Improved Modelling of Federated Datasets using Mixtures-of-Dirichlet-Multinomials

Jonathan Scott [1 2] **Áine Cahill** [2]

Abstract

In practice, training using federated learning can be orders of magnitude slower than standard centralized training. This severely limits the amount of experimentation and tuning that can be done, making it challenging to obtain good performance on a given task. Server-side proxy data can be used to run training simulations, for instance for hyperparameter tuning. This can greatly speed up the training pipeline by reducing the number of tuning runs to be performed overall on the true clients. However, it is challenging to ensure that these simulations accurately reflect the dynamics of the real federated training. In particular, the proxy data used for simulations often comes as a single centralized dataset without a partition into distinct clients, and partitioning this data in a naive way can lead to simulations that poorly reflect real federated training. In this paper we address the challenge of how to partition centralized data in a way that reflects the statistical heterogeneity of the true federated clients. We propose a fully federated, theoretically justified, algorithm that efficiently learns the distribution of the true clients and observe improved server-side simulations when using the inferred distribution to create simulated clients from the centralized data.

1. Introduction

Federated learning (FL) (McMahan et al., 2017) is a machine learning paradigm in which a (possibly very large) number of data holding devices, called clients, collaborate with a central server to train a model while keeping their data private and stored on device. FL has become the default for training on distributed private data with successful applications in a range of settings, including on mobile devices (Hard et al., 2018; Ramaswamy et al., 2019; Granqvist et al., 2020) as well as in healthcare (Brisimi et al., 2018; Huang et al., 2019; Rieke et al., 2020). Despite this, FL poses a multitude of challenges. These include: statistical and systems heterogeneity, high communication costs, high latency, low client compute and scheduling difficulties that arise from practical restrictions, such as a device needing to be charging and not in use to participate in training (Kairouz et al., 2021). This makes on device training not only technically challenging but also significantly more time consuming than standard centralized training. This effect is compounded by the fact that in most modern machine learning pipelines we do not train a model just once, but rather many times, to select the right architecture, optimization algorithm, hyperparameters (HP) etc. Running large scale model selection and HP tuning on real clients in a federated network is often infeasible, making it difficult to obtain good performance.

One way to address this issue is using simulations on the server. For instance, to select good model hyperparameters, which are then used directly in live training, thereby avoiding expensive HP tuning in the true federated network. This is possible because, in practice, it is common for the server to have some related proxy data that can be used to simulate the real federated training. For instance this could be public data from some related task, data from a particular sub-sample of consenting clients, or client data with certain sensitive features removed. Usually, however, the server-side data come without a client identifier, that is to say, it is a single central dataset with no natural partitioning into distinct FL clients. The question then arises how to use these data to create simulations that actually match the dynamics of live training. The naive approach would be to create clients from the proxy data by simply sub-sampling the data points in an IID fashion. However, this approach can fail to capture client data distribution heterogeneity and it is a well documented fact in FL that clients having non-IID data has a significant effect on FL training dynamics (Zhao et al., 2018; Li et al., 2020a;b).

In this paper we address the challenge of partitioning centralized data in a way that reflects the statistical heterogeneity of the true FL clients. The goal being to use this partition to run server-side federated training simulations,

*Equal contribution [1]Institute of Science and Technology Austria (ISTA) [2]Apple. Correspondence to: Jonathan Scott <jonathan.scott@ist.ac.at>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

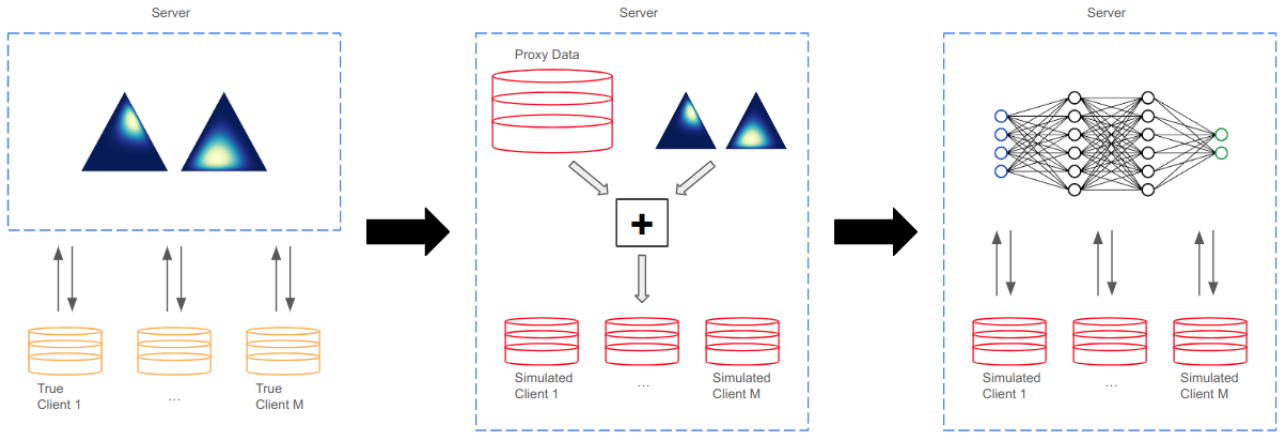


Figure 1. Proposed approach to server-side simulations. From left to right: learn Mixture-of-Dirichlet-Multinomials distribution from true federated clients (in this case 2 mixture components); use learned distribution to partition server proxy data into simulated clients; run server-side simulated federated model training using the simulated clients.

see Figure 1. On a high level our approach does this by choosing some categorical feature of the client data (for example the target class) and representing each client as a histogram over this feature. We propose using a Mixture-of-Dirichlet-Multinomials (MDM) as a probability distribution over these histograms, meaning that each client corresponds to a single draw from this distribution. We then use the observed client histograms to infer the parameters of our distribution by maximum likelihood estimation (MLE) using the true FL clients. Thus we obtain a learned distribution, which we can sample from, and these samples are histograms that look like the histograms we observed in the true clients. It is now simple to create simulated clients from our centralized data. First sample a histogram from our learned distribution, then sub-sample the centralized data so that its histogram matches this sampled histogram. In this way we create simulated clients whose data distributions match the true client distributions along the chosen feature.

There are several key features of our approach that make it effective in overcoming the challenges of a federated setting. Firstly, the inference algorithm is specifically designed to preserve client privacy. It adheres to the hard restriction that the server only works with aggregates of client statistics. This means a secure aggregator can be used, which is crucial to not exposing any single client's data. It also ensures the algorithm is compatible with differential privacy (DP) both on a user or an example level. By post-processing this means that simulations using DP inferred parameters do not degrade our privacy budget. Secondly, inference is efficient in terms of computation and communication. Clients do not run expensive model training, rather they only compute statistics of their data. These statistics, which are transferred between the server and clients, are low dimensional. This ensures low communication overhead. Finally, the proposed inference algorithm is designed not to have hyperparameters that need to be tuned, as such it can be run one-shot on the true clients. This is crucial given our goal is to limit how often we train on the true federated clients. The only hyperparameter choice that needs to be made is the number of mixture components of the distribution and, given the aforementioned efficiency, in practice we simply run a few reasonable options for this HP independently but in parallel. We are, to the best of our knowledge, the first work to deal with the challenge of explicitly modelling the statistical heterogeneity of FL clients. Our main contributions are:

- We propose a novel mixture distribution to model histograms of FL clients.
- We derive, with theoretical justification, a fully federated algorithm to efficiently infer the MLE parameters of this distribution over the true FL clients.
- We demonstrate empirically that this algorithm is able to successfully infer meaningful parameters.
- We show that using these inferred parameters to create simulated clients on the server leads to more representative training simulations.

Our experiments are implemented using the pfl-research framework (Granqvist et al., 2024). Our code can be found at `https://github.com/apple/pfl-research/tree/develop/publications/mdm`.

2. Related Work

Dirichlet Distributions in FL Dirichlet distributions are a popular tool used by FL researchers and practitioners to

create heterogeneous federated datasets out of existing centralized data to evaluate new learning methods (Yurochkin et al., 2019; Hsu et al., 2019; Li et al., 2022). Given some chosen value of the $\boldsymbol{\alpha}$ parameter, a federated dataset is created by sampling a class proportion vector $\mathbf{p} \sim \text{Dir}(\boldsymbol{\alpha})$ for each client and assigning data points to the client so that the assigned classes match the sampled class proportions. The level of heterogeneity is controlled by the magnitude of $\boldsymbol{\alpha}$. In this approach it is assumed that we know a priori how we would like to choose $\boldsymbol{\alpha}$, and the resulting federated dataset will reflect the desired level of heterogeneity specified by $\boldsymbol{\alpha}$. This paper focuses on the opposite scenario in which there exists some federated dataset, and we would like to choose the distribution parameters, such as $\boldsymbol{\alpha}$, that well reflect the heterogeneity of this dataset. This is a classic Bayesian inference problem which has been well studied in the context of Dirichlet distributions (Minka, 2000; Blei et al., 2001; Huang, 2005). The distribution we would like to infer is our own proposed Mixture-of-Dirichlet-Multinomials. A similar approach is taken by Holmes et al. (2012). Here the authors propose using a mixture of Dirichlet multinomials to model Microbial Metagenomic data, though their approach differs to ours in several key ways. Firstly, the proposed mixture distribution is different as Holmes et al. (2012) do not explicitly model the number of sample distributions for each component. Secondly, the inference procedures differ since Holmes et al. (2012) take a Maximum a posteriori (MAP) approach and solve the resulting MAP optimization using EM and leveraging optimization algorithms such BFGS. This approach assumes all data is centrally available and is incompatible with federated learning. In contrast, we take an approach of maximum likelihood estimation, and solve the MLE optimization using parameter update rules that are derived using generalized EM and compatible with the restrictions imposed by an FL setting.

Server-side data in FL Although a key tenant of FL is that data are decentralized and stored locally on client devices, it is common in practice that the central server possesses related proxy data. A number of works propose using server-side data to improve federated training in a range of settings. For instance, Hard et al. (2018) use public data to pre-train language models for improved next word prediction, before running federated training. Other works use server-side data during federated training. Song et al. (2022b) use server data to improve meta-gradient calculation in a meta learning approach to personalized FL. Gao et al. (2022); Dimitriadis et al. (2020) alternate between training on the clients and training on the server to mitigate the effects of client statistical heterogeneity and improve convergence and performance of acoustic models. Mai et al. (2022) analyse the convergence behaviour of such an alternating update scheme theoretically. Finally, in federated semi supervised learning, it is often assumed that the server possesses a small amount of labeled data which is used in combination with unlabelled client data (Diao et al., 2022; Jeong et al., 2021). Our viewpoint and approach differ from the above in that we are interested in how to use server side data to run realistic simulations that guide our downstream federated training.

Clustered and meta FL The notion that clients in a federated network can belong to different groups or clusters has received ample attention, particularly with respect to personalization in federated learning (Sattler et al., 2019; Ghosh et al., 2020; Mansour et al., 2020). Closely related to this is the multi-task or meta learning viewpoint of FL (Smith et al., 2017; Fallah et al., 2020; Scott et al., 2024) in which clients are thought of as samples (or tasks) drawn from some meta-distribution over clients. We draw inspiration from both the meta and clustered viewpoints in FL. We propose a meta-distribution over clients and infer the parameters of this distribution that explain the observed clients. Moreover, the form of this distribution is motivated by the view that clients naturally form clusters, with shared statistical properties within clusters.

3. Method

3.1. Background

Notation Let M denote the number of clients in a federated learning system. We use square bracket notation $[n] \coloneqq \{1, \dots, n\}$ to denote the first n integers. We assume that each client $i \in [M]$ has n_i data points $\mathbf{z}_1^i, \dots \mathbf{z}_{n_i}^i \in \mathbb{R}^d$. For instance, for a classification task we could have $\mathbf{z}_j^i = (\mathbf{x}_j^i, y_j^i)$. As is common in FL the clients may be statistically heterogeneous, meaning that the underlying data generation processes of the clients may differ from each other. This can occur in variety of ways including, but not limited to, differences between label distributions, feature distributions or the number of samples clients have (Song et al., 2022a). We assume we have an upper bound, N, on the number of samples any client holds, so that $n_i \leq N$ for all $i \in [M]$. Let $f \in [d]$ be some categorical feature of the data, which can take up to C different values, i.e. $z_{jf}^i \in [C]$ for all i, j. For instance f could be the target class in the case of a C-class classification task. We call f the *modelled feature*. For the modelled feature f each client computes a histogram, or count vector, $\mathbf{c}_i \in \mathbb{Z}^C$ of the feature. Namely, entry l of this vector counts how often f took value l across the client's dataset: $c_{il} = \sum_{j=1}^{n_i} \mathbb{1}_{z_{jf}^i = l}$. Note that $\sum_{l=1}^{C} c_{il} = n_i$. Thus each client is now represented as a tuple $(\mathbf{c}_i, n_i)$.

Mixture-of-Dirichlet-Multinomials Our goal is now to construct a statistical model (probability distribution) of the $(\mathbf{c}_i, n_i)$. Then drawing a new sample from this distribution is

like drawing the modelled feature histogram and the number of samples information of a new simulated client.

