focus on reviewing two groups of works: personalized federated learning and federated uncertainty quantification.

5.1. Personalized Federated Learning

However, in real settings, there always exists statistical heterogeneity across clients (Kairouz et al., 2021; Li et al., 2020; Sattler et al., 2019). There are many efforts on extending the FL methods for heterogeneous clients to achieve personalization (Achituve et al., 2021; Chen et al., 2022; T Dinh et al., 2020; Tan et al., 2022; Fallah et al., 2020; Deng et al., 2020; Hong et al., 2022; Jeong & Hwang, 2022), adopting meta-learning, client clustering, multi-task learning, model interpolation, knowledge distillation, and lottery ticket hypothesis. For example, several works train a global model as a meta-model and then fine-tune the parameters for each client (Sim et al., 2019; Jiang et al., 2019), which still have difficulty for generalization (Caldas et al., 2018a; Marfoq et al., 2021). Clients can be assigned to many clusters, and clients in the same cluster are assumed to use the same model via clustered FL techniques (Ghosh et al., 2020; Shlezinger et al., 2020; Sattler et al., 2020). As a result, the federated model will not be ideal because clients from various clusters would not share pertinent information. Another group of approaches uses multi-task learning to learn customized models in the FL environment (Smith et al., 2017; Vanhaesebrouck et al., 2017; Caldas et al., 2018b), enabling more complex relationships between clients' models. They did not, however, take into account the diverse statistical diversity. The study in (Marfoq et al., 2021) takes into account conditional client distribution but makes the assumption that their marginal distributions are stable. Our method, however, models the diversity of joint distributions among clients. For each client, some works attempt to jointly train a global model and a local model, but they may fail if some local distributions deviate significantly from the average distribution. (Corinzia et al., 2019; Deng et al., 2020). (Shamsian et al., 2021) proposed to carry out personalization in federated learning via a hypernetwork. Similar to this, Dai et al. suggested using decentralized sparse training to generate PFL that is effective at communication (Dai et al., 2022). Some researchers addressed the heterogeneity by adopting knowledge distillation (Zhu et al., 2021; Chen & Chao, 2021; Lin et al., 2020).

5.2. Uncertainty Quantification and OOD Detection for Personalized Federated Learning

In the context of federated learning, when *client drift* happens, i.e., the distribution of the data on different devices becomes increasingly dissimilar over time, it is desirable to detect novel clients or instances that are out-of-distribution. However, because it calls for unsupervised density estimation, this topic has not received much attention in the literature. Unsupervised federated clustering (Lubana et al., 2022) or representation learning (Zhuang et al., 2022) techniques have been described in several publications. However, these techniques cannot be used to directly estimate the joint distribution of instances, and it is difficult to perform OOD detection tasks with them. To address the issue, some researchers proposed a Bayesian approach to PFL. For example, FedPop (Kotelevskii et al., 2022) is the first personalized FL approach that allows uncertainty quantification. Using an empirical Bayes prediction approach, FedPop enables personalization and on-device uncertainty measurement. FedPop, however, is unable to simulate the joint mixed distribution, which prevents it from addressing the joint distribution heterogeneity issue. Additionally, it is unable to carry out sample-wise uncertainty quantification.

6. Conclusion

In this paper, we address the challenge of *joint distribution heterogeneity* in Personalized Federated Learning (PFL). Existing PFL methods mainly focus on modeling concept shift, which results in suboptimal performance when joint data distributions across clients diverge. These methods also fail to effectively address the problem of client drift, making it difficult to detect new samples and adapt to unseen client data. To tackle these issues, we propose a novel approach called FedGMM, which uses Gaussian mixture models to fit the joint data distributions across FL devices. This approach effectively addresses the problem and allows for uncertainty quantification, making it easy to recognize new clients and samples. Furthermore, we present a federated Expectation-Maximization (EM) algorithm for learning model parameters, which is theoretically guaranteed to converge. The results of our extensive experiments on three benchmark FL datasets and a synthetic dataset show that our proposed method outperforms state-of-the-art baselines.

References

Achituve, I., Shamsian, A., Navon, A., Chechik, G., and Fetaya, E. Personalized federated learning with gaussian processes. In *NeurIPS*, 2021.

Aledhari, M., Razzak, R., Parizi, R. M., and Saeed, F. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020.

Amiri, M. M., Gunduz, D., Kulkarni, S. R., and Poor, H. V. Federated learning with quantized global model updates. 2020.

Balakrishnan, S., Li, T., Tianyi Zhou, N. H., Smith, V., and Bilmes, J. Diverse client selection for federated learning via submodular maximization. In *ICLR*, 2022.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018a.

Caldas, S., Smith, V., and Talwalkar, A. Federated kernelized multi-task learning. In *Proc. SysML Conf.*, pp. 1–3, 2018b.

Chen, H., Ding, J., Tramel, E., Wu, S., Sahu, A. K., Avestimehr, S., and Zhang, T. Self-aware personalized federated learning. In *NeurIPS*, 2022.

