Efficient Clustering Frameworks for Federated Learning Systems

Jichan Chung
Avishek Ghosh
Dong Yin
Kangwook Lee
Kannan Ramchandran

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-212
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-212.html

August 11, 2023

Efficient Clustering Frameworks for Federated Learning Systems
by Jichan Chung

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Kannan Ramchandran
Research Advisor

2023.8.11
(Date)

* * * * * * *

Irene Chen

Professor Irene Y. Chen
Second Reader

2023.8.11
(Date)

Efficient Clustering Frameworks for Federated Learning Systems

by

Jichan Chung

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Spring 2023

Efficient Clustering Frameworks for Federated Learning Systems

Abstract

Efficient Clustering Frameworks for Federated Learning Systems

by

Jichan Chung

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Kannan Ramchandran, Chair

We address the problem of Federated Learning (FL) where users are distributed and their datapoints are partitioned into clusters. This setup captures settings where users have their own objectives (learning tasks) but by aggregating their data with others in the same cluster (same learning task), they can leverage the strength in numbers in order to perform more efficient Federated Learning. We propose a framework dubbed the Iterative Federated Clustering Algorithm (IFCA), which alternately estimates the cluster identities of the users and optimizes model parameters for the user clusters via gradient descent. We analyze the convergence rate of this algorithm first in a linear model with squared loss and then for generic strongly convex and smooth loss functions. We show that in both settings, with good initialization, IFCA converges at an exponential rate, and discuss the optimality of the statistical error rate. When the clustering structure is ambiguous, we propose to train the models by combining IFCA with the weight sharing technique in multi-task learning. In the experiments, we show that our algorithm can succeed even if we relax the requirements on initialization with random initialization and multiple restarts. We also present experimental results showing that our algorithm is efficient in non-convex problems such as neural networks. We demonstrate the benefits of IFCA over the baselines on several clustered FL benchmarks. We also develop an extension of our framework for a more general setting where statistical heterogeneity can exist across clients, named UIFCA . For synthetic data, we observe that UIFCA can correctly recover the cluster information of individual datapoints. We also provide analysis of UIFCA on MNIST dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In many modern data-intensive applications such as recommendation systems, image recognition, and conversational AI, distributed computing has become a crucial component. In many applications, data are stored in end users' own devices such as mobile phones and personal computers, and in these applications, fully utilizing the on-device machine intelligence is an important direction for next-generation distributed learning. Federated Learning (FL) [40, 25, 39] is a recently proposed distributed computing paradigm that is designed towards this goal, and has received significant attention. Many statistical and computational challenges arise in Federated Learning, due to the highly decentralized system architecture. In this report, we propose an efficient algorithm that aims to address one of the major challenges in FL—dealing with heterogeneity in the data distribution.

In Federated Learning, since the data source and computing nodes are end users' personal devices, the issue of data heterogeneity, also known as non-i.i.d. data, naturally arises. Exploiting data heterogeneity is particularly crucial in applications such as recommendation systems and personalized advertisement placement, and it benefits both the users' and the enterprises. For example, mobile phone users who read news articles may be interested in different categories of news like politics, sports or fashion; advertisement platforms might need to send different categories of ads to different groups of customers. These indicate that leveraging the heterogeneity among the users is of potential interest—on the one hand, each machine itself may not have enough data and thus we need to better utilize the similarity among the users; on the other hand, if we treat the data from all the users as i.i.d. samples, we may not be able to provide personalized predictions. This problem has recently received much attention [51, 49, 21].

In this report, we study two different formulations of FL with non-i.i.d. data. First, we consider *clustered Federated Learning* [49, 38] setup, which assumes that the users are partitioned into different clusters; for example, the clusters may represent groups of users interested in politics, sports, etc, and our goal is to train models for every cluster of users. In order to achieve this goal, we propose a framework and analyze a distributed method, named the *Iterative Federated Clustering Algorithm (IFCA)* for clustered FL. We further establish convergence rates of our algorithm, for both linear models and general strongly convex losses under the assumption of good initialization. We prove exponential convergence speed, and for both settings, we can obtain *near optimal* statistical error rates in certain regimes. We also present experimental evidence of its performance in practical settings: We show that our algorithm can succeed even if we relax the initialization requirements with random initialization and multiple restarts; and we also present results showing that our algorithm is efficient on neural networks. We demonstrate the effectiveness of IFCA on two clustered FL benchmarks created based on the MNIST and CIFAR-10 datasets, respectively, as well as the Federated EMNIST dataset [2] which is a more realistic benchmark for FL and has ambiguous cluster structure.