The building block of our distribution is the Dirichlet-Multinomial (DM) distribution. DM is a compound distribution over vectors of discrete counts parameterized by $\boldsymbol{\alpha} \in \mathbb{R}_+^C$ and $n \in \mathbb{N}$. A vector of counts $\mathbf{c} \in \mathbb{Z}^C$ is sampled by first drawing a probability vector from a Dirichlet distribution $\mathbf{p} \sim \text{Dir}(\boldsymbol{\alpha})$ then sampling the counts from a multinomial distribution $\mathbf{c} \sim \text{Mult}(n, \mathbf{p})$. The probability mass function for DM is given by

$$p(\mathbf{c} \mid n, \boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)\Gamma(n+1)}{\Gamma(n+\alpha_0)} \prod_{j=1}^{C} \frac{\Gamma(c_j+\alpha_j)}{\Gamma(\alpha_j)\Gamma(c_j+1)}, \quad (1)$$

where Γ is the gamma function and $\alpha_0 \coloneqq \sum_{j=1}^{C} \alpha_j$. We extend DM in two ways to better model clients in real federated learning scenarios. Firstly, since users may have different numbers of samples, we are interested in a distribution where n is not fixed. Hence, we jointly model the distribution of the number of samples and the count vector. We assume independence of n and $\boldsymbol{\alpha}$ so that we have

$$p(\mathbf{c}, n \mid \boldsymbol{\alpha}, \boldsymbol{\pi}) = p(\mathbf{c} \mid n, \boldsymbol{\alpha})\pi_n, \quad (2)$$

where $\pi_n \coloneqq p(n)$ parameterizes the probability that a client has n samples in total. Since we assume that the number of samples a client can possess is upper bounded by some constant N, we have that $\boldsymbol{\pi} \in \mathbb{R}^N$ and $\sum_{j=1}^{N} \pi_j = 1$.

Our second extension comes from the observation that in practice clients in a federated learning scenario might be naturally partitioned into a number of groups/clusters, where each cluster comes from a different meta distribution over clients. The natural way to model such an observation is using a mixture model. Let K be the number of components in the mixture. Then we define our Mixture-of-Dirichlet-Multinomials (MDM) model as the joint distribution over histograms and sample counts with the following probability mass function

$$q(\mathbf{c}, n \mid \boldsymbol{\tau}, \mathbf{A}, \boldsymbol{\Pi}) = \sum_{k=1}^{K} \tau_k p(\mathbf{c}, n \mid \boldsymbol{\alpha}_k, \boldsymbol{\pi}_k), \quad (3)$$

where $\boldsymbol{\tau} \in \mathbb{R}^K$ is the weight of each component, $\mathbf{A} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_K] \in \mathbb{R}^{K \times C}$ are the per component Dirichlet parameters and $\boldsymbol{\Pi} = [\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_K] \in \mathbb{R}^{K \times N}$ are the per component number of samples distributions. Additionally, p is defined through equations (1) and (2).

3.2. Parameter Inference

Given M clients in a federated learning system, each of which has a modelled feature histogram, number of samples tuple: $(\mathbf{c}_i, n_i)$, our goal is to infer $\boldsymbol{\tau}$, $\boldsymbol{\Pi}$ and $\mathbf{A}$ from Equation (3) that well explain the observed clients. Note that under the statistical model (3) each client corresponds to a single draw, or data point, from this distribution. Therefore, we can think of (3) as a 'meta' distribution over the clients. We take the approach of maximum likelihood estimation (MLE) to infer the statistical model parameters. Therefore we aim to maximize the log likelihood of the observed data under the MDM statistical model (3). The log likelihood is:

$$L(\boldsymbol{\tau}, \mathbf{A}, \boldsymbol{\Pi}) = \sum_{i=1}^{M} \log q(\mathbf{c}_i, n_i \mid \boldsymbol{\tau}, \mathbf{A}, \boldsymbol{\Pi}). \quad (4)$$

The objective function (4) is in general non-concave and we propose an EM-based algorithm to maximize it in a federated setting. In the remainder of this section we state this algorithm, which consists of two main parts: a single round of initialization of the statistical model parameters; followed by a multi-round iterative procedure that aims to maximize (4). Prior to starting inference it is necessary to specify the number of components, K, to use. In this section we assume the value of K has already been chosen. In Appendix A we outline a procedure for the server to choose the best value of K to use. In Section 3.3 we prove that the parameter updates used in the algorithm converge by deriving them as part of a generalized EM approach to maximizing the objective.

Initialization We start by initializing the parameters $\boldsymbol{\tau}$, $\boldsymbol{\Pi}$ and $\mathbf{A}$. The full procedure is given in Algorithm 1. The simplest parameter to initialize is $\boldsymbol{\tau}$, we set

$$\boldsymbol{\tau}^{(0)} = [\frac{1}{K}, \ldots, \frac{1}{K}], \quad (5)$$

namely all mixture components start with equal weight. Next we must initialize $\boldsymbol{\Pi}$ and $\mathbf{A}$, both of which consist of a parameter for each component k, namely $\boldsymbol{\pi}_k$ and $\boldsymbol{\alpha}_k$. To do this the server samples a single cohort of clients S_0 and each client $i \in S_0$ uniformly at random chooses a component, k_i, to contribute to initializing. To initialize the estimated number of samples distribution the server obtains a histogram of the number of samples of each client in each component and then normalizes this per component. Specifically, client i initializes a matrix of zeros $\mathbf{E}^i \in \mathbb{R}^{K \times N}$ and sets $E^i_{k_i n_i} = 1$. The server obtains the aggregate $\mathbf{E} = \sum_{i \in S_0} \mathbf{E}^i$ and then initializes $\boldsymbol{\Pi}$ by normalizing each row of $\mathbf{E}$ to obtain a per component probability distribution:

$$\boldsymbol{\pi}_k^{(0)} = \frac{1}{m_k} \mathbf{E}_k, \quad (6)$$

where $m_k = \sum_{j=1}^{S} E_{kj}$ is the number of clients that contributed to component k. Finally, we use a per component moment matching estimate to initialize $\mathbf{A}$. Namely, $\boldsymbol{\alpha}_k^{(0)}$ is chosen so that the first two moments of a Dirichlet distribution with parameter $\boldsymbol{\alpha}_k^{(0)}$ match the empirical moments of

Algorithm 1 Dirichlet-Multinomial Mixture Initialization

Input: client count tuples $(\mathbf{c}_i, n_i)_{i=1}^M$
$S_0 \leftarrow$ server samples a cohort of clients
for client i **in** S_0 **do**
  $k_i \sim \mathcal{U}\{1, \dots, K\}$
  $\mathbf{E}^i = \mathbf{0}^{K\times N}$, $E^i_{k_i n_i} = 1$
  $\mathbf{P}^i = \mathbf{0}^{K\times C}$, $\mathbf{P}^i_{k_i} = \frac{1}{n_i}\mathbf{c}_i$
  $\mathbf{Q}^i = \mathbf{0}^{K\times C}$, $\mathbf{Q}^i_{k_i} = (\frac{1}{n_i}\mathbf{c}_i) \odot (\frac{1}{n_i}\mathbf{c}_i)$
end for
$\mathbf{E} = \sum_{i\in S_0} \mathbf{E}^i$, $\mathbf{P} = \sum_{i\in S_0} \mathbf{P}^i$, $\mathbf{Q} = \sum_{i\in S_0} \mathbf{Q}^i$
$m_k = \sum_{j=1}^S E_{kj}$
$\bar{\mathbf{P}}_k = \frac{1}{m_k}\mathbf{P}_k$, $\bar{\mathbf{Q}}_k = \frac{1}{m_k}\mathbf{Q}_k$
$\boldsymbol{\tau}^{(0)} = [\frac{1}{K}, \dots, \frac{1}{K}]$
$\boldsymbol{\pi}^{(0)}_k = \frac{1}{m_k}\mathbf{E}_k$
$\boldsymbol{\alpha}^{(0)}_k = \frac{\bar{P}_{k1} - \bar{Q}_{k1}}{\bar{Q}_{k1} - \bar{P}^2_{k1}}\bar{\mathbf{P}}_k$
Output: $\boldsymbol{\tau}^{(0)}$, $\boldsymbol{\Pi}^{(0)}$, $\mathbf{A}^{(0)}$

the normalized histograms of the clients that are assigned to component k. That is to say, we assume that the client normalized histograms are drawn from a Dirichlet and we initialize under the constraint that moments of this Dirichlet match the empirical moments of the normalized histograms. Client i initializes zero matrices $\mathbf{P}^i, \mathbf{Q}^i \in \mathbb{R}^{K\times C}$ and sets $\mathbf{P}^i_{k_i} = \frac{1}{n_i}\mathbf{c}_i$ and $\mathbf{Q}^i_{k_i} = (\frac{1}{n_i}\mathbf{c}_i) \odot (\frac{1}{n_i}\mathbf{c}_i)$ where $\odot$ denotes entry-wise product. The server then obtains the aggregates

$$\mathbf{P} = \sum_{i\in S_0} \mathbf{P}^i \qquad \mathbf{Q} = \sum_{i\in S_0} \mathbf{Q}^i \tag{7}$$

and normalizes both row-wise by the number of clients contributing to each row (component), which is m_k.

$$\bar{\mathbf{P}}_k = \frac{1}{m_k}\mathbf{P}_k \qquad \bar{\mathbf{Q}}_k = \frac{1}{m_k}\mathbf{Q}_k. \tag{8}$$

Thus the server has computed the empirical estimates of the first two moments of the client normalized histograms for each component. Under the constraint that the first two moments of the kth Dirichlet must match the empirical moments (8) we have that

$$\boldsymbol{\alpha}^{(0)}_k = \frac{\bar{P}_{k1} - \bar{Q}_{k1}}{\bar{Q}_{k1} - \bar{P}^2_{k1}}\bar{\mathbf{P}}_k, \tag{9}$$

which gives $\mathbf{A}$'s initialization. Appendix B.1 derives (9).

Solving the MLE Our goal is to infer the maximum likelihood estimates for $\boldsymbol{\tau}$, $\boldsymbol{\Pi}$ and $\mathbf{A}$ which we do via an EM based approach, stated in full in Algorithm 2. In Section 3.3 we derive the update formulas and prove their convergence.

The server starts by running the initialization procedure from Algorithm 1. Inference of the parameters then takes place over T rounds. During each round the server samples a cohort of clients and sends to each the previous rounds computed estimates of the parameters. Each sampled client i then computes how well each component k describes their data using this latest estimate of the parameters. Let Z_i denote the latent (unobserved) random variable indicating to which mixture component client i belongs. The client then computes:

$$\omega^i_k = \Pr\{Z_i = k \mid \mathbf{c}_i, n_i, \boldsymbol{\tau}^{(t)}, \boldsymbol{\Pi}^{(t)}, \mathbf{A}^{(t)}\}, \tag{10}$$
$$\propto p(Z_i = k \mid \boldsymbol{\tau}^{(t)}) p(\mathbf{c}_i, n_i \mid \boldsymbol{\pi}^{(t)}_k, \boldsymbol{\alpha}^{(t)}_k), \tag{11}$$
$$\propto \tau^{(t)}_k p(\mathbf{c}_i \mid n_i, \boldsymbol{\alpha}^{(t)}_k) \pi^{(t)}_{k,n_i}, \tag{12}$$

using equation (1). The aggregates of the ω^i_k are needed by the server to compute the update to $\boldsymbol{\tau}$. For $\boldsymbol{\Pi}$ and $\mathbf{A}$ the client uses ω^i_k to weight their contribution to the kth component parameter updates on the server. Intuitively, ω^i_k tells us how likely it is that client i was drawn from component k and the higher this value is the more client i contributes to the kth component parameter update.

For $\boldsymbol{\Pi}$ each client i initializes $\mathbf{E}^i = \mathbf{0}^{K\times N}$ and sets column n_i equal to $\boldsymbol{\omega}^i$, so that $E^i_{kn_i} = \omega^i_k$ for $k = 1, \dots, K$. This corresponds to a soft assignment of the clients number of samples to the overall histogram computed across all clients. For the update to $\mathbf{A}$ the client computes two quantities to be aggregated by the server to be used in the parameter update:

$$\mathbf{u}^i_k = \omega^i_k \left(\psi(\mathbf{c}_i + \boldsymbol{\alpha}^{(t)}_k) - \psi(\boldsymbol{\alpha}^{(t)}_k)\right), \tag{13}$$
$$v^i_k = \omega^i_k \left(\psi(n_i + (\alpha^{(t)}_k)_0) - \psi((\alpha^{(t)}_k)_0)\right), \tag{14}$$

where ψ is the digamma function, which in (13) is applied entry-wise. The server gets aggregates of the client statistics

$$\boldsymbol{\omega} = \sum_{i\in S_{t+1}} \boldsymbol{\omega}^i \qquad \mathbf{E} = \sum_{i\in S_{t+1}} \mathbf{E}^i \tag{15}$$
$$\mathbf{u}_k = \sum_{i\in S_{t+1}} \mathbf{u}^i_k \qquad v_k = \sum_{i\in S_{t+1}} v^i_k \tag{16}$$

and uses these to compute the parameter updates

$$\boldsymbol{\tau}^{(t+1)} = \frac{1}{|S_{t+1}|}\boldsymbol{\omega}, \tag{17}$$
$$\boldsymbol{\pi}^{(t+1)}_k = \frac{1}{\omega_k}\mathbf{E}_k, \tag{18}$$
$$\boldsymbol{\alpha}^{(t+1)}_k = \frac{1}{v_k}\boldsymbol{\alpha}^{(t)}_k \odot \mathbf{u}_k. \tag{19}$$

3.3. Theoretical Results

In this section we state our theoretical result, in which we derive the previously stated parameter update formulas.