Chen, H.-Y. and Chao, W.-L. Fed{be}: Making bayesian model ensemble applicable to federated learning. In *ICLR*, 2021.

Cheng, J. and Vasconcelos, N. Learning deep classifiers consistent with fine-grained novelty detection. In *CVPR*, pp. 1664–1673, 2021.

Cheng, X., Lu, W., Huang, X., Hu, S., and Chen, K. Haflo: Gpu-based acceleration for federated logistic regression. 2021.

Corinzia, L., Beuret, A., and Buhmann, J. M. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

Dai, R., Shen, L., He, F., Tian, X., and Tao, D. Edispfl: Towards communication-efficient personalized federated learning via decentralized sparse training. In *ICML*, 2022.

Deng, Y., Kamani, M. M., and Mahdavi, M. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. An efficient framework for clustered federated learning. *NeurIPS*, 33:19586–19597, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, pp. 770–778, 2016.

Hong, J., Wang, H., Wang, Z., and Zhou, J. Efficient split-mix federated learning for on-demand and in-situ customization. In *ICLR*, 2022.

Hou, C., Thekumparampil, K. K., Fanti, G., and Oh, S. Fedchain: Chained algorithms for near-optimal communication cost in federated learning. In *ICLR*, 2022.

Hyeon-Woo, N., Ye-Bin, M., and Oh, T.-H. Fedpara: Low-rank hadamard product for communication-efficient federated learning. In *ICLR*, 2022.

Jeong, W. and Hwang, S. J. Factorized-fl: Personalized federated learning with parameter factorization and similarity matching. In *NeurIPS*, 2022.

Jiang, Y., Konečnỳ, J., Rush, K., and Kannan, S. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

Jolliffe, I. T. Principal component analysis. In *Principal Component Analysis*. Springer Verlag, New York, 1986.

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*, 14(1–2):1–210, 2021.

Kotelevskii, N. Y., Vono, M., Durmus, A., and Moulines, E. Fedpop: A bayesian approach for personalised federated learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *NeurIPS*, 2022.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Lam, M., Wei, G.-Y., Brooks, D., Reddi, V. J., and Mitzenmacher, M. Gradient disaggregation: Breaking privacy in federated learning by reconstructing the user participant matrix. 2021.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Lim, W. Y. B., Luong, N. C., Hoang, D. T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., and Miao, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3): 2031–2063, 2020.

Lin, T., Kong, L., Stich, S. U., and Jaggi, M. Ensemble distillation for robust model fusion in federated learning. In *NeurIPS*, pp. 2351–2363, 2020.

Liu, L., Zhang, J., Song, S., , and Letaief, K. B. Hierarchical quantized federated learning: Convergence analysis and system design. 2021.

Liu, W., Wang, X., Owens, J., and Li, Y. Energy-based out-of-distribution detection. In *NeurIPS*, 2020.

Lubana, E. S., Tang, C. I., Kawsar, F., Dick, R., and Mathur, A. Orchestra: Unsupervised federated learning via globally consistent clustering. 2022.

Marcel, S. and Rodriguez, Y. Torchvision the machine-vision package of torch. In *ACM MM*, pp. 1485–1488, 2010.

Marfoq, O., Neglia, G., Bellet, A., Kameni, L., and Vidal, R. Federated multi-task learning under a mixture of distributions. *NeurIPS*, 34, 2021.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, pp. 1273–1282. PMLR, 2017.

Mothukuri, V., Parizi, R. M., Pouriyeh, S., Huang, Y., Dehghantanha, A., and Srivastava, G. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019.

Purushotham, S., Carvalho, W., Nilanon, T., and Liu, Y. Variational recurrent adversarial deep domain adaptation. In *ICLR*, 2017.

Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Robust and communication-efficient federated learning from non-iid data. *IEEE TNNLS*, 31(9):3400–3413, 2019.

Sattler, F., Müller, K.-R., and Samek, W. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE TNNLS*, 32(8): 3710–3722, 2020.

Shahid, O., Pouriyeh, S., Parizi, R. M., Sheng, Q. Z., Srivastava, G., and Zhao, L. Communication efficiency in federated learning: Achievements and challenges. 2020.

Shamsian, A., Navon, A., Fetaya, E., and Chechik, G. Personalized federated learning using hypernetworks. *ICML*, 2021.

Sharma, K., Zhang, Y., Ferrara, E., and Liu, Y. Identifying coordinated accounts on social media through hidden influence and group behaviours. In *SIGKDD*, pp. 1441–1451, 2021.

Shlezinger, N., Rini, S., and Eldar, Y. C. The communication-aware clustered federated learning problem. In *IEEE ISIT*, pp. 2610–2615. IEEE, 2020.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *J. Big Data*, 6:60, 2019.

Shukla, S. N. and Marlin, B. Interpolation-prediction networks for irregularly sampled time series. In *ICLR*, 2019.

Sim, K. C., Zadrazil, P., and Beaufays, F. An investigation into on-device personalization of end-to-end automatic speech recognition models. *arXiv preprint arXiv:1909.06678*, 2019.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. *NeurIPS*, 30, 2017.