In the second part of the study, we extend our algorithm to be applicable for a more general type of heterogeniety and dataset; We now assume that a user can hold data from multiple clusters, without any information of its membership, and task-specific label is not available. An example case would be a user reading articles from different topics, but the articles may not have tags that indicate which subjects it belongs to. We develop UIFCA , a generative model-based clustering method for *unsupervised* dataset, that decodes the *heterogeniety within the client*, based on the IFCA's approach. We provide a comparison of UIFCA and k-FED algorithm [8](an off-the-shelf federated clustering approach), on several different types of synthetic cluster-structured data. We also evaluate our algorithm on MNIST dataset, and provide comparison against ClusterGAN [43], a popular recent approach to unsupervised data in the centralized setting combined with FedAvg algorithm. For all cases, we evaluate for a particular type of client heterogeneity where a constant portion of the client's data belongs to a single distribution, and the remaining portion are drawn from the mixture of all distributions. For the synthetic datasets, we observe that our method can correctly recover cluster information for both i.i.d and non-i.i.d. cases, while the baseline fails in non-i.i.d. We also provide analysis of our method on MNIST dataset.

The rest of this report discusses the development of IFCA and UIFCA . The outline is as follows. In Chapter 2, we survey existing works related to our problem along with the potential drawbacks and for improvement in these works. Chapter 3 describes the IFCA algorithm in detail, with its analysis and experimental results. Chapter 4 presents the UIFCA algorithm that extends IFCA for user-level heterogeniety and unsupervised datasets, with its experimental results. Lastly, Chapter 5 draws conclusions from our work and describes directions for future work.

1.1 Notation

We use $[r]$ to denote the set of integers $\{1, 2, \dots, r\}$. We use $\|\cdot\|$ to denote the ℓ_2 norm of vectors. We use $x \gtrsim y$ if there exists a sufficiently large constant $c > 0$ such that $x \geq cy$, and define $x \lesssim y$ similarly. We use $\text{poly}(m)$ to denote a polynomial in m with arbitrarily large constant degree.

Chapter 2

Related Work

During the preparation of the initial draft, we became aware of a concurrent and independent work by Mansour et al. [38], in which the authors propose clustered FL as one of the formulations for personalization in Federated Learning. The algorithms proposed in our report and by Mansour et al. are similar. However, our report makes an important contribution by establishing the *convergence rate* of the *population loss function* under good initialization, which simultaneously guarantees both convergence of the training loss and generalization to test data; whereas in [38], the authors provided only *generalization* guarantees. We discuss other related work in the following.

2.1 Federated Learning and non-i.i.d. data

Learning with a distributed computing framework has been studied extensively in various settings [65, 46, 32]. As mentioned in Chpater 1, Federated Learning [40, 39, 25, 19] is one of the modern distributed learning frameworks that aims to better utilize the data and computing power on edge devices. A central problem in FL is that the data on the users' personal devices are usually non-i.i.d. Several formulations and solutions have been proposed to tackle this problem. A line of research focuses on learning a single global model from non-i.i.d. data [64, 48, 31, 50, 34, 42]. Other lines of research focus more on learning personalized models [51, 49, 12]. In particular, the MOCHA algorithm [51] considers a multi-task learning setting and forms a deterministic optimization problem with the correlation matrix of the users being a regularization term. Our work differs from MOCHA since we consider a statistical setting with cluster structure. Another approach is to formulate Federated Learning with non-i.i.d. data as a meta learning problem [5, 21, 12]. In this setup, the objective is to first obtain a single global model, and then each device fine-tunes the model using its local data. The underlying assumption of this formulation is that the data distributions among different users are similar, and the global model can serve as a good initialization. The formulation of clustered FL has been considered in two recent works [49, 16]. Both of the two works use *centralized* clustering algorithm such as K-means, in which the center machine has to identify the cluster identities of all the users, leading to high computational cost at the center. As a result, these algorithms may not be suitable for large models such as deep neural networks or applications with a large number of users. For the *decentralized* approach, k-FED [8] proposes a clustering algorithm that is a variant of K-means [36], with focus on reducing the number communication rounds.