Algorithm 2 Dirichlet-Multinomial Mixture MLE

Input: client count tuples $(\mathbf{c}_i, n_i)_{i=1}^M$, steps T
Initialize $\boldsymbol{\tau}^{(0)}$, $\boldsymbol{\Pi}^{(0)}$, $\mathbf{A}^{(0)}$ using Algorithm 1.
for $t = 0$ **to** $T-1$ **do**
  $S_{t+1} \leftarrow$ server samples cohort of clients
  Server sends $(\boldsymbol{\tau}^{(t)}, \boldsymbol{\Pi}^{(t)}, \mathbf{A}^{(t)})$ to each client in S_{t+1}
  for client i **in** S_{t+1} **do**
    $\omega_k^i = \frac{\tau_k^{(t)} p(\mathbf{c}_i | n_i, \boldsymbol{\alpha}_k^{(t)}) \pi_{k,n_i}^{(t)}}{\sum_{k=1}^K \tau_k^{(t)} p(\mathbf{c}_i | n_i, \boldsymbol{\alpha}_k^{(t)}) \pi_{k,n_i}^{(t)}}$
    $\mathbf{E}^i = \mathbf{0}^{K \times N}$, $E_{kn_i}^i = \omega_k^i$
    $\mathbf{u}_k^i = \omega_k^i \left(\psi(\mathbf{c}_i + \boldsymbol{\alpha}_k^{(t)}) - \psi(\boldsymbol{\alpha}_k^{(t)}) \right)$
    $v_k^i = \omega_k^i \left(\psi(n_i + (\boldsymbol{\alpha}_k^{(t)})_0) - \psi((\boldsymbol{\alpha}_k^{(t)})_0) \right)$
  end for
  $\boldsymbol{\omega} = \sum_{i \in S_{t+1}} \boldsymbol{\omega}^i$, $\mathbf{E} = \sum_{i \in S_{t+1}} \mathbf{E}^i$
  $\mathbf{u}_k = \sum_{i \in S_{t+1}} \mathbf{u}_k^i$, $v_k = \sum_{i \in S_{t+1}} v_k^i$
  $\boldsymbol{\tau}^{(t+1)} = \frac{1}{|S_{t+1}|} \boldsymbol{\omega}$
  $\boldsymbol{\pi}_k^{(t+1)} = \frac{1}{\omega_k} \mathbf{E}_k$
  $\boldsymbol{\alpha}_k^{(t+1)} = \frac{1}{v_k} \boldsymbol{\alpha}_k^{(t)} \odot \mathbf{u}_k$
end for
Output: $\boldsymbol{\tau}^{(T)}$, $\boldsymbol{\Pi}^{(T)}$, $\mathbf{A}^{(T)}$

Theorem 3.1. *Let $(\mathbf{c}_i, n_i)_{i=1}^M$ be observed histogram, sample count data and $(\boldsymbol{\tau}^{(0)}, \boldsymbol{\Pi}^{(0)}, \mathbf{A}^{(0)})$ be an initialization of the parameters of the Mixture-of-Dirichlet-Multinomials model (3). For $t \geq 1$, $i = 1, \ldots, N$ and $k = 1, \ldots, K$ let*

$$\omega_k^i = p(Z_i = k \mid \mathbf{c}_i, n_i, \boldsymbol{\tau}^{(t)}, \boldsymbol{\Pi}^{(t)}, \mathbf{A}^{(t)}),$$

where Z_i is the latent variable indicating the mixture component that the ith sample was drawn from. Then the iteration

$$\tau_k^{(t+1)} = \frac{1}{M} \sum_{i=1}^M \omega_k^i$$

$$\pi_{kj}^{(t+1)} = \frac{1}{\sum_{i=1}^M \omega_k^i} \sum_{i=1}^M \omega_k^i \mathbb{1}_{n_i = j}$$

$$\alpha_{kj}^{(t+1)} = \alpha_{kj}^{(t)} \frac{\sum_{i=1}^M \omega_k^i \left(\psi(c_{ij} + \alpha_{kj}^{(t)}) - \psi(\alpha_{kj}^{(t)}) \right)}{\sum_{i=1}^M \omega_k^i \left(\psi(n_i + (\alpha_k^{(t)})_0) - \psi((\alpha_k^{(t)})_0) \right)}$$

corresponds to a generalized EM update of the log likelihood (4) and hence converges to a stationary point.

The proof is in Appendix B.2. These update rules are identical to those in Algorithm 2 except that the latter are computed on a subset of the data (equivalently a cohort of clients). Thus the updates in Algorithm 2 can be thought of as stochastic versions of the iteration given in Theorem 3.1.

4. Experiments

We separate our empirical evaluation into two orthogonal aspects. In Section 4.1 we investigate the inference of the MDM parameters, $\boldsymbol{\tau}$, $\mathbf{A}$ and $\boldsymbol{\Pi}$. We evaluate Algorithms 1 and 2 in terms of how accurately they recover ground truth parameters. We also examine whether they infer meaningful parameters on real federated datasets for which no ground truth values exist. In Section 4.2 we investigate the utility of the parameters after they have been inferred. We evaluate how well simulations run on the server using these inferred parameters reflect training on the real federated clients.

Datasets We evaluate using synthetic data that follows the MDM distribution, CIFAR10 (Krizhevsky, 2009), FEMNIST (Caldas et al., 2018) and Folktables (Ding et al., 2021). CIFAR10 is a non-federated multi-class classification dataset with 10 classes which we partition into clients with target class histograms that follow an MDM distribution. For inference we use the target class as the modelled feature over which we compute our histograms. FEMNIST is a federated dataset with a predefined partitioning into clients. It is a character recognition dataset with 62 classes (including digits and lower and upper case letters). Each character is uniquely identified with one of the 3,550 clients in the dataset. For FEMNIST we use the target class as the modelled feature. Folktables is a US census dataset where each datapoint corresponds to an individual person present in the census. The dataset has a natural partitioning into 2,373 clients based on geographical location. Specifically, a client holds the data of all individuals that live in the same PUMA code region. We model two separate features of the data: the race feature and the income feature. For full details of the dataset and its federated partitioning see Appendix C.2

Baselines The primary baseline for simulating users on the server is the IID approach. This first computes the true per client number of samples distribution. Then, to simulate a client, it draws a number of samples count n from this distribution and IID samples n points from the centralized dataset. In this baseline each client's distribution of the modelled feature (and all other features) is the same as the marginal distribution of that feature in the centralized dataset. We therefore call this baseline *fully IID* simulation. As a sanity check we also consider an oracle that cannot be used in practice but gives a valuable comparison to our MDM simulation. In this oracle, which we call *conditionally IID* simulation, we ensure that the simulated clients exactly follow the marginal distributions of the modelled feature of the true clients, but when conditioned on the modelled feature, the remaining features are IID. This is done as follows. First, each true client computes a histogram over the modelled feature. For each of these true histograms we

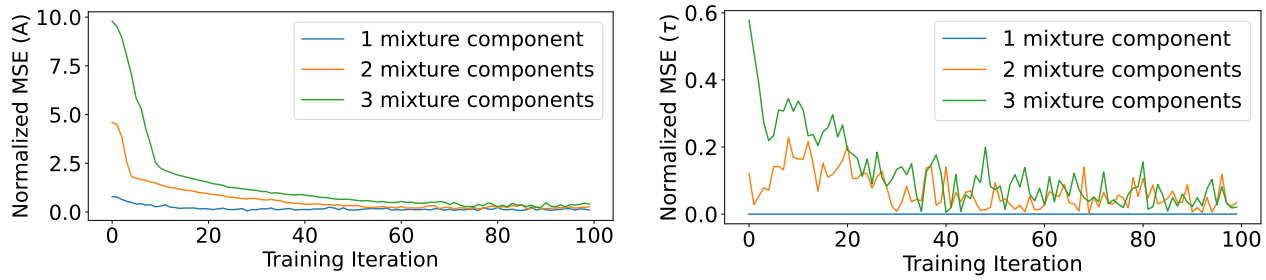


Figure 2. Normalized mean squared error (MSE) between the ground truth distribution parameter value and the inferred parameter value over time. Ground truth corresponds to medium levels of client statistical heterogeneity. On the left for $\mathbf{A}$, on the right for $\boldsymbol{\tau}$.

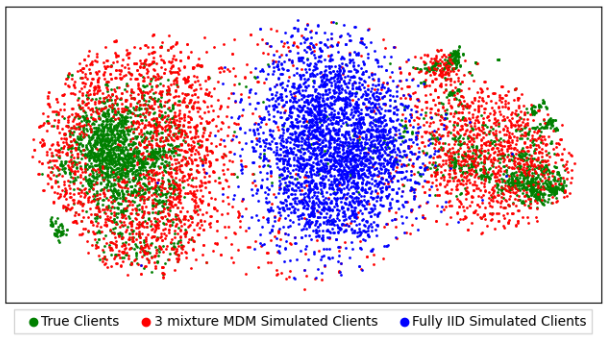


Figure 3. t-SNE visualization of FEMNIST clients, each point corresponds to a single client's class histogram. True clients (green), fully IID simulated clients (blue) and MDM clients (red).

randomly sample points from the centralized dataset while ensuring that the histogram of the modelled feature for these samples matches the true histogram. The simulated clients in this oracle capture the heterogeneity in the modelled feature but nothing more. The best we can hope for is for our MDM simulations to match this oracle as we also only capture the heterogeneity in the modelled feature. Note that in a real FL setting this oracle cannot be used as it requires each client to share with the server their histogram of the modelled feature, which could be a serious privacy leak.

4.1. Distribution Parameter Inference

Inference with known parameters We first assess the correctness of Algorithms 1 and 2 in terms of how accurately they recover ground truth distribution parameters for synthetic clients that follow our assumed MDM distribution. We test this over a range of different settings for $\boldsymbol{\tau}$, $\mathbf{A}$ and $\boldsymbol{\Pi}$. For 1, 2 and 3 mixture components we choose parameter values corresponding to low, medium and high levels of client heterogeneity. The exact values as well as additional experiment details can be found in Appendix C.3. The results for medium levels of heterogeneity can be see in Figure 2, with the others in Appendix D.1. We plot the mean squared error (MSE) of the inferred parameters from the ground truth values, normalized by the size of the ground truth parameter (see Appendix C.1), against the number of training iterations, T. As we can see in all cases we obtain approximate convergence and quickly in terms of the number rounds required. As expected, convergence is slower when we have more mixture components to infer.

Inference without known parameters We now consider the more challenging and interesting scenario where the true clients do not necessarily follow our assumed MDM distribution. We use the FEMNIST and Folktables datasets which are naturally partitioned into heterogeneous federated clients. For FEMNIST we take the target class to be the modelled feature and represent each client as a histogram over the classes they hold. For Folktables we model two features of the data independently, namely the race feature and the income feature. The race feature is categorical, with $C = 9$ possible values. The income feature is continuous, so in order to model it using MDM we convert it into a categorical feature by binning. Specifically, each client bins their continuous income value into one of $C = 41$ bins, with each bin having a range of \$5,000. We now represent clients as histograms over this binned income feature. For full details of the feature modelling on Folktables see Appendix C.2. We then run MDM inference on these histograms to obtain values for $\boldsymbol{\tau}$, $\mathbf{A}$ and $\boldsymbol{\Pi}$. Using these inferred parameter values we then sample from our distribution to obtain the histograms of new simulated clients. We then sample data for each client from the centralized version of the corresponding dataset so that each simulated client matches their sampled class histogram. As a baseline we also consider histograms corresponding to fully IID sampled clients. We plot 2-dimensional t-SNE visualizations of the results, for FEMNIST in Figure 3, and Folktables in Figure 4. Each point corresponds to a client (histogram), in green we have the true clients, in blue we have the simulated fully IID clients and in red the simulated MDM clients. We can draw several conclusions from these visualizations. Firstly, we

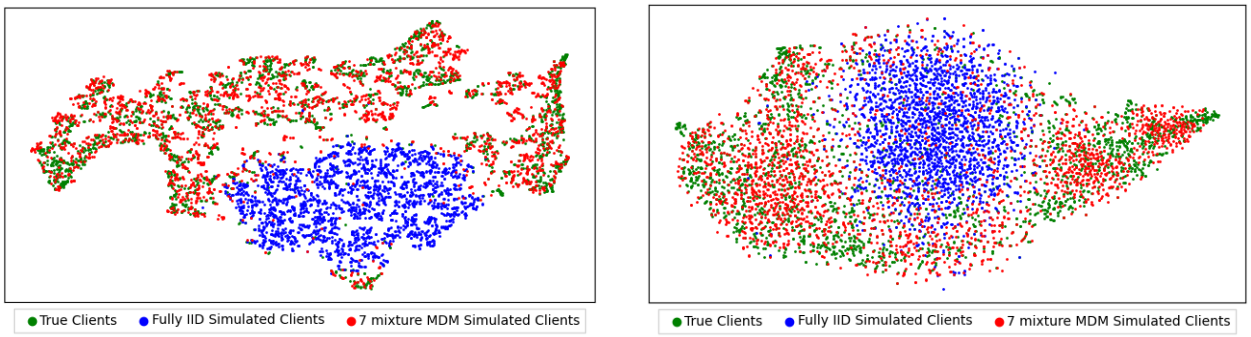


Figure 4. t-SNE visualizations of Folktables clients, each point corresponds to a single client's histogram over the race feature (left) and over the binned income feature (right). True clients (green), fully IID simulated clients (blue) and MDM clients (red). Inferred in both cases using $K = 7$.

observe in all three cases that the true users (green) exist in several distinct clusters, indicating that there are indeed different types of client heterogeneity present in the modelled feature. This validates our motivation for using a mixture model over the clients. Secondly, we see that in all cases the fully IID approach clearly fails to capture the heterogeneity in the client modelled feature distributions. Each of these simulated clients look like a subset of the centralized dataset, which poorly represents the true clients. Finally, in all cases the MDM distribution, with the parameters learned by Algorithms 1 and 2, does an admirable job of simulating the modelled feature heterogeneity of the true clients. This can be seen in the closeness between the true client distribution (green points) and the simulated client distribution (red points) in the t-SNE visualizations. In Appendix D.1 we include ablation studies where we infer the MDM parameters using a range of values for K and compare the differences in the corresponding simulated clients.