T Dinh, C., Tran, N., and Nguyen, J. Personalized federated learning with moreau envelopes. *NeurIPS*, 33:21394–21405, 2020.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. Towards personalized federated learning. *IEEE TNNLS*, 2022.

Vanhaesebrouck, P., Bellet, A., and Tommasi, M. Decentralized collaborative learning of personalized models over networks. In *Artificial Intelligence and Statistics*, pp. 509–517. PMLR, 2017.

Vaze, S., Han, K., Vedaldi, A., and Zisserman, A. Open-set recognition: A good closed-set classifier is all you need. In *ICLR*, 2022.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. In *ICLR*, 2020.

Wang, K., Mathews, R., Kiddon, C., Eichner, H., Beaufays, F., and Ramage, D. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

Wang, T., Cheng, W., Luo, D., Yu, W., Ni, J., Tong, L., Chen, H., and Zhang, X. Personalized federated learning via heterogeneous modular networks. In *IEEE ICDM*, 2022.

Xu, C., Hong, Z., Huang, M., and Jiang, T. Acceleration of federated learning with alleviated forgetting in local training. In *ICLR*, 2022.

Yu, Y., Wei, A., Karimireddy, S. P., and Yi Ma, M. I. J. Federated learning with matched averaging. In *arXiv:2207.06343*, 2022.

Zhu, Z., Hong, J., and Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. In *ICML*, pp. 12878–12889, 2021.

Zhuang, W., Wen, Y., and Zhang, S. Divergence-aware federated self-supervised learning. In *ICLR*, 2022.

A. The Client-Server Training Algorithm.

In this section, we detail our algorithm FedGMM in Algorithm 1. Specifically, At each round, clients and server are communicated as follows.

- (1) the central server broadcasts the aggregated base models to all clients (line 2), including Gaussian parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and supervised learning models $(\boldsymbol{\theta})$;
- (2) each client locally updates the parameter of the base models and the mixture weights (line 3-9) according to Equation (E) and (M);
- (3) the clients send the updated components $(\boldsymbol{\mu}^{(t)}_{m_1,c}, \boldsymbol{\Sigma}^{(t)}_{m_1,c}), \boldsymbol{\theta}^{(t)}_{m_2,c}$ and the summed response $\gamma^{(t)}_c(m_1, m_2) = \sum_{i\in[N_c]} q^{(t)}_{\mathbf{s}_{c,i}}(m_1, m_2)$ back to the server (line 10);
- 4) the server aggregates the updates including Gaussian parameters and supervised component (line 12-17);

In Algorithm 2, we also provide a pure unsupervised federated (client-server) GMM algorithm. We will prove its convergence property of it in the next section. The federated learning process for the Gaussian mixture is a novel federated unsupervised learning approach, which may be of independent interest.

B. Proof of Theorem 1.

In this section, we provide theoretical proof for Theorem 1, that indicating the log-likelihood in our proposed federated EM algorithm will finally converge to a maximum. Before presenting the proof, we first define the surrogate function and present two lemmas regarding the monotonicity of the updates with respect to the surrogate function.

First, we lower bound the likelihood F with surrogate function G's as:

$$
\begin{aligned}
F(\boldsymbol{\mu}_{1:M}, \pi_{1:C}) &= \sum_{c=1}^{C}\sum_{i=1}^{N_c} \log\Big(\sum_{m=1}^{M} \pi_c^{(t)}(m)\mathcal{N}(\mathbf{x}_{c,i}; bmu_m^{(t)}, Ib)\Big) \\
&\geq \sum_{c=1}^{C}\sum_{i=1}^{N_c}\sum_{m=1}^{M} q^{(t)}_{\mathbf{s}_{c,i}}(m)\Big[\log\big(\pi_c(m)\big) + \log\big(\mathcal{N}(\mathbf{x}_{c,i}; \boldsymbol{\mu}_m, \mathbf{I})\big) - \log\big(q^{(t)}_{\mathbf{s}_{c,i}}(m)\big)\Big] \\
&= \sum_{c=1}^{C} \underbrace{\sum_{i=1}^{N_c}\sum_{m=1}^{M} q^{(t)}_{\mathbf{s}_{c,i}}(m)\Big[\log\big(\pi_c(m)\big) + \frac{d\log(2\pi)}{2} - \frac{1}{2}\|\mathbf{x}_{c,i} - \boldsymbol{\mu}_m\|_2^2 - \log\big(q^{(t)}_{\mathbf{s}_{c,i}}(m)\big)\Big]}_{G_c^{(t)}(\boldsymbol{\mu}_{1:M}, \pi_c)},
\end{aligned}
$$

where the first inequality is due to Jensen's inequality. In other words, we have for any time step $t > 0$

$$
F(\boldsymbol{\mu}_{1:M}, \pi_{1:C}) \geq G^{(t)}(\boldsymbol{\mu}_{1:M}, \pi_{1:C}) := \sum_{c=1}^{C} G_c^{(t)}(\boldsymbol{\mu}_{1:M}, \pi_c).
$$

The inequality becomes equality when $q^{(t)}_{\mathbf{s}_{c,i}}(m) \propto \pi_c(m)\mathcal{N}(\mathbf{x}_{c,i}; \boldsymbol{\mu}_m, \mathbf{I})$, that is, when the E-step is performed. Therefore, we have $F(\boldsymbol{\mu}^{(t-1)}_{1:M}, \pi^{(t-1)}_{1:C}) = G_c^{(t)}(\boldsymbol{\mu}^{(t-1)}_{1:M}, \pi^{(t-1)}_{1:C})$.