2.2 Latent variable problems

As mentioned in Chapter 1, our formulation can be considered as a statistical estimation problem with latent variables in a distributed setting, and the latent variables are the cluster identities. The latent variable problem is a classical topic in statistics and non-convex optimization; examples include Gaussian mixture models (GMM) [58, 30], mixture of linear regressions [9, 55, 62], and phase retrieval [13, 41]. Expectation Maximization (EM) and Alternating Minimization (AM) are two popular approaches to solving these problems. Despite the wide applications, their convergence analyses in the finite sample setting are known to be hard, due to the non-convexity nature of their optimization landscape. In recent years, some progress has been made towards understanding the convergence of EM and AM in the centralized setting [44, 6, 61, 1, 54]. For example, if started from a suitable point, they have fast convergence rate, and occasionally they enjoy super-linear speed of convergence [58, 14]. In this report, we provide new insights to these algorithms in the FL setting.

2.3 Clustering methods using deep neural network for the centralized environment

Many attempts have been made in using deep neural network to the clustering problem for the centralized environment. ClusterGAN [43] designs a new GAN architecture that learns representations that forms clusters in latent space. A variant of Variational Autoencoder model was proposed to capture the cluster informations [10]. Self-conditioned GAN [35] trains GAN model conditioned by pseudo-labels, where pseudo-labels are iteratively assigned from K-means clustering on the learned representation. DeepCluster [3] trains deep classifiers with similar idea. These works assumes full access to training data that captures the distribution they come from. We consider the same problem for the federated, where above assumption is not available. We find that methods involving DNN to cluster a unsupervised dataset are not well studied in the federated settings, therefore we analyze DNN-based methods for centralized setting in the federated settings, as well as applying IFCA algorithm to solve the same problem.

Chapter 3

Iterative Federated Clustering Algorithm (IFCA)

3.1 Problem Formulation

We begin with a standard statistical learning setting of empirical risk minimization (ERM). Our goal is to learn parametric models by minimizing some loss functions defined by the data. We consider a distributed learning setting where we have one center machine and m worker machines (i.e., each worker machine corresponds to a user in the Federated Learning framework). The center machine and worker machines can communicate with each other using some predefined communication protocol. We assume that there are k different data distributions, $\mathcal{D}_1, \dots, \mathcal{D}_k$, and that the m machines are partitioned into k disjoint clusters, $S_1^*, \dots, S_k^*$. We assume no knowledge of the cluster identity of each machine, i.e., the partition $S_1^*, \dots, S_k^*$ is not revealed to the learning algorithm. We assume that every worker machine $i \in S_j^*$ contains n i.i.d. data points $z^{i,1}, \dots, z^{i,n}$ drawn from $\mathcal{D}_j$, where each data point $z^{i,j}$ consists of a pair of feature and response denoted by $z^{i,\ell} = (x^{i,\ell}, y^{i,\ell})$.

Let $f(\theta; z) : \Theta \to \mathbb{R}$ be the loss function associated with data point z, where $\Theta \subseteq \mathbb{R}^d$ is the parameter space. In this paper, we choose $\Theta = \mathbb{R}^d$. Our goal is to minimize the population loss function $F^j(\theta) := \mathbb{E}_{z\sim\mathcal{D}_j}[f(\theta; z)]$ for all $j \in [k]$. For the purpose of theoretical analysis in Section 3.3, we focus on the strongly convex losses, in which case we can prove guarantees for estimating the unique solution that minimizes each population loss function. In particular, we try to find solutions $\{\widehat{\theta}_j\}_{j=1}^k$ that are close to $\theta_j^* = \text{argmin}_{\theta\in\Theta} F^j(\theta)$, $j \in [k]$. In our problem, since we only have access to finite data, we take advantage of the empirical loss functions. In particular, let $Z \subseteq \{z^{i,1}, \dots, z^{i,n}\}$ be a subset of the data points on the i-th machine. We define the empirical loss associated with Z as $F_i(\theta; Z) = \frac{1}{|Z|} \sum_{z\in Z} f(\theta; z)$. When it is clear from the context, we may also use the shorthand notation $F_i(\theta)$ to denote an empirical loss associated with some (or all) data on the i-th worker.