4.2. Federated Training Simulations

We now turn our attention to using our inferred parameters to run simulations on the server. We evaluate how closely model performance when training on simulated clients reflects performance when training on the true federated clients. As federated datasets we use FEMNIST as well as CIFAR10, partitioned into clients which follow an MDM distribution using the same settings of the parameters as in Section 4.1. The server-side proxy data is the centralized version of our federated dataset, i.e. the same data but without client identifiers. This setup ensures there is no distribution shift between the server and client data, which could be a confounding factor when evaluating the differences between simulated and true model training, beyond just the partitioning of the server data. In Appendix D.2 we provide results for additional experiments that simulate the practically relevant scenario when the server simulation data and client data are disjoint. We choose a range of HP settings and for each setting we train a model with federated averaging on the true clients, the MDM simulated clients, the fully IID simulated clients and the conditionally IID simulated clients. For FEMNIST we vary the local learning rate and local number of epochs used in FedAvg while for CIFAR10 we vary local batch size, local learning rate and local number of epochs. See Appendix C.4 for model and HP details. We compare the final accuracies of these trained models across the range of HP settings. The closer the final accuracy of the model trained on simulation clients is to the one trained on the true clients, the better, and more predictive the simulation is. Figures 5 and 6 show the results for FEMNIST and CIFAR10. The results for CIFAR10 are for true clients generated using 2 mixtures and high heterogeneity, see Section 4.1. The results for the other settings are in Appendix D.2. We can draw several conclusions from these plots. Firstly, for both datasets the accuracies are highest on the fully IID simulated clients across (nearly) all settings of the hyperparameters. In other words this baseline suffers from giving an overly optimistic prediction as to the performance on the true clients. This is to be expected given that federated averaging tends to perform better in the presence of low client heterogeneity. Secondly, training on the MDM simulated clients (red) gives a better indicator of performance on the true clients (dotted green) as seen by the closeness of the curves. Across all HP settings the mean of the absolute accuracy difference between the true clients and our MDM simulated clients was 1.6% compared to 3.3% for the fully IID simulated clients for FEMNIST and 0.8% compared to 6.6% for CIFAR10. Finally, while on CIFAR10 the MDM simulation exhibits near identical performance to the true clients, with non-trivial differences occurring on two particular settings on HPs which we attribute to randomness in the training process, for FEMNIST there is a non-negligible gap between MDM and the true clients. In general the MDM simulations overestimate performance on the true clients. We do observe, however, that the MDM simulations very closely match the conditionally

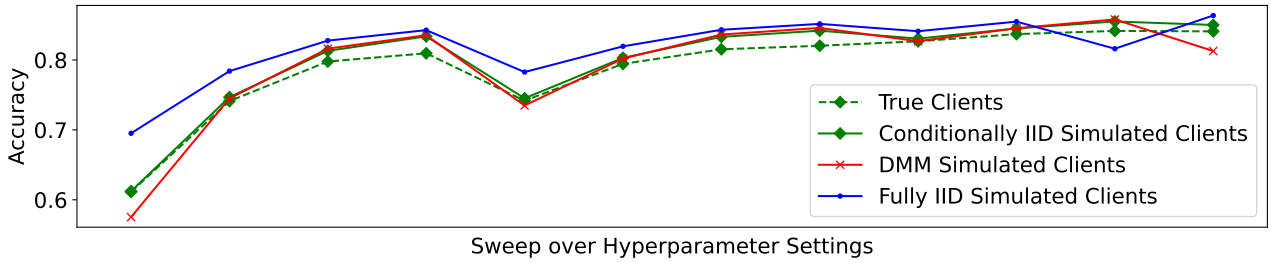


Figure 5. FEMNIST test accuracy when training with FedAvg for different settings of local learning rate and local epochs. True clients (dotted green), conditionally IID simulated clients (green), learned MDM simulated clients (red) and fully IID simulated clients (blue).

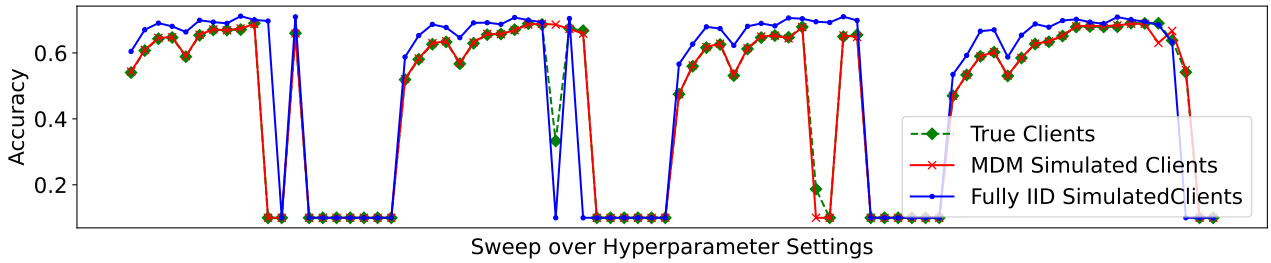


Figure 6. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue).

IID oracle (green) on FEMNIST. Recall that this oracle captures exactly the label heterogeneity of the true clients (but nothing more). This again confirms that the parameters we inferred for FEMNIST in Section 4.1 successfully model the true client label distributions, and that this transfers over to federated model training. It also confirms the existence of client heterogeneity within the true dataset that goes beyond just the client label distributions. In other words the MDM simulated clients make it part of the way in capturing the true client heterogeneity but there is still remaining heterogeneity in the other features that is affecting training.

5. Conclusion

In this paper we presented a novel approach to modelling heterogeneous FL clients using a Mixture-of-Dirichlet-Multinomials. We proposed an efficient and fully federated optimization procedure to infer the maximum likelihood estimates of the distribution parameters and showed theoretically the convergence of the update formulas. We empirically evaluated both the correctness of the proposed algorithm, in terms how well it infers the distributional parameters, as well as the utility of the inferred parameters themselves. We found that simulations run using clients generated with the inferred parameters were more representative of true federated training than those using IID clients. The proposed algorithm is private and compatible with differential privacy, on either a user or example level. In future work we would like to further investigate the combination of DP with our proposed inference method, both in the central DP setting, where a secure aggregator adds noise to the aggregated statistics, and the local DP setting where the clients themselves add noise prior to aggregation.

Acknowledgements

We would like to thank: Mona Chitnis and everyone in the Private Federated Learning team at Apple for their help and support throughout the entire project; Audra McMillan, Martin Pelikan, Anosh Raj and Barry Theobold for feedback on the initial versions of the paper; and Christoph Lampert for valuable feedback on the paper structure and suggestions for additional experiments.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. In Dietterich, T. G., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems:*

Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada], pp. 601–608. MIT Press, 2001. URL https://proceedings.neurips.cc/paper/2001/hash/296472c9542ad4d4788d543508116cbc-Abstract.html.

Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C., and Shi, W. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, 112:59–67, 2018. ISSN 1386-5056. doi: https://doi.org/10.1016/j.ijmedinf.2018.01.007. URL https://www.sciencedirect.com/science/article/pii/S138650561830008X.

Caldas, S., Wu, P., Li, T., Konečný, J., McMahan, H. B., Smith, V., and Talwalkar, A. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018. URL http://arxiv.org/abs/1812.01097.

Diao, E., Ding, J., and Tarokh, V. Semifl: Semi-supervised federated learning for unlabeled clients with alternate training. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Dimitriadis, D., Kumatani, K., Gmyr, R., Gaur, Y., and Eskimez, S. E. A federated approach in training acoustic models. In *INTERSPEECH 2020*, October 2020.

Ding, F., Hardt, M., Miller, J., and Schmidt, L. Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Fallah, A., Mokhtari, A., and Ozdaglar, A. E. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Gao, Y., Parcollet, T., Zaiem, S., Fernández-Marqués, J., de Gusmao, P. P. B., Beutel, D. J., and Lane, N. D. End-to-end speech recognition from federated acoustic models. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2022, Virtual and Singapore, 23-27 May 2022*, pp. 7227–7231. IEEE, 2022. doi: 10.1109/ICASSP43922.2022.9747161. URL https://doi.org/10.1109/ICASSP43922.2022.9747161.

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. An efficient framework for clustered federated learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Granqvist, F., Seigel, M., van Dalen, R. C., Cahill, Á., Shum, S., and Paulik, M. Improving on-device speaker verification using federated learning with privacy. In Meng, H., Xu, B., and Zheng, T. F. (eds.), *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pp. 4328–4332. ISCA, 2020. URL https://doi.org/10.21437/Interspeech.2020-2944.

Granqvist, F., Song, C., Cahill, Á., van Dalen, R. C., Pelikan, M., Chan, Y. S., Feng, X., Krishnaswami, N., Jina, V., and Chitnis, M. pfl-research: simulation framework for accelerating research in private federated learning. *CoRR*, abs/2404.06430, 2024. doi: 10.48550/ARXIV.2404.06430. URL https://doi.org/10.48550/arXiv.2404.06430.

Hard, A., Rao, K., Mathews, R., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018. URL http://arxiv.org/abs/1811.03604.

Holmes, I., Harris, K., and Quince, C. Dirichlet multinomial mixtures: Generative models for microbial metagenomics. *PLOS ONE*, 7:1–15, 02 2012. doi: 10.1371/journal.pone.0030126. URL https://doi.org/10.1371/journal.pone.0030126.

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

Huang, J. Maximum likelihood estimation of dirichlet distribution parameters, 2005.

Huang, L., Shea, A. L., Qian, H., Masurkar, A., Deng, H., and Liu, D. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *J. Biomed. Informatics*, 99, 2019. doi: 10.1016/J.JBI.2019.103291. URL https://doi.org/10.1016/j.jbi.2019.103291.

Jeong, W., Yoon, J., Yang, E., and Hwang, S. J. Federated semi-supervised learning with inter-client consistency & disjoint learning. In *9th International Conference on*

Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1–2), 2021.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Li, Q., Diao, Y., Chen, Q., and He, B. Federated learning on non-iid data silos: An experimental study. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*, pp. 965–978. IEEE, 2022. URL https://doi.org/10.1109/ICDE53745.2022.00077.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In Dhillon, I. S., Papailiopoulos, D. S., and Sze, V. (eds.), *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020a. URL https://proceedings.mlsys.org/book/316.pdf.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL https://openreview.net/forum?id=HJxNAnVtDS.

Mai, V. S., La, R. J., and Zhang, T. Federated learning with server learning: Enhancing performance for non-iid data. *CoRR*, abs/2210.02614, 2022. doi: 10.48550/ARXIV.2210.02614. URL https://doi.org/10.48550/arXiv.2210.02614.

Mansour, Y., Mohri, M., Ro, J., and Suresh, A. T. Three approaches for personalization with applications to federated learning. *CoRR*, abs/2002.10619, 2020. URL https://arxiv.org/abs/2002.10619.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Minka, T. P. Estimating a dirichlet distribution, 2000.

Ramaswamy, S., Mathews, R., Rao, K., and Beaufays, F. Federated learning for emoji prediction in a mobile keyboard. *CoRR*, abs/1906.04329, 2019. URL http://arxiv.org/abs/1906.04329.

Rieke, N., Hancox, J., Li, W., Milletarì, F., Roth, H. R., Albarqouni, S., Bakas, S., Galtier, M. N., Landman, B. A., Maier-Hein, K., Ourselin, S., Sheller, M., Summers, R. M., Trask, A., Xu, D., Baust, M., and Cardoso, M. J. The future of digital health with federated learning. *npj Digital Medicine*, 3(1):119, 09 2020. ISSN 2398-6352. doi: 10.1038/s41746-020-00323-1. URL https://doi.org/10.1038/s41746-020-00323-1.

Sattler, F., Müller, K., and Samek, W. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *CoRR*, abs/1910.01991, 2019. URL http://arxiv.org/abs/1910.01991.

Scott, J., Zakerinia, H., and Lampert, C. H. PeFLL: Personalized federated learning by learning to learn. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/pdf/2306.05515.pdf.

Smith, V., Chiang, C., Sanjabi, M., and Talwalkar, A. Federated multi-task learning. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4424–4434, 2017.