Algorithm 1 Algorithm of FedGMM

for $t = 1, 2, \dots$ **do**
  server broadcasts $\{\boldsymbol{\mu}_m^{(t-1)}, \boldsymbol{\Sigma}_m^{(t-1)}\}_{m\in[M_1]}, \{\boldsymbol{\theta}_m^{(t-1)}\}_{m\in[M_2]}$ to all clients
  for client $c \in [C]$ **do**
    for component $m_1 \in [M_1], m_2 \in [M_2]$ **do**
      for sample $\mathbf{s}_{c,i} = (\mathbf{x}_{c,i}, \mathbf{y}_{c,i})$, $i \in [N_c]$ **do**
        Set $q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2) \propto \pi_c^{(t-1)}(m_1, m_2) \cdot \mathcal{N}\big(\mathbf{x}_{c,i}; \boldsymbol{\mu}_{m_1}^{(t-1)}, \boldsymbol{\Sigma}_{m_1}^{(t-1)}\big) \cdot \exp\big(-L_{\text{CE}}(s_{c,i}; \boldsymbol{\theta}_{m_2}^{(t-1)})\big)$
      end for
      Set for all $m_1 \in [M_1], m_2 \in [M_2]$:

$$\pi_c^{(t)}(m_1, m_2) = \frac{1}{N_c} \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2)$$

$$\boldsymbol{\mu}_{m_1,c}^{(t)} = \frac{\sum_{i\in[N_c]} \sum_{m_2\in[M_2]} q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2)\mathbf{x}_{c,i}}{\sum_{i\in[N_c]} \sum_{m_2\in[M_2]} q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2)}$$

$$\boldsymbol{\Sigma}_{m_1,c}^{(t)} = \frac{\sum_{i\in[N_c]} \sum_{m_2\in[M_2]} q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2)(\mathbf{x}_{c,i} - \boldsymbol{\mu}_{m_1,c}^{(t)})(\mathbf{x}_{c,i} - \boldsymbol{\mu}_{m_1,c}^{(t)})^\top}{\sum_{i\in[N_c]} \sum_{m_2\in[M_2]} q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2)}$$

$$\boldsymbol{\theta}_{m_2,c}^{(t)} = \arg\min_{\boldsymbol{\theta}} \sum_{i\in[N_c]} \sum_{m_1\in[M_1]} q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2) L_{\text{CE}}(\mathbf{x}_{c,i}, \mathbf{y}_{c,i}; \boldsymbol{\theta})$$

    end for
    client c sends $\{\boldsymbol{\mu}_{m_1,c}^{(t)}, \boldsymbol{\Sigma}_{m_1,c}^{(t)}, \gamma_c^{(t)}(m_1, m_2) = \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m_1, m_2)\}$ to the server
  end for
  for Gaussian component $m_1 \in [M_1]$ **do**
    server aggregates

$$\boldsymbol{\mu}_{m_1}^{(t)} = \frac{\sum_{c\in[C]} \sum_{m_2\in[M_2]} \gamma_c^{(t)}(m_1, m_2)\boldsymbol{\mu}_{m_1,c}^{(t)}}{\sum_{c\in[C]} \sum_{m_2\in[M_2]} \gamma_c^{(t)}(m_1, m_2)}$$

$$\boldsymbol{\Sigma}_{m_1}^{(t)} = \frac{\sum_{c\in[C]} \sum_{m_2\in[M_2]} \gamma_c^{(t)}(m_1, m_2)\boldsymbol{\Sigma}_{m_1,c}^{(t)}}{\sum_{c\in[C]} \sum_{m_2\in[M_2]} \gamma_c^{(t)}(m_1, m_2)}$$

  end for
  for Supervised component $m_2 \in [M_2]$ **do**
    server aggregates

$$\boldsymbol{\theta}_{m_2}^{(t)} = \frac{\sum_{c\in[C]} \sum_{m_1\in[M_1]} \gamma_c^{(t)}(m_1, m_2)\boldsymbol{\theta}_{m_2,c}^{(t)}}{\sum_{c\in[C]} \sum_{m_1\in[M_1]} \gamma_c^{(t)}(m_1, m_2)}$$

  end for
end for

Algorithm 2 Federated GMM (Unsupervised)