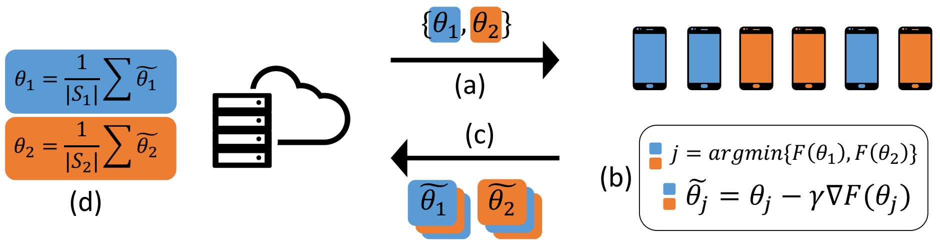

$\theta_1 = \frac{1}{|S_1|}\sum \widetilde{\theta_1}$
$\theta_2 = \frac{1}{|S_2|}\sum \widetilde{\theta_2}$
(d)
$\{\theta_1, \theta_2\}$
(a)
(c)
$\widetilde{\theta_1}$
$\widetilde{\theta_2}$
(b)
$j = argmin\{F(\theta_1), F(\theta_2)\}$
$\widetilde{\theta_j} = \theta_j - \gamma\nabla F(\theta_j)$

Figure 3.1: An overview of IFCA (model averaging). (a) The server broadcast models. (b) Worker machines identify their cluster memberships and run local updates. (c) The worker machines send back the local models to server. (d) Average the models within the same estimated cluster S_j.

3.2 Algorithm

In this section, we provide details of our algorithm. We name this scheme *Iterative Federated Clustering Algorithm* (IFCA). The main idea is to alternatively minimize the loss functions while estimating the cluster identities. We discuss two variations of IFCA, namely gradient averaging and model averaging. The algorithm is formally presented in Algorithm 1 and illustrated in Figure 4.2.

The algorithm starts with k initial model parameters $\theta_j^{(0)}$, $j \in [k]$. In the t-th iteration of IFCA, the center machine selects a random subset of worker machines, $M_t \subseteq [m]$, and broadcasts the current model parameters $\{\theta_j^{(t)}\}_{j=1}^k$ to the worker machines in M_t. Here, we call M_t the set of *participating devices*. Recall that each worker machine is equipped with local empirical loss function $F_i(\cdot)$. Using the received parameter estimates and F_i, the i-th worker machine ($i \in M_t$) estimates its cluster identity via finding the model parameter with lowest loss, i.e., $\widehat{j} = \text{argmin}_{j\in[k]} F_i(\theta_j^{(t)})$ (ties can be broken arbitrarily). If we choose the option of gradient averaging, the worker machine then computes a (stochastic) gradient of the local empirical loss F_i at $\theta_{\widehat{j}}^{(t)}$, and sends its cluster identity estimate and gradient back to the center machine. After receiving the gradients and cluster identity estimates from all the participating worker machines, the center machine then collects all the gradient updates from worker machines whose cluster identity estimates are the same and conducts gradient descent update on the model parameter of the corresponding cluster. If we choose the option of model averaging (similar to the Federated Averaging algorithm [39]), each participating device needs to run τ steps of local (stochastic) gradient descent updates, get the updated model, and send the new model and its cluster identity estimate to the center machine. The center machine then averages the new models from the worker machines whose cluster identity estimates are the same.

Algorithm 1 Iterative Federated Clustering Algorithm (IFCA)

```
1: Input: number of clusters $k$, step size $\gamma$, $j \in [k]$, initialization $\theta_j^{(0)}$, $j \in [k]$
   number of parallel iterations $T$, number of local gradient steps $\tau$ (for model averaging).
2: for $t = 0, 1, \dots, T-1$ do
3:   center machine: broadcast $\theta_j^{(t)}$, $j \in [k]$
4:   $M_t \leftarrow$ random subset of worker machines (participating devices)
5:   for worker machine $i \in M_t$ in parallel do
6:     cluster identity estimate $\widehat{j} = \mathrm{argmin}_{j\in[k]} F_i(\theta_j^{(t)})$
7:     define one-hot encoding vector $s_i = \{s_{i,j}\}_{j=1}^k$ with $s_{i,j} = \mathbf{1}\{j = \widehat{j}\}$
8:     option I (gradient averaging):
9:       compute (stochastic) gradient: $g_i = \widehat{\nabla} F_i(\theta_{\widehat{j}}^{(t)})$, send back $s_i$, $g_i$ to the center
       machine
10:    option II (model averaging):
11:      $\widetilde{\theta}_i = \mathsf{LocalUpdate}(\theta_{\widehat{j}}^{(t)}, \gamma, \tau)$, send back $s_i$, $\widetilde{\theta}_i$ to the center machine
12:  center machine:
13:  option I (gradient averaging): $\theta_j^{(t+1)} = \theta_j^{(t)} - \frac{\gamma}{m} \sum_{i\in M_t} s_{i,j} g_i$, $\forall\ j \in [k]$
14:  option II (model averaging): $\theta_j^{(t+1)} = \sum_{i\in M_t} s_{i,j}\widetilde{\theta}_i / \sum_{i\in M_t} s_{i,j}$, $\forall\ j \in [k]$
15: return $\theta_j^{(T)}$, $j \in [k]$
    $\mathsf{LocalUpdate}(\widetilde{\theta}^{(0)}, \gamma, \tau)$ at the $i$-th worker machine
16: for $q = 0, \dots, \tau - 1$ do
17:   (stochastic) gradient descent $\widetilde{\theta}^{(q+1)} = \widetilde{\theta}^{(q)} - \gamma \widehat{\nabla} F_i(\widetilde{\theta}^{(q)})$
18: return $\widetilde{\theta}^{(\tau)}$
```