Song, C., Granqvist, F., and Talwar, K. Flair: Federated learning annotated image repository. In *NeurIPS*, 2022a. URL https://arxiv.org/abs/2207.08869.

Song, J., Oh, M. H., and Kim, H. Personalized federated learning with server-side information. *IEEE Access*, 10: 120245–120255, 2022b.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K. H., Hoang, T. N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning (ICML)*, volume 97, pp. 7252–7261. PMLR, 2019.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *CoRR*, abs/1806.00582, 2018. URL http://arxiv.org/abs/1806.00582.

A. How to select the number of mixture components

We detail here a strategy for choosing which K to use in Algorithms 1 and 2 based on selecting the value that maximizes the log likelihood of a validation cohort of clients.

A.1. Choosing K by validation log likelihood

1. Run Algorithms 1 and 2 until convergence on multiple choices of K in parallel.
2. Sample a new cohort of clients that we have not yet seen and for each choice of K evaluate the log likelihood, quation 4, on this cohort of clients.
3. Use the K that gave the highest log likelihood on this validation cohort of clients. In the case of (approximate) ties choose the smallest K.

The crucial point here is that this procedure to choose K can be done one shot and does not require multiple federated training runs which would be highly inefficient. This is because practically it is possible to run inference using multiple values of K in parallel due to the very low computational and communication related overheads of Algorithms 1 and 2.

A.2. Experimental evaluation

We provide here an experimental evaluation of the proposed method. Our evaluation procedure is as follows. We first fix the values of the ground truth MDM parameters. Specifically, we set the ground truth number of mixture components to $K = 3$, the mixture weights to

$$\boldsymbol{\tau} = [0.2, 0.5, 0.3],$$

the Dirichlet parameters to

$$\mathbf{A} = \begin{bmatrix} 0.1 & 0.2 & 0.1 & 0.3 & 0.1 \\ 1 & 4 & 1 & 2 & 0.5 \\ 10 & 5 & 3 & 2 & 30 \end{bmatrix},$$

and we set the number of samples of each component to 100, with $\mathbf{\Pi}$ defined accordingly. We then draw M clients (histograms) from this ground truth MDM distribution, and for $K \in \{1, \ldots, 6\}$ we infer MDM parameters using Algorithms 1 and 2 from the paper, using this sample of clients. Finally, we sample a new validation cohort of 1000 clients from the true distribution and we compute the mean log likelihood of this validation cohort, using the parameters infered for each K. We do this for $M = 100, 200$ and 1000 and plot the results in Figure 7.

As we can see in the figure, for each M, step 3 of the given procedure (choose the smallest value of K that gives the maximum log likelihood), leads to us choosing to use the correct ground truth value of $K = 3$. Interestingly, when we infer using a smaller number of clients (100 and 200) we observe some overfitting when K is chosen to be too large. This effect is largely mitigated by inferring on a greater number of clients.

B. Proofs

B.1. Derivation of moment matching estimate

Let $\mathbf{p} \sim \text{Dir}(\boldsymbol{\alpha})$ and recall that $\alpha_0 := \sum_{j=1}^{C} \alpha_j$. Then for all $j \in [C]$ we have that the first two moments of the Dirichlet are:

$$\mathbb{E}\, p_j = \frac{\alpha_j}{\alpha_0}, \tag{20}$$

$$\mathbb{E}\, p_j^2 = \frac{\alpha_j(1+\alpha_j)}{\alpha_0(1+\alpha_0)}. \tag{21}$$

Moreover, from these two equations it can be checked that

$$\alpha_0 = \frac{\mathbb{E}\, p_1 - \mathbb{E}\, p_1^2}{\mathbb{E}\, p_1^2 - (\mathbb{E}\, p_1)^2}. \tag{22}$$

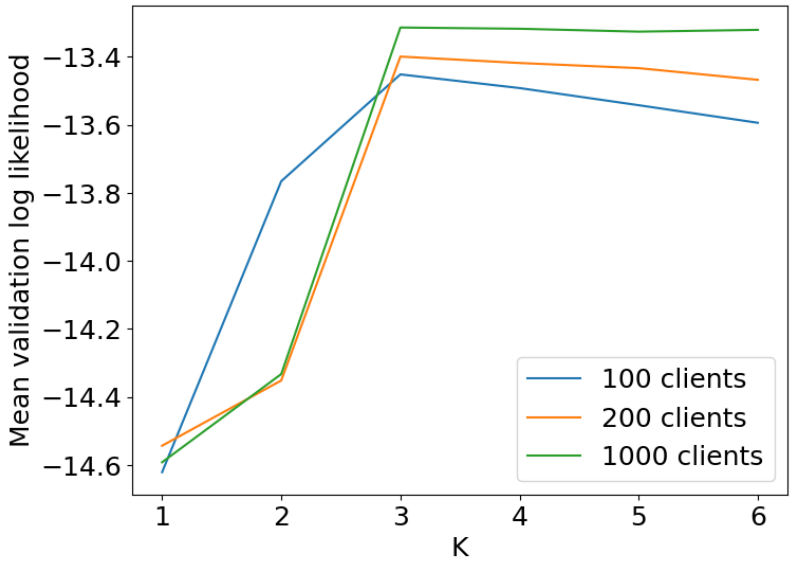


Figure 7. Mean log likelihood on the validation cohort of clients with the parameters inferred using different values of K.

Therefore, combining equations (20) and (22) we obtain:

$$\alpha_j = \frac{\mathbb{E}\, p_1 - \mathbb{E}\, p_1^2}{\mathbb{E}\, p_1^2 - (\mathbb{E}\, p_1)^2}\, \mathbb{E}\, p_j. \tag{23}$$

This is precisely the moment matching estimate used in algorithm 1 where each client normalizes their count vector $\mathbf{c}_i$ to a probability vector $\mathbf{p}_i = \frac{1}{n_i}\mathbf{c}_i$ which we then assume is drawn from a Dirichlet distribution. Equation (23) is then used to initialize $\boldsymbol{\alpha}$ where the true expectations are replaced by the empirical mean over the sampled clients.

B.2. Proof of Theorem 3.1

We let $\boldsymbol{\theta} = (\boldsymbol{\tau}, \mathbf{A}, \boldsymbol{\Pi})$ denote the parameter variables and $\boldsymbol{\theta}^{(t)} = (\boldsymbol{\tau}^{(t)}, \mathbf{A}^{(t)}, \boldsymbol{\Pi}^{(t)})$ denote the current values of the parameters after t steps. Let Z_i be the latent (unobserved) variable denoting which component the ith observation was sampled from. Following standard expectation maximization our goal will be to improve Q, which is guaranteed to lead to improvements in the log likelihood L.

$$\begin{aligned}
Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) &= \mathop{\mathbb{E}}_{Z \sim p(\cdot \mid \mathbf{C}, \mathbf{n}, \boldsymbol{\theta}^{(t)})} \log p(\mathbf{C}, \mathbf{n}, Z \mid \boldsymbol{\theta}) && (24)\\
&= \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_k^i \log p(\mathbf{c}_i, n_i, Z_i = k \mid \boldsymbol{\theta}) && (25)\\
&= \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_k^i (\log p(\mathbf{c}_i, n_i \mid \boldsymbol{\alpha}_k, \boldsymbol{\pi}_k) + \log \tau_k) && (26)\\
&= \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_k^i (\log p(\mathbf{c}_i \mid n_i, \boldsymbol{\alpha}_k) + \log \boldsymbol{\pi}_{k n_i} + \log \tau_k) && (27)\\
&= \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_k^i \log p(\mathbf{c}_i \mid n_i, \boldsymbol{\alpha}_k) + \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_k^i \log \boldsymbol{\pi}_{k n_i} + \sum_{i=1}^{M} \sum_{k=1}^{K} \omega_k^i \log \tau_k && (28)
\end{aligned}$$

where $\omega_k^i = p(Z_i = k \mid \mathbf{c}_i, n_i, \boldsymbol{\theta}^{(t)})$ is the probability that the ith observation came from the kth mixture component. This can be computed via Bayes rule:

$$\omega_k^i = \frac{p(\mathbf{c}_i, n_i \mid Z_i = k, \boldsymbol{\theta}^{(t)}) p(Z_i = k \mid \boldsymbol{\theta}^{(t)})}{\sum_{k=1}^K p(\mathbf{c}_i, n_i \mid Z_i = k, \boldsymbol{\theta}^{(t)}) p(Z_i = k \mid \boldsymbol{\theta}^{(t)})} \tag{29}$$

$$= \frac{p(\mathbf{c}_i \mid n_i, \boldsymbol{\alpha}_k^{(t)}) \pi_{n_i}^{(t)} \tau_k^{(t)}}{\sum_{k=1}^K p(\mathbf{c}_i \mid n_i, \boldsymbol{\alpha}_k^{(t)}) \pi_{n_i}^{(t)} \tau_k^{(t)}}. \tag{30}$$

Now that we have computed Q we seek to find new parameter values at step $t+1$ that increase the value of Q compared to at step t. Note that in standard EM we would be looking to compute $\boldsymbol{\theta}^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)})$, however, in order to make the later application to federated learning practical we are satisfied here with a weaker condition, namely we aim to find $\boldsymbol{\theta}^{(t+1)}$ such that $Q(\boldsymbol{\theta}^{(t+1)} \mid \boldsymbol{\theta}^{(t)}) \geq Q(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}^{(t)})$. This is still sufficient to guarantee improvements in the log likelihood L, which follows from the standard proof of correctness of EM.

Given that in (28) our variables appear in a decoupled form we can handle each separately. For $\boldsymbol{\Pi}$ and $\boldsymbol{\tau}$ this is quite simple and we can find a closed form solution that in fact maximizes each term. For $\mathbf{A}$ this is trickier as the DM log likelihood does not have a closed form solution. Instead we derive a fixed point update based on the one in (Minka, 2000), that leads to an improvement in the term.

Firstly, for $\boldsymbol{\Pi}$. We compute the Laplacian

$$F(\boldsymbol{\Pi}, \Lambda) = \sum_{i=1}^M \sum_{k=1}^K \omega_k^i \log \boldsymbol{\pi}_{k n_i} - \sum_{k=1}^K \lambda_k \sum_{j=1}^S (\pi_{kj} - 1), \tag{31}$$

$$= \sum_{j=1}^S \sum_{k=1}^K (\sum_{i=1}^M \omega_k^i \mathbb{1}_{n_i = j}) \log \boldsymbol{\pi}_{kj} - \sum_{k=1}^K \lambda_k \sum_{j=1}^S (\pi_{kj} - 1). \tag{32}$$

Taking derivatives we obtain

$$\frac{\partial F}{\partial \pi_{kj}} = \frac{1}{\pi_{kj}} \sum_{i=1}^M \omega_k^i \mathbb{1}_{n_i = j} - \lambda_k, \tag{33}$$

$$\frac{\partial F}{\partial \lambda_k} = \sum_{j=1}^S (\pi_{kj} - 1). \tag{34}$$

Setting to 0 and solving for π_{kj} we obtain

$$\pi_{kj} = \frac{1}{\sum_{i=1}^M \omega_k^i} \sum_{i=1}^M \omega_k^i \mathbb{1}_{n_i = j}, \tag{35}$$

where $\mathbb{1}_{n_i = j} = 1$ if $n_i = j$ and 0 otherwise. Secondly, for $\boldsymbol{\tau}$. We again define the Laplacian as

$$G(\boldsymbol{\tau}, \lambda) = \sum_{i=1}^M \sum_{k=1}^K \omega_k^i \log \tau_k - \lambda (\sum_{k=1}^K \tau_k - 1). \tag{36}$$