for $t = 1, 2, \dots$ **do**
  server broadcasts $\{\boldsymbol{\mu}_m^{(t-1)}, \boldsymbol{\Sigma}_m^{(t-1)}\}_{m\in[M]}$ to all clients
  for client $c \in [C]$ **do**
    for component $m \in [M]$ **do**
      for sample $\mathbf{s}_{c,i} = (\mathbf{x}_{c,i}, \mathbf{y}_{c,i})$, $i \in [N_c]$ **do**
        Set $q_{\mathbf{s}_{c,i}}^{(t)}(m) \propto \pi_c^{(t-1)}(m) \cdot \mathcal{N}\big(\mathbf{x}_{c,i}; \boldsymbol{\mu}_m^{(t-1)}, \boldsymbol{\Sigma}_m^{(t-1)}\big)$
      end for
      Set for all $m \in [M]$:

$$\pi_c^{(t)}(m) = \frac{1}{N_c} \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)$$

$$\boldsymbol{\mu}_{m,c}^{(t)} = \frac{\sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)\mathbf{x}_{c,i}}{\sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)}$$

$$\boldsymbol{\Sigma}_{m,c}^{(t)} = \frac{\sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)(\mathbf{x}_{c,i} - \boldsymbol{\mu}_{m,c}^{(t)})(\mathbf{x}_{c,i} - \boldsymbol{\mu}_{m,c}^{(t)})^\top}{\sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)}$$

    end for
    client c sends $\{\boldsymbol{\mu}_{m_1,c}^{(t)}, \boldsymbol{\Sigma}_{m_1,c}^{(t)}, \gamma_c^{(t)}(m) = \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)\}$ to the server
  end for
  for Gaussian component $m \in [M]$ **do**
    server aggregates

$$\boldsymbol{\mu}_m^{(t)} = \frac{\sum_{c\in[C]} \gamma_c^{(t)}(m)\boldsymbol{\mu}_{m,c}^{(t)}}{\sum_{c\in[C]} \gamma_c^{(t)}(m)}$$

$$\boldsymbol{\Sigma}_m^{(t)} = \frac{\sum_{c\in[C]} \gamma_c^{(t)}(m)\boldsymbol{\Sigma}_{m,c}^{(t)}}{\sum_{c\in[C]} \gamma_c^{(t)}(m)}$$

  end for
end for

Lemma 2. At any time step t, $G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t-1)}) \geq G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t-1)}, \pi_{1:C}^{(t-1)})$.

Proof. Notice that

$$\begin{aligned}
\boldsymbol{\mu}_m^{(t)} &= \frac{\sum_{c\in[C]} \gamma_c^{(t)}(m)\boldsymbol{\mu}_{m,c}^{(t)}}{\sum_{c\in[C]} \gamma_c^{(t)}(m)} \\
&= \frac{\sum_{c\in[C]} \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)\boldsymbol{\mu}_{m,c}^{(t)}}{\sum_{c\in[C]} \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)} \\
&= \frac{\sum_{c\in[C]} \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m) \frac{\sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)\mathbf{x}_{c,i}}{\sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)}}{\sum_{c\in[C]} \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)} \\
&= \frac{\sum_{c\in[C]} \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)\mathbf{x}_{c,i}}{\sum_{c\in[C]} \sum_{i\in[N_c]} q_{\mathbf{s}_{c,i}}^{(t)}(m)},
\end{aligned}$$

where the first and the second equation come from the definition of $\boldsymbol{\mu}_m^{(t)}$ and $\boldsymbol{\mu}_{m,c}^{(t)}$, respectively.

It is easy to verify that, $\boldsymbol{\mu}_m^{(t)}$ is the minimizer of the objective$\sum_{c=1}^{C} \sum_{i=1}^{N_c} q_{\mathbf{s}_{c,i}}^{(t)}(m)\|\mathbf{x}_{c,i} - \boldsymbol{\mu}\|_2^2$. Therefore, we have

$$\sum_{c=1}^{C}\sum_{i=1}^{N_c} q_{\mathbf{s}_{c,i}}^{(t)}(m)\|\mathbf{x}_{c,i} - \boldsymbol{\mu}_m^{(t)}\|_2^2 \leq \sum_{c=1}^{C}\sum_{i=1}^{N_c} q_{\mathbf{s}_{c,i}}^{(t)}(m)\|\mathbf{x}_{c,i} - \boldsymbol{\mu}_m^{(t-1)}\|_2^2. \tag{6}$$

And further,

$$\begin{aligned}
G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t-1)}) &:= \sum_{c=1}^{C}\sum_{i=1}^{N_c}\sum_{m=1}^{M} q_{\mathbf{s}_{c,i}}^{(t)}(m)\Big[\log\big(\pi_c^{(t-1)}(m)\big) + \frac{d\log(2\pi)}{2} - \frac{1}{2}\|\mathbf{x}_{c,i} - \boldsymbol{\mu}_m^{(t)}\|_2^2 - \log\big(q_{\mathbf{s}_{c,i}}^{(t)}(m)\big)\Big] \\
&\geq \sum_{c=1}^{C}\sum_{i=1}^{N_c}\sum_{m=1}^{M} q_{\mathbf{s}_{c,i}}^{(t)}(m)\Big[\log\big(\pi_c^{(t-1)}(m)\big) + \frac{d\log(2\pi)}{2} - \frac{1}{2}\|\mathbf{x}_{c,i} - \boldsymbol{\mu}_m^{(t-1)}\|_2^2 - \log\big(q_{\mathbf{s}_{c,i}}^{(t)}(m)\big)\Big] \\
&= G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t-1)}, \pi_{1:C}^{(t-1)}).
\end{aligned}$$

□

Lemma 3. At any time step t, $G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t)}) \geq G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t-1)})$.