Practical implementation of IFCA

We clarify a few issues regarding the practical implementation of IFCA. In some real-world problems, the cluster structure may be ambiguous, which means that although the distributions of data from different clusters are different, there exists some common properties of the data from all the users that the model should leverage. For these problems, we propose to use the weight sharing technique in multi-task learning [4] and combine it with IFCA. More specifically, when we train neural network models, we can share the weights for the first a few layers among all the clusters so that we can learn a good representation using all the available data, and then run IFCA algorithm only on the last (or last few) layers to address the different distributions among different clusters. Using the notation in Algorithm 1, we run IFCA on a subset of the coordinates of $\theta_j^{(t)}$, and run vanilla gradient averaging or Federated Averaging on the remaining coordinates. Another benefit of this implementation is that we can reduce the communication cost: Instead of sending k models to all the worker machines, the center machine only needs to send k different versions of a subset of all the weights, and one single copy of the shared layers.

Another technique to reduce communication cost is that when the center machine observes that the cluster identities of all the worker machines are stable, i.e., the estimates of their cluster identities do not change for several parallel iterations, then the center machine can stop sending k models to each worker machine, and instead, it can simply send the model corresponding to each worker machine's cluster identity estimate.

3.3 Theoretical Guarantees

In this section, we present convergence guarantees of IFCA. In order to streamline our theoretical analysis, we make several simplifications: we consider the IFCA with gradient averaging, and assume that all the worker machines participate in every rounds of IFCA, i.e., $M_t = [m]$ for all t. In addition, we also use the *re-sampling* technique for the purpose of theoretical analysis. In particular, suppose that we run a total of T parallel iterations. We partition the n data points on each machine into $2T$ disjoint subsets, each with $n' = \frac{n}{2T}$ data points. For the i-th machine, we denote the subsets as $\widehat{Z}_i^{(0)}, \dots, \widehat{Z}_i^{(T-1)}$ and $Z_i^{(0)}, \dots, Z_i^{(T-1)}$. In the t-th iteration, we use $\widehat{Z}_i^{(t)}$ to estimate the cluster identity, and use $Z_i^{(t)}$ to conduct gradient descent. As we can see, we use fresh data samples for each iteration of the algorithm. Furthermore, in each iteration, we use different set of data points for obtaining the cluster estimate and computing the gradient. This is done in order to remove the inter-dependence between the cluster estimation and the gradient computation, and ensure that in each iteration, we use fresh i.i.d. data that are independent of the current model parameter. We would like to emphasize that re-sampling is a standard tool used in statistics [44, 20, 60, 61, 14], and that it is for theoretical tractability only and is not required in practice as we show in Section 3.4.

Under these conditions, the update rule for the parameter vector of the j-th cluster can be written as

$$S_j^{(t)} = \{i \in [m] : j = \mathrm{argmin}_{j' \in [k]} F_i(\theta_{j'}^{(t)}; \widehat{Z}_i^{(t)})\}, \quad \theta_j^{(t+1)} = \theta_j^{(t)} - \frac{\gamma}{m} \sum_{i \in S_j^{(t)}} \nabla F_i(\theta_j^{(t)}; Z_i^{(t)}),$$

where $S_j^{(t)}$ denotes the set of worker machines whose cluster identity estimate is j in the t-th iteration. In the following, we discuss the convergence guarantee of IFCA under two models: in Section 3.3, we analyze the algorithm under a linear model with Gaussian features and squared loss, and in Section 3.3, we analyze the algorithm under a more general setting of strongly convex loss functions.