Taking derivatives we obtain

$$\frac{\partial G}{\partial \tau_k} = \frac{1}{\tau_k} \sum_{i=1}^M \omega_k^i - \lambda, \tag{37}$$

$$\frac{\partial G}{\partial \lambda} = \sum_{k=1}^K \tau_k - 1. \tag{38}$$

Setting to 0 and solving for τ_k we obtain

$$\tau_k = \frac{1}{N}\sum_{i=1}^{M}\omega_k^i. \tag{39}$$

Finally, we deal with $\mathbf{A}$. Let

$$H(\mathbf{A}) = \sum_{i=1}^{M}\sum_{k=1}^{K}\omega_k^i \log p(\mathbf{c}_i \mid n_i, \boldsymbol{\alpha}_k) \tag{40}$$

$$= \sum_{k=1}^{K}\sum_{i=1}^{M}\omega_k^i \log \frac{\Gamma((\boldsymbol{\alpha}_k)_0)\Gamma(n_i+1)}{\Gamma(n_i+(\boldsymbol{\alpha}_k)_0)}\prod_{j=1}^{C}\frac{\Gamma(\mathbf{c}_{ij}+\boldsymbol{\alpha}_{kj})}{\Gamma(\boldsymbol{\alpha}_{kj})\Gamma(\mathbf{c}_{ij}+1)} \tag{41}$$

$$= \sum_{k=1}^{K}\sum_{i=1}^{M}\omega_k^i \left(\log \frac{\Gamma((\boldsymbol{\alpha}_k)_0)}{\Gamma(n_i+(\boldsymbol{\alpha}_k)_0)} + \sum_{j=1}^{C}\log\frac{\Gamma(\mathbf{c}_{ij}+\boldsymbol{\alpha}_{kj})}{\Gamma(\boldsymbol{\alpha}_{kj})}\right) + D \tag{42}$$

where D is constant w.r.t $\mathbf{A}$. We now recall the following two bounds from (Minka, 2000)

$$\frac{\Gamma(x)}{\Gamma(n+x)} \geq \frac{\Gamma(\hat{x})}{\Gamma(n+\hat{x})}\exp\left((\hat{x}-x)b\right) \tag{43}$$

where $b = \psi(n+\hat{x}) - \psi(\hat{x})$, and ψ is the digamma function, and

$$\frac{\Gamma(n+x)}{\Gamma(x)} \geq cx^a \tag{44}$$

where $a = (\psi(n+\hat{x}) - \psi(\hat{x}))\hat{x}$ and $c = \frac{\Gamma(n+\hat{x})}{\Gamma(\hat{x})}\hat{x}^{-a}$. These bounds hold for all $x, \hat{x} \geq 0$ and all $n \geq 1$. Moreover, both bounds are tight when $x = \hat{x}$. We apply (43) with $x = (\boldsymbol{\alpha}_k)_0$ and $\hat{x} = (\boldsymbol{\alpha}_k^{(t)})_0$ and (44) with $x = \boldsymbol{\alpha}_{kj}$ and $\hat{x} = \boldsymbol{\alpha}_{kj}^{(t)}$. Plugging these into (42) and collecting everything that does not depend on $\mathbf{A}$ into the D' term we obtain

$$H(\mathbf{A}) \geq \sum_{k=1}^{K}\sum_{i=1}^{M}\omega_k^i\left((1-(\boldsymbol{\alpha}_k)_0)b_{ki} + \sum_{j=1}^{C}a_{kij}\log\boldsymbol{\alpha}_{kj}\right) + D' = H'(\mathbf{A}). \tag{45}$$

Now if we maximize H' and set $\mathbf{A}^{(t+1)} = \text{argmax}_{\mathbf{A}}\, H'(\mathbf{A})$ and use the fact that the bounds are tight at $\mathbf{A} = \mathbf{A}^{(t)}$ we obtain

$$H(\mathbf{A}^{(t)}) = H'(\mathbf{A}^{(t)}) \leq H'(\mathbf{A}^{(t+1)}) \leq H(\mathbf{A}^{(t+1)}) \tag{46}$$

as was our initial goal. So all that remains is to maximize $H'(\mathbf{A})$. Taking partial derivatives w.r.t $\boldsymbol{\alpha}_{kj}$ and setting to 0 we obtain

$$\sum_{i=1}^{M}\omega_k^i\left(-b_{ki} + a_{kij}\frac{1}{\boldsymbol{\alpha}_{kj}}\right) = 0 \tag{47}$$

hence

$$\boldsymbol{\alpha}_{kj} = \frac{\sum_{i=1}^{M}\omega_k^i a_{kij}}{\sum_{i=1}^{M}\omega_k^i b_{ki}}. \tag{48}$$

Therefore, we obtain the update rule

$$\boldsymbol{\alpha}_{kj}^{(t+1)} = \boldsymbol{\alpha}_{kj}^{(t)}\frac{\sum_{i=1}^{M}\omega_k^i\left(\psi(\mathbf{c}_{ij}+\boldsymbol{\alpha}_{kj}^{(t)}) - \psi(\boldsymbol{\alpha}_{kj}^{(t)})\right)}{\sum_{i=1}^{M}\omega_k^i\left(\psi(n_i+(\boldsymbol{\alpha}_k^{(t)})_0) - \psi((\boldsymbol{\alpha}_k^{(t)})_0)\right)}. \tag{49}$$

C. Experiment Details

C.1. Normalized MSE

In Section 4.1 we use normalized mean squared as a metric to measure how accurately we recover the ground truth MDM distribution parameters. Here we define normalized MSE as:

$$NMSE(\mathbf{x}, \mathbf{y}) \coloneqq \sqrt{||\frac{\mathbf{x}-\mathbf{y}}{\mathbf{y}}||^2} \tag{50}$$

where the division is entry-wise. Thus we are measuring the MSE but normalized by the size of $\mathbf{y}$, which in our case will be the ground truth parameters. This simply has the effect of ensuring that our metric is invariant to the size of the parameters we are trying to recover.

C.2. Folktables Dataset

We provide here details of the Folktables dataset used in our experimental evaluation. Folktables, (Ding et al., 2021), is a US census dataset from the year 2018 with a natural partitioning into heterogeneous federated clients. Each datapoint corresponds to an individual, with many features describing the individual, including age, gender, race, employment details, etc. The dataset comes with a number of different possible prediction tasks, such as predicting employment, commute time, health etc. given user features. Full details on the dataset can be found at the GitHub page: `https://github.com/socialfoundations/folktables/tree/main`.

The data contains a partitioning into federated clients based on location. Specifically, a feature of the data is the PUMA code, which is a code specifying the area the individual is registered to live in. Splitting the data based on PUMA code gives 2373 clients, each client holding the data of all individuals that live in the corresponding region. Due to the natural geographical population heterogeneity within the United States this partitioning leads to clients that are statistically heterogeneous in a multitude of factors. Figure 8 shows the histogram of the number of samples per client, which shows heterogeneity in the amount of data per client.

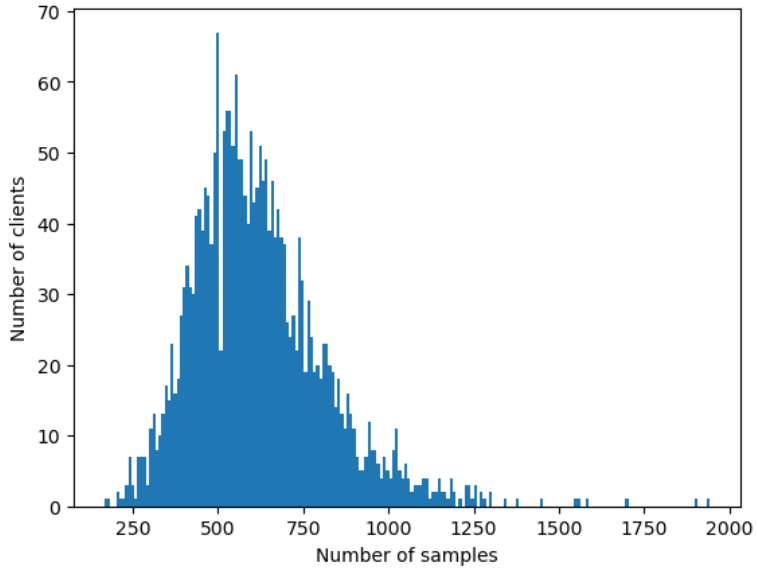


Figure 8. Histogram of the number of samples per client in the federated partition of the Folktables dataset.

Modeling the Race Feature In the Folktables dataset this feature is categorical with $C = 9$ different possible values. Given the sensitivity of this feature, as well as it's importance in many associated questions of fairness in downstream tasks, this is a good example of a feature one would want to ensure can be both privately learned and accurately represented in our server-side simulations.

Modeling the Income Feature As discussed in Section 4 income is a non-categorical feature of the data that describes an individual's annual income. These are real values, which in the 2018 census data range from 0 to 1423000. We bin the values using intervals of length 5000 up to 200000, with the final bin being all values greater than this. That is to say the bins are defined by the following 41 semi-closed intervals:

$$[0, 5000),\ [5000, 10000),\ \ldots\ ,\ [195000, 200000),\ [200000, \infty).$$

Thus our binned income is now a categorical feature with $C = 41$ possible values.

C.3. Hyperparameters for inference

Heterogeneity and number of components	Parameter Values
Low heterogeneity 1 mixture component	$\mathbf{A} = \begin{bmatrix} 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 1.0 \end{bmatrix}$
Low heterogeneity 2 mixture components	$\mathbf{A} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}$
Low heterogeneity 3 mixture components	$\mathbf{A} = \begin{bmatrix} 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 0.333 & 0.334 & 0.333 \end{bmatrix}$
Medium heterogeneity 1 mixture component	$\mathbf{A} = \begin{bmatrix} 0.1 & 0.2 & 0.6 & 1.0 & 2.0 & 0.1 & 1.0 & 2.0 & 0.5 & 0.5 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 1.0 \end{bmatrix}$
Medium heterogeneity 2 mixture components	$\mathbf{A} = \begin{bmatrix} 0.1 & 0.2 & 0.6 & 1.0 & 2.0 & 0.1 & 1.0 & 2.0 & 0.5 & 0.5 \\ 2.5 & 2.6 & 2.7 & 2.8 & 3.0 & 2.5 & 2.0 & 3.0 & 1.0 & 0.9 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 0.4 & 0.6 \end{bmatrix}$
Medium heterogeneity 3 mixture components	$\mathbf{A} = \begin{bmatrix} 5.0 & 4.0 & 5.0 & 1.0 & 1.0 & 1.0 & 5.0 & 4.0 & 5.0 & 1.0 \\ 0.1 & 0.2 & 0.6 & 1.0 & 2.0 & 0.1 & 1.0 & 2.0 & 0.5 & 0.5 \\ 2.5 & 2.6 & 2.7 & 2.8 & 3.0 & 2.5 & 2.0 & 3.0 & 1.0 & 0.9 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 0.5 & 0.2 & 0.3 \end{bmatrix}$
High heterogeneity 1 mixture component	$\mathbf{A} = \begin{bmatrix} 0.1 & 0.2 & 0.15 & 0.18 & 0.1 & 0.05 & 0.08 & 0.4 & 0.2 & 0.12 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 1.0 \end{bmatrix}$
High heterogeneity 2 mixture components	$\mathbf{A} = \begin{bmatrix} 0.1 & 0.2 & 0.15 & 0.18 & 0.1 & 0.05 & 0.08 & 0.4 & 0.2 & 0.12 \\ 2.5 & 2.6 & 2.0 & 3.2 & 1.5 & 0.9 & 0.8 & 1.3 & 3.1 & 2.4 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 0.1 & 0.9 \end{bmatrix}$
High heterogeneity 3 mixture components	$\mathbf{A} = \begin{bmatrix} 5.0 & 5.0 & 0.2 & 0.2 & 3.1 & 3.0 & 3.2 & 0.8 & 0.9 & 5.0 \\ 0.1 & 0.2 & 0.15 & 0.18 & 0.1 & 0.05 & 0.08 & 0.4 & 0.2 & 0.12 \\ 2.5 & 2.6 & 2.0 & 3.2 & 1.5 & 0.9 & 0.8 & 1.3 & 3.1 & 2.4 \end{bmatrix}$, $\boldsymbol{\tau} = \begin{bmatrix} 0.8 & 0.05 & 0.15 \end{bmatrix}$

Table 1. Mixture-of-Dirichlet-Multinomial distribution parameter values.

In Section 4.1 we evaluate how well Algorithms 1 and 2 are able to recover the ground truth distribution parameters, when they exist, and infer meaningful values for the parameters when run on real federated data.

For the synthetic data which follows a MDM distribution we ran experiments using 1, 2 or 3 ground truth mixture components with three different settings of ground truth distribution parameters in each case. These settings corresponded to low, medium and high levels of client statistical heterogeneity. The exact values are given in Table 1. Algorithm 1 was run using a client cohort size of 1000 followed by Algorithm 2 which was run for 100 global rounds with a client cohort size of 1000.

For inference on FEMNIST we there are no ground truth distribution parameters for us to set, we use the existing partition of the dataset into the true clients for inference. We run inference using 2 and 3 mixture components. In both cases we run algorithm 1 using a client cohort size of 3400 followed by Algorithm 2 for 50 global rounds with a client cohort size of 3400.

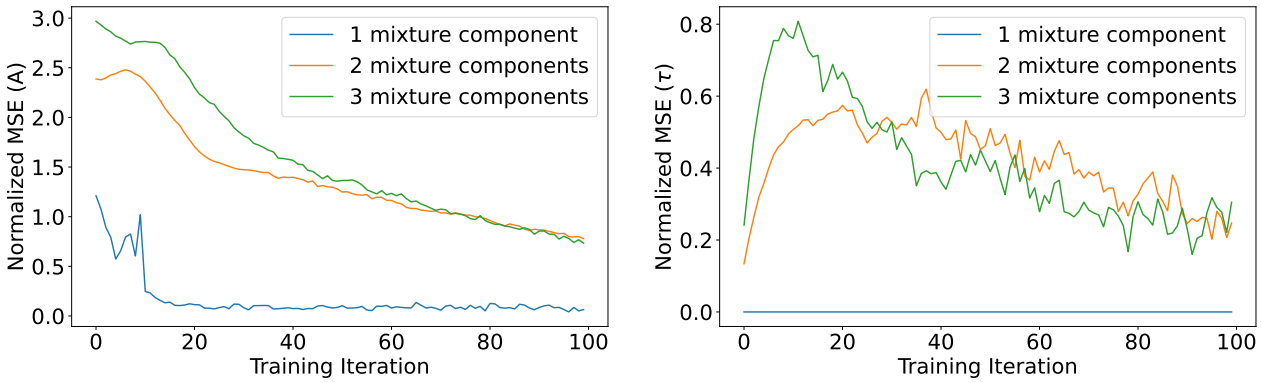


Figure 9. Normalized mean squared error (MSE) between the ground truth distribution parameter value and the inferred parameter value over time. Ground truth corresponds to low levels of client statistical heterogeneity. On the left for $\mathbf{A}$, on the right for $\boldsymbol{\tau}$.