Proof. Notice that

$$G_c^{(t)}(\boldsymbol{\mu}_{1:M}, \pi_c) := \sum_{i=1}^{N_c}\sum_{m=1}^{M} q_{\mathbf{s}_{c,i}}^{(t)}(m)\Big[\log\big(\pi_c(m)\big) + \log\big(\mathcal{N}(\mathbf{x}_{c,i}; \boldsymbol{\mu}_m, \mathbf{I})\big) - \log\big(q_{\mathbf{s}_{c,i}}^{(t)}(m)\big)\Big].$$

We have for any π and any $\boldsymbol{\mu}_{1:M}$,

$$\begin{aligned}
G_c^{(t)}(\boldsymbol{\mu}_{1:M}, \pi_c^{(t)}) - G_c^{(t)}(\boldsymbol{\mu}_{1:M}, \pi) &= \sum_{i=1}^{N_c}\sum_{m=1}^{M} q_{\mathbf{s}_{c,i}}^{(t)}(m)\Big[\log\big(\pi_c^{(t)}(m)\big) - \log\big(\pi_c(m)\big)\Big] \\
&= N_c \sum_{m=1}^{M} \frac{1}{N_c}\sum_{i=1}^{N_c} q_{\mathbf{s}_{c,i}}^{(t)}(m)\Big[\log\big(\pi_c^{(t)}(m)\big) - \log\big(\pi_c(m)\big)\Big] \\
&= N_c \sum_{m=1}^{M} \pi_c^{(t)}(m)\Big[\log\big(\pi_c^{(t)}(m)\big) - \log\big(\pi_c(m)\big)\Big] \\
&= N_c \cdot \mathrm{KL}(\pi_c^{(t)}\|\pi_c) \geq 0,
\end{aligned}$$

where the third equation comes from the definition of $\pi_c^{(t)}$, and the last equation comes from the definition of the KL-divergence.

Therefore, we have

$$\begin{aligned} G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t)}) &= \sum_{c=1}^{C} G_c^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_c^{(t)}) \\ &\geq \sum_{c=1}^{C} G_c^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_c^{(t-1)}) \\ &= G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t-1)}). \end{aligned}$$

□

Proof of Theorem 1. By Lemma 2 and Lemma 3, we have for any $t > 0$,

$$G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t)}) \geq G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t-1)}, \pi_{1:C}^{(t-1)}),$$

which further gives:

$$F(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t)}) \geq G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t)}) \geq G^{(t)}(\boldsymbol{\mu}_{1:M}^{(t-1)}, \pi_{1:C}^{(t-1)}) = F(\boldsymbol{\mu}_{1:M}^{(t-1)}, \pi_{1:C}^{(t-1)}).$$

Here, the first inequality holds because $G^{(t)}$ is a surrogate that always satisfies $F(\cdot) \geq G^{(t)}(\cdot)$; the last equation holds as we discussed at the beginning of this section.

This actually shows that $F(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t)})$ is monotonically increasing, and since $F(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t)})$ is upper bounded by some constant F^*, it is easy to show

$$\frac{1}{T}\sum_{t=2}^{T} |F(\boldsymbol{\mu}_{1:M}^{(t)}, \pi_{1:C}^{(t)}) - F(\boldsymbol{\mu}_{1:M}^{(t-1)}, \pi_{1:C}^{(t-1)})| \leq \frac{1}{T}(F^* - F(\boldsymbol{\mu}_{1:M}^{(1)}, \pi_{1:C}^{(1)})) = O(T^{-1}).$$

□

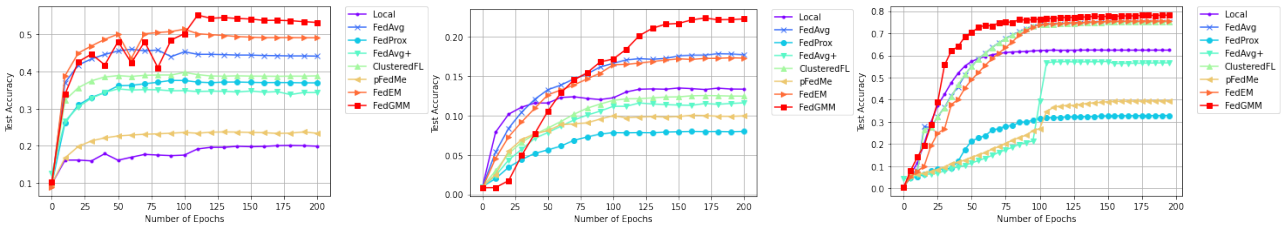
C. Appendix for Experiments.