Linear models with squared loss

In this section, we analyze our algorithm in a concrete linear model. This model can be seen as a warm-up example for more general problems with strongly convex loss functions that we discuss in Section 3.3, as well as a distributed formulation of the widely studied mixture of linear regression problem [60, 61]. We assume that the data on the worker machines in the

j-th cluster are generated in the following way: for $i \in S_j^*$, the feature-response pair of the i-th worker machine machine satisfies

$$y^{i,\ell} = \langle x^{i,\ell}, \theta_j^* \rangle + \epsilon^{i,\ell},$$

where $x^{i,\ell} \sim \mathcal{N}(0, I_d)$ and the additive noise $\epsilon^{i,\ell} \sim \mathcal{N}(0, \sigma^2)$ is independent of $x^{i,\ell}$. Furthermore, we use the squared loss function $f(\theta; x, y) = (y - \langle x, \theta \rangle)^2$. As we can see, this model is the mixture of linear regression model in the distributed setting. We observe that under the above setting, the parameters $\{\theta_j^*\}_{j=1}^k$ are the minimizers of the population loss function $F^j(\cdot)$.

We proceed to analyze our algorithm. We define $p_j := |S_j^*|/m$ as the fraction of worker machines belonging to the j-th cluster, and let $p := \min\{p_1, p_2, \ldots, p_k\}$. We also define the minimum separation Δ as $\Delta := \min_{j \neq j'} \|\theta_j^* - \theta_{j'}^*\|$, and $\rho := \frac{\Delta^2}{\sigma^2}$ as the signal-to-noise ratio. Before we establish our convergence result, we state a few assumptions. Here, recall that n' denotes the number of data that each worker uses in each step.

Assumption 1. *The initialization of parameters $\theta_j^{(0)}$ satisfy $\|\theta_j^{(0)} - \theta_j^*\| \leq \frac{1}{4}\Delta$, $\forall\ j \in [k]$.*

Assumption 2. *Without loss of generality, we assume that $\max_{j \in [k]} \|\theta_j^*\| \lesssim 1$, and that $\sigma \lesssim 1$. We also assume that $n' \gtrsim (\frac{\rho+1}{\rho})^2 \log m$, $d \gtrsim \log m$, $p \gtrsim \frac{\log m}{m}$, $pmn' \gtrsim d$, and $\Delta \gtrsim \frac{\sigma}{p}\sqrt{\frac{d}{mn'}} + \exp(-c(\frac{\rho}{\rho+1})^2 n')$ for some universal constant c.*

In Assumption 1, we assume that the initialization is close enough to θ_j^*. We note that this is a standard assumption in the convergence analysis of mixture models [1, 59], due to the non-convex optimization landscape of mixture model problems. In Assumption 2, we put mild assumptions on n', m, p, and d. The condition that $pmn' \gtrsim d$ simply assumes that the total number of data that we use in each iteration for each cluster is at least as large as the dimension of the parameter space. The condition that $\Delta \gtrsim \frac{\sigma}{p}\sqrt{\frac{d}{mn'}} + \exp(-c(\frac{\rho}{\rho+1})^2 n')$ ensures that the iterates stay close to θ_j^*.

We first provide a single step analysis of our algorithm. We assume that at a certain iteration, we obtain parameter vectors θ_j that are close to the ground truth parameters θ_j^*, and show that θ_j converges to θ_j^* at an exponential rate with an error floor.

Theorem 1. *Consider the linear model and assume that Assumptions 1 and 2 hold. Suppose that in a certain iteration of the IFCA algorithm we obtain parameter vectors θ_j with $\|\theta_j - \theta_j^*\| \leq \frac{1}{4}\Delta$. Let θ_j^+ be iterate after this iteration. Then there exist universal constants $c_1, c_2, c_3, c_4 > 0$ such that when we choose step size $\gamma = c_1/p$, with probability at least $1 - 1/\mathrm{poly}(m)$, we have for all $j \in [k]$,*

$$\|\theta_j^+ - \theta_j^*\| \leq \frac{1}{2}\|\theta_j - \theta_j^*\| + c_2 \frac{\sigma}{p}\sqrt{\frac{d}{mn'}} + c_3 \exp\left(-c_4 (\frac{\rho}{\rho+1})^2 n'\right).$$

We prove Theorem 1 in Appendix **??**. Here, we briefly summarize the proof idea. Using the initialization condition, we show that the set $\{S_j\}_{j=1}^k$ has a significant overlap with $\{S_j^*\}_{j=1}^k$. In the overlapped set, we then argue that the gradient step provides a