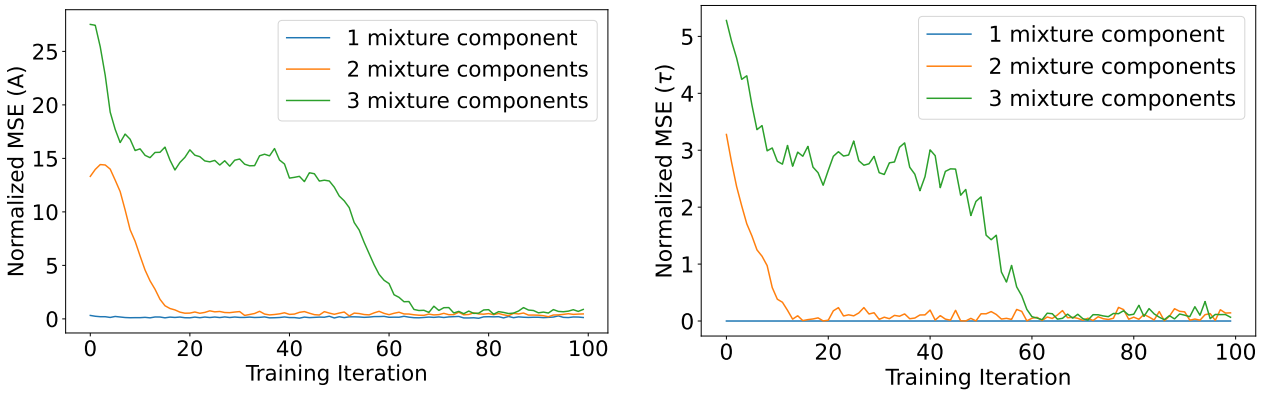


Figure 10. Normalized mean squared error (MSE) between the ground truth distribution parameter value and the inferred parameter value over time. Ground truth corresponds to high levels of client statistical heterogeneity. On the left for $\mathbf{A}$, on the right for $\boldsymbol{\tau}$.

C.4. Hyperparameters for model training

In Section 4.2 we train a model using Federated Averaging on the true clients and on various types of simulated clients over both CIFAR10 and FEMNIST. For both datasets the model used is a CNN with 2 convolutional layers, one dense hidden layer and ReLU activations. In our experiments we compare the performance on the true clients against the simulated clients while varying the certain important hyperparameters.

For CIFAR10 we vary the local batch size over $[10, 15, 20, 25]$, the local number of epochs over $[1, 2, 5, 10]$ and the local learning rate over $[0.005, 0.01, 0.05, 0.1, 0.5]$. We report over all combinations of these HPs. The remaining hyperparameters are fixed and equal across true client and all simulated client training. Global learning rate for FedAvg is 1.0, client cohort size is 50, and the number of global training rounds is 1500.

For FEMNIST we vary the local number of epochs over $[1, 2, 5, 10]$ and the local learning rate over $[0.005, 0.01, 0.05]$. We report over all combinations of these HPs. The remaining hyperparameters are fixed and equal across true client and all simulated client training. Global learning rate for FedAvg is 1.0, client cohort size is 50, the number of global training rounds is 1500 and the local batch size is 10.

D. Additional Experiments

Here we include additional experiments and figures relating to the empirical evaluation in Section 4.

Figure 11. t-SNE visualisation of FEMNIST clients, each point corresponds to a single client's class histogram. True clients (green), fully IID simulated clients (blue) and MDM clients (red). On the left we infer using 2 mixture components and on the right we use 3 components.

D.1. Distribution Parameter Inference

Inference with known parameters In Section 4.1 we first investigated how well we are able to recover ground truth parameters when the clients follow the assumed MDM distribution. We reported the MSE over time for clients corresponding the medium levels of heterogeneity. Figures 9 and 10 show the results for when running inference on clients with low and high levels of client statistical heterogeneity respectively. Recall the exact values are given in Table 1.

Inference without known parameters In Section 4.1 we additionally evaluated inferring the MDM distribution on federated datasets with a natural partitioning into federated clients. For FEMNIST we showed results when inferring using 3 mixture components. Here we show results when inferring with different numbers of components. The experimental setup is exactly as described in Section 4.1. We plot t-SNE visualisations of the simulated clients when we infer using both 2 and 3 mixture components. The results are shown in Figure 11. We see that in both cases the inferred MDM distribution does a superior job of capturing the class heterogeneity of the true clients compared to the fully IID baseline. We see in the case of 2 components that the inference is dominated by the large left and lower right clusters of true clients while adding the flexibility of an extra component gives better coverage of the small upper right hand cluster. For the Folktables dataset we also infer over a range of different values of K, namely $K = 1, 3, 5, 7$. The results when modeling the race feature are shown in Figure 12. As we can see in all cases the MDM distribution is able to well model the true federated clients while the fully IID baseline performs poorly. We observe qualitatively that $K \geq 3$ leads to a better simulation of the true federated clients, with more complete coverage of the tails. The results for modelling the binned income feature are shown in Figure 13. As we can see, with the exception of $K = 1$, the simulated MDM clients exhibit strong similarity to the true clients with larger values of K being better. It is interesting to observe that for the case $K = 1$ the MDM clients fail quite badly at simulating the heterogeneity of the true clients, again confirming the importance of the mixture model beyond just using a single Dirichlet-multinomial.

D.2. Federated Training Simulations

We provide here additional experiments when running training simulations on MDM simulated clients as described in Section 4.2.

Disjoint Server and Client Data We provide here additional experiments in the setting that the server-side data and client data are different and non-overlapping. We do this in the setting of the hyperparameter sweep experiments described in Section 4.2 of the paper. We again use the FEMNIST dataset. Previously our server side data was the whole FEMNIST dataset, i.e. data of all clients shuffled together with client identifiers removed. We now create disjoint server and client datasets by assigning roughly half of the data to the server and leaving the other half as our true clients. Concretely we do this split as follows:

We take the first 1302 FEMNIST clients (in the original ordering of clients in the dataset). This corresponds to roughly half of the data, and we leave these clients unchanged. These are our true federated clients. The data of the remaining clients, client indices [1302:], is shuffled together into a single dataset and is used as our server-side data.

This splitting introduces a domain shift between the server data and the true client data. That is because the clients in FEMNIST are not randomly ordered and there is in fact a difference between the first roughly 1300 clients and the remaining clients (both in terms of the number of samples they possess and the statistical heterogeneity of the clients). We refer you to figure 3 in the paper, this difference is in fact exactly shown by the left and right clusters of the true clients (green) in the t-SNE visualisation.

We now proceed identically to the experimental evaluation of Section 4.2. We infer our MDM distribution parameters using Algorithms 1 and 2 on the true clients. We use these inferred parameters to partition the server-side data into simulated MDM clients. We then train on these simulated clients using a range of different HP settings. We compare the obtained accuracy to the accuracy of training on the true clients as well as our Fully IID and Conditionally IID simulated client baselines. The results are shown in Figure 14. As we can see there is a much greater similarity between the MDM simulations and true clients than between the IID simulations and the true clients.

CIFAR10 with MDM partitioning Here we include the additional results for training on MDM partitioned clients of CIFAR10. There correspond to the settings of the parameters given in Table 1. In Section 4.1 we showed results for 2 components and high heterogeneity. The results for the remaining settings are shown in Figures 15 - 22. In general these results exhibit very similar properties to what we observed in Section 4.1. We do see in the cases of Medium and High heterogeneity with 3 components some slightly larger deviations between the MDM simulations, and the true clients, although they are still very similar. This is a reflection of the fact that the parameters we inferred in Section 4.1 for these settings differed more than for the other settings.

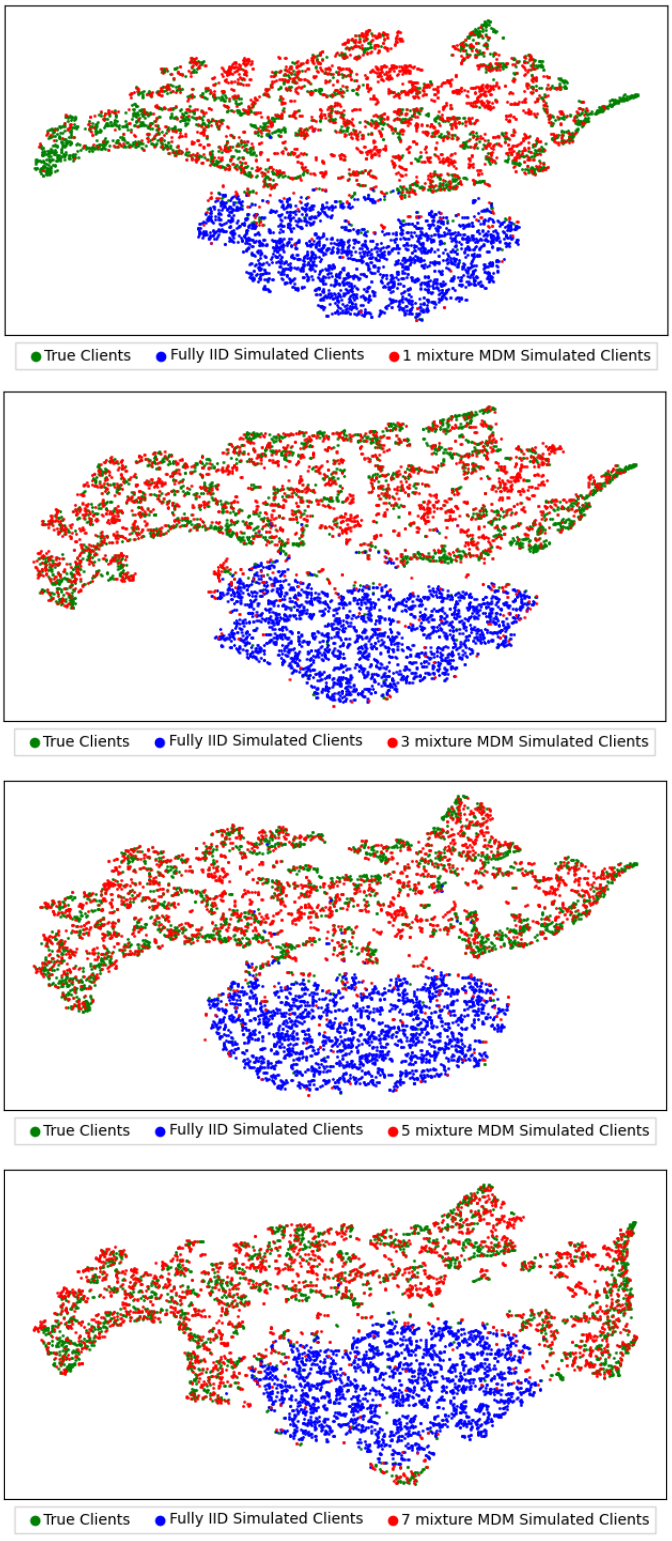
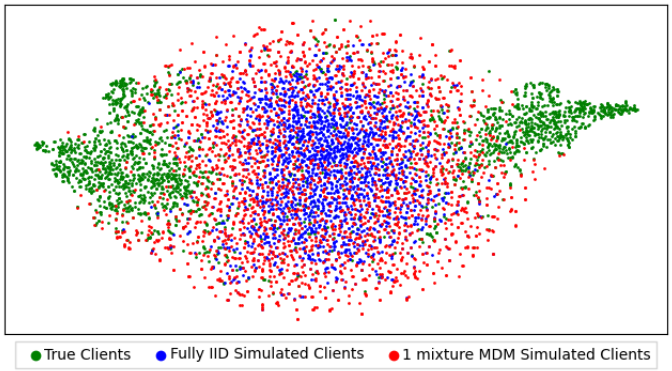
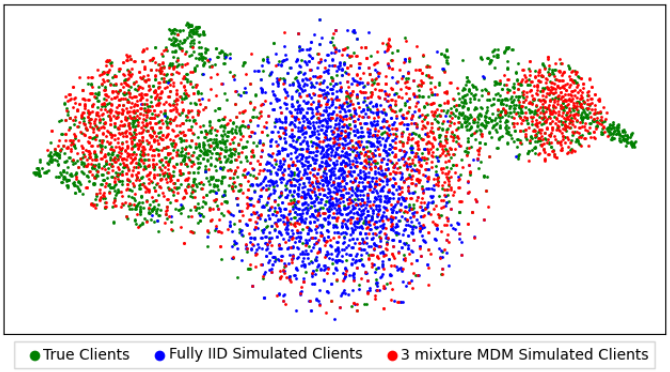
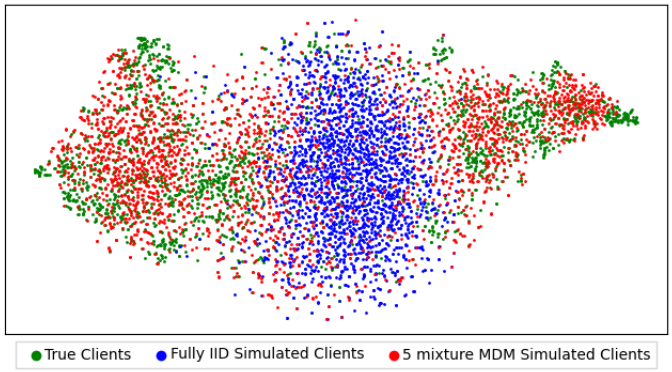


Figure 12. t-SNE visualisations of Folktables clients, each point corresponds to a single client's histogram of the race feature. True clients (green), fully IID simulated clients (blue) and MDM clients (red). Inferred using (from top to bottom) $K = 1, 3, 5, 7$.