C.1. Details of Training Configuration

Hardware and Implementations. In this paper, we implemented our method on a Linux machine with 8 NVIDIA A100 GPUs, each with 80GB of memory. The software environment is CUDA 11.6 and Driver Version 520.61.05. We used Python 3.9.13 and Pytorch 1.12.1 (Paszke et al., 2019) to construct our project.

Hyperparameters, Architecture, and Dataset Split. In our experiments, we use grid search to obtain the best performance. We provide all of the hyperparameters as well as their configurations in the following:

- Optimizer: SGD is chosen as the local solver, as in (Marfoq et al., 2021). For each method, we follow (Marfoq et al., 2021) to tune the learning rate via grid search in the range $\{10^{-0.5}, 10^{-1}, 10^{-1.5}, 10^{-2}, 10^{-2.5}, 10^{-3}\}$ to obtain the best performances. For our proposed FedGMM, the learning rate is set to 0.01 on CIFA-R10, 0.001 on CIFAR-100 and FEMNIST.
- Number of Components: M_1 and M_2 of FedGMM are tuned via grid search. For our method M_1=3 and M_2=3. The setting is consistent with the setting of FedEM.
- Epochs and Batch Size: The total communication round is set to 200, and the batch size is set to 128.
- Supervised Learning Model Architecture: For fairness, for all baseline methods, including Local, FedAvg (McMahan et al., 2017), FedProx (Li et al., 2020), FedAvg+ (Jiang et al., 2019) and Clustered FL (Sattler et al., 2020),



(a) CIFAR-10 (b) CIFAR-100 (c) FEMNIST

Figure 4. Test accuracy on different datasets *w.r.t.* training epochs.

pFedMe (T Dinh et al., 2020) and FedEM (Marfoq et al., 2021), the supervised backbone is the same as ours. Following (Marfoq et al., 2021), we apply MobileNet-v2 as the supervised encoder backbone for CIFAR-10 and CIFAR-100 datasets. For FEMNIST, we use a 2-layer CNN + 2-layer FFN as the encoder, that is two convolutional layers (with 3×3 kernels), max pooling, and dropout, followed by a 128 unit dense layer as in (Reddi et al., 2020). We use Torchvision (Marcel & Rodriguez, 2010) to implement the MobileNet-v2.

- Dataset Split: For training, we sub-sampled 15% from FEMNIST datasets. Detailed dataset partitioning can be found in (Marfoq et al., 2021). The performance of our method is evaluated on the local test data on each client and we report the average accuracy of all clients.

C.2. Convergence Plots

Figure 4 shows the evolution of average test accuracy overtime for each experiment shown in Table 2. As shown in the table and the figure, FedGMM outperforms all the baselines. This is a result of its ability to construct personalized models based on the joint data distribution, effectively capturing the heterogeneity of each sample across different clients.

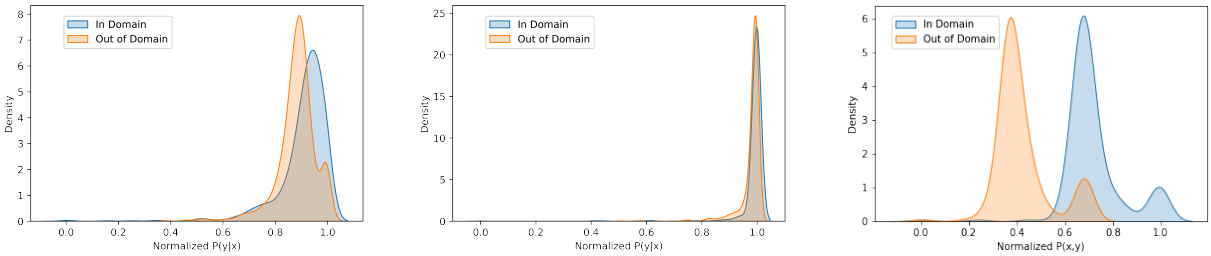
C.3. More Results on OOD Detection

To evaluate the OOD detection performance of FedGMM we first create a federated setting of MNIST by distributing samples with the same label across the clients according to a symmetric Dirichlet distribution with parameter 0.4, as in (Marfoq et al., 2021). Then the overall data are equally partitioned into two sets before being further dispatched to clients. The first set of data remains unchanged, and the second set of data is further equally partitioned into two subsets: 1) In the first subset of data, we simulate heterogeneity of $\mathbb{P}_c(\mathbf{x})$ by transforming sampled images with 90-degree rotation, horizontal flip, and inverse (Shorten & Khoshgoftaar, 2019) (such transformations are denoted by $T(\cdot)$); 2) In the second subset of data, we simulate heterogeneity of $\mathbb{P}_c(\mathbf{y}|\mathbf{x})$ by altering labels of sampled images to a randomly generated permutation (denoted by P_A).

During the evaluation stage, we examine whether a model can detect a testing sample is known or novel by the following steps: 1) we create two identical sets of test samples drawn from the same distribution of training data. The first set of test data remains unchanged. For the second set of test data, we simulate a different set of heterogeneity of $\mathbb{P}_c(\mathbf{x})$ by transforming sampled images with a scale factor of 0.5, 90-degree rotation, and horizontal flip (Shorten & Khoshgoftaar, 2019). 2) we labeled the first set of data as in-domain data and the second set of data as out-of-domain data.

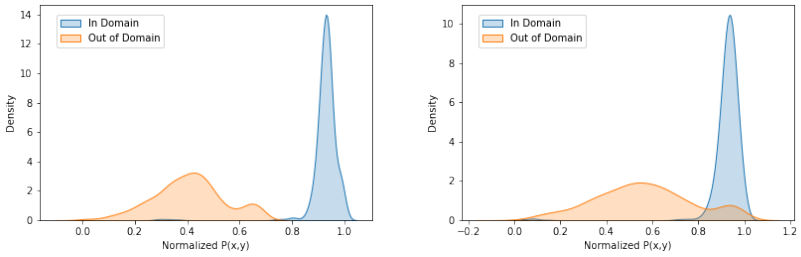
Similar to (Liu et al., 2020), in Figure 6, we visualized the normalized likelihood histogram of known and novel samples for FedGMM, FedEM, and FedAvg. The figures indicate the likelihoods of FedGMM are more distinguishable for known and novel samples than the baselines.

To further demonstrate the effectiveness of FedGMM, we visualized the frequency of samples *w.r.t.* the normalized likelihood against $\mathbb{P}(\boldsymbol{x})$ and $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$. For perturbing $\mathbb{P}(\boldsymbol{x})$, we only simulated a different set of heterogeneity of $\mathbb{P}_c(\mathbf{x})$ by transforming sampled images with a scale factor of 0.8, and 90-degree rotation (Shorten & Khoshgoftaar, 2019). For perturbing $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$, we only altered the labels of sampled images to a randomly generated permutation. The figures indicate the joint likelihood of FedGMM are more distinguishable against the changes of $\mathbb{P}(\boldsymbol{x})$ but slightly less distinguishable against the changes of $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$.



(a) FedAvg (b) FedEM (c) FedGMM

Figure 5. The frequency of samples *w.r.t.* the normalized likelihood for (a) FedAvg (b) FedEM and (c) FedGMM.



(a) Perturbed $\mathbb{P}(\boldsymbol{x})$ (b) Perturbed $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$

Figure 6. The frequency of samples *w.r.t.* the normalized likelihood for FedGMM on perturbed $\mathbb{P}(\boldsymbol{x})$ and $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$.

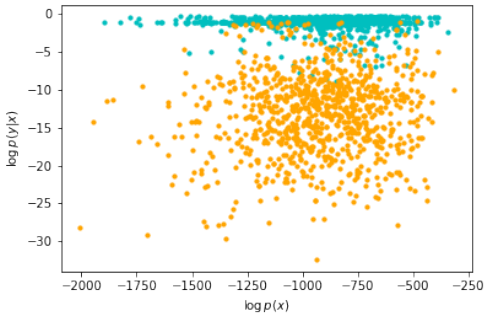
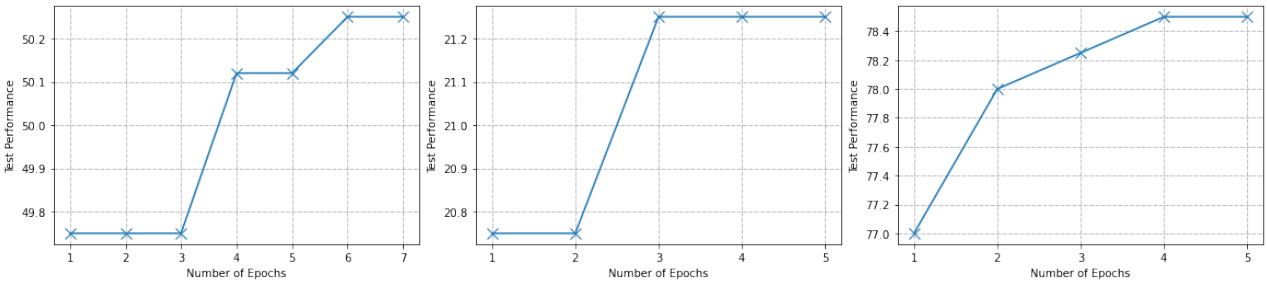


Figure 7. $\log \mathbb{P}(\boldsymbol{x})$ vs $\log \mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$ *w.r.t.* change of $\mathbb{P}(\boldsymbol{y}|\boldsymbol{x})$.

C.4. More Results on Adaptation to Unseen Clients

As discussed, FedGMM is flexible, enabling easy inference of new clients who did not participate in the training phase. The adaptation to unseen clients is accomplished by learning their personalized mixture weights. Such generalization only incurs minimal computational cost. We plot the accuracy with respect to the adaptation of π in Figure 8 on different datasets, from which we can see the adaptation only needs a small computational overhead.



(a) CIFAR-10 (b) CIFAR-100 (c) FEMNIST

Figure 8. Performance of FedGMM adapting to unseen clients (CIFAR-10, CIFAR-100, and FEMNIST) *w.r.t.* number of epochs.