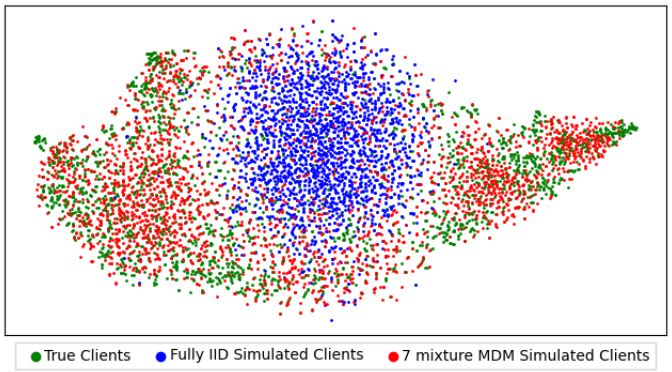


Figure 13. t-SNE visualisations of Folktables clients, each point corresponds to a single client's histogram of the binned income feature. True clients (green), fully IID simulated clients (blue) and MDM clients (red). Inferred using (from top to bottom) $K = 1, 3, 5, 7$.

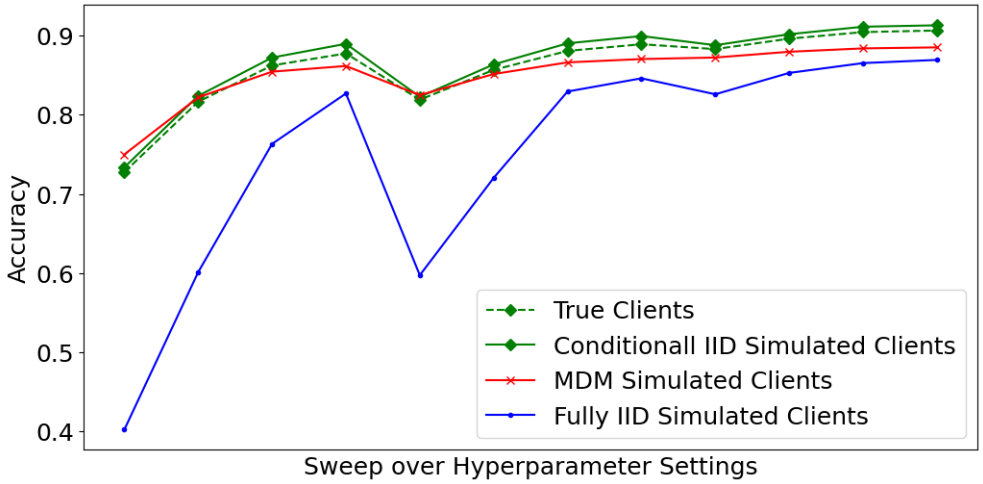


Figure 14. FEMNIST test accuracy when training with FedAvg for different settings of local learning rate and local epochs. True clients (dotted green), conditionally IID simulated clients (green), learned MDM simulated clients (red) and fully IID simulated clients (blue).

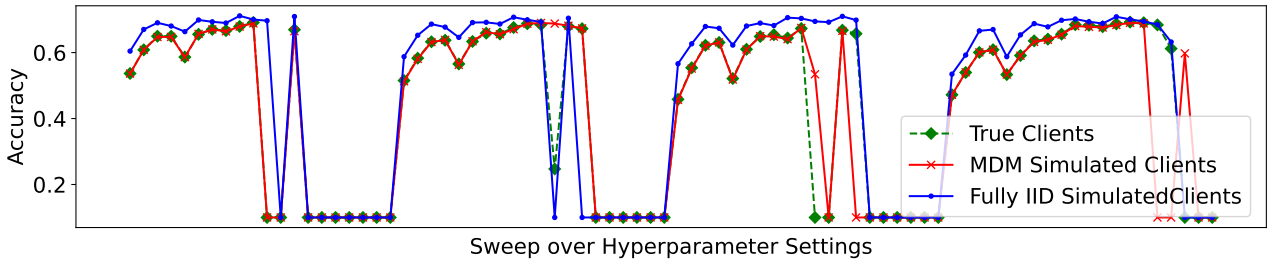


Figure 15. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue). Ground truth: Low Heterogeneity and 1 mixture component.

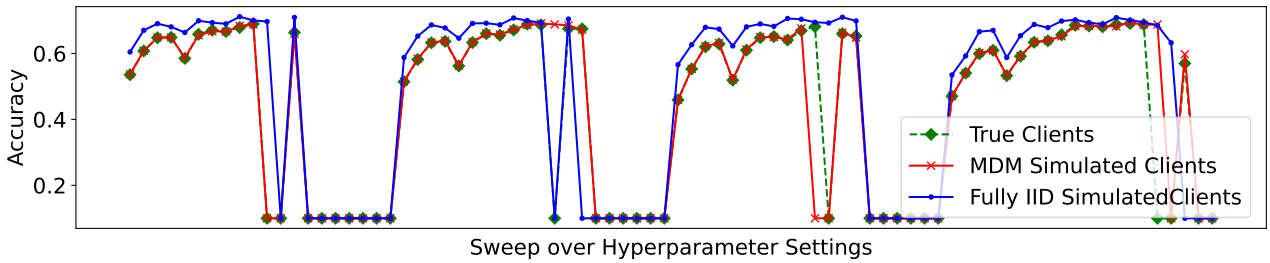


Figure 16. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue). Ground truth: Low Heterogeneity and 2 mixture component.

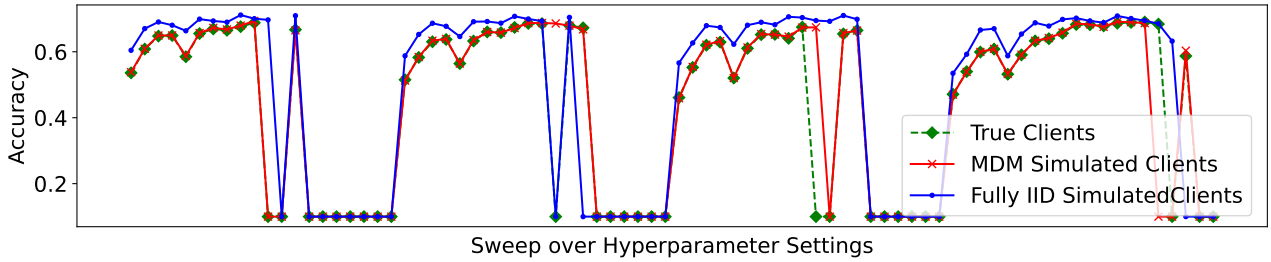


Figure 17. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue). Ground truth: Low Heterogeneity and 3 mixture component.

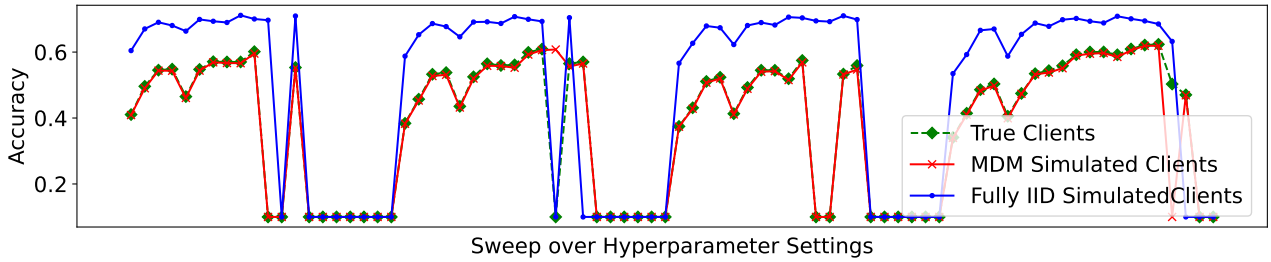


Figure 18. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue). Ground truth: Medium Heterogeneity and 1 mixture component.

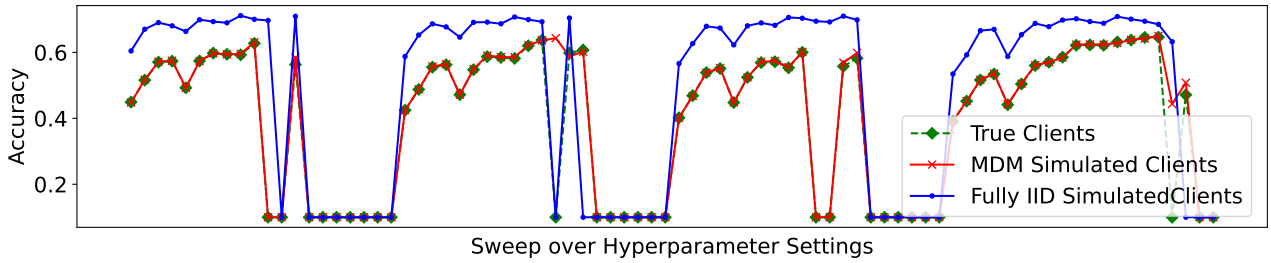


Figure 19. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue). Ground truth: Medium Heterogeneity and 2 mixture component.

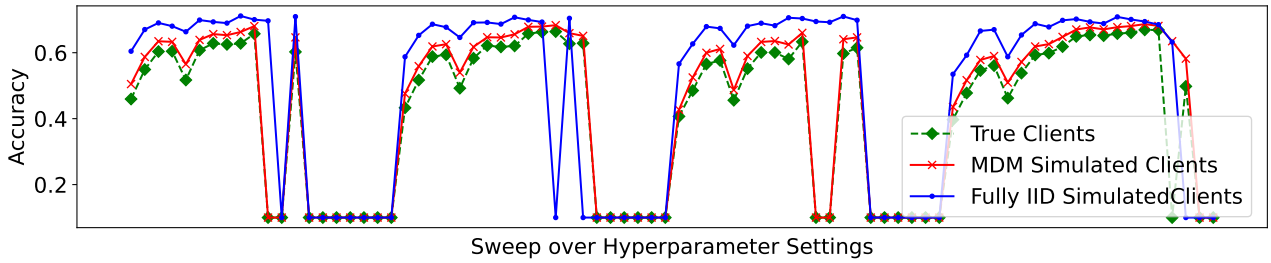


Figure 20. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue). Ground truth: Medium Heterogeneity and 3 mixture component.

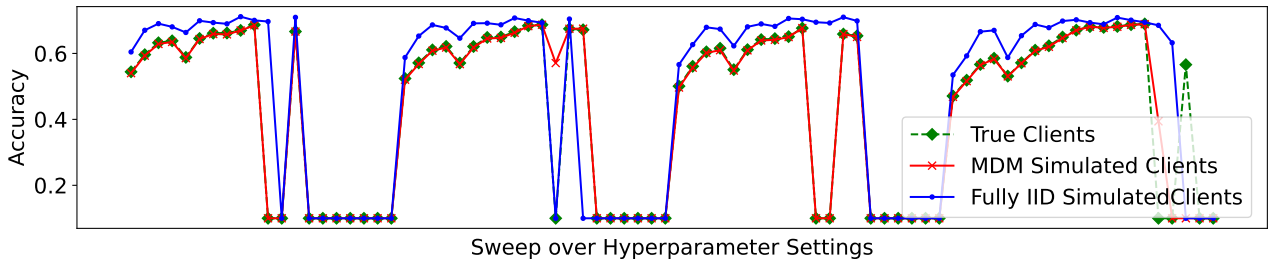


Figure 21. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue). Ground truth: High Heterogeneity and 1 mixture component.

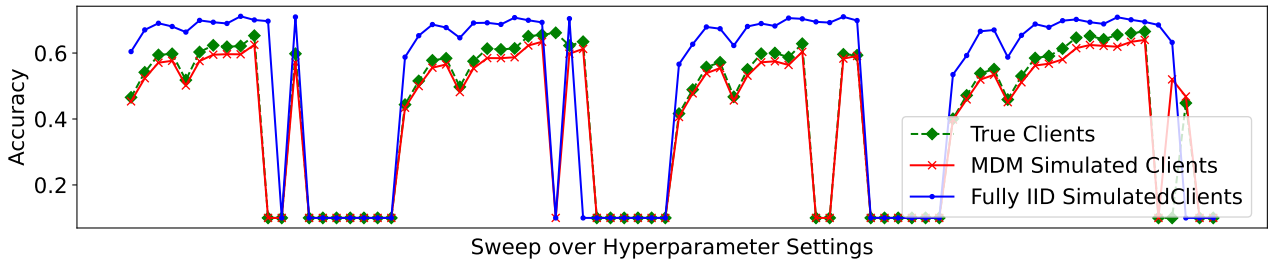


Figure 22. CIFAR10 test accuracy when training with FedAvg for different settings of local batch size, local learning rate and local epochs. True clients (dotted green), learned MDM simulated clients (red) and fully IID simulated clients (blue). Ground truth: High Heterogeneity and 3 mixture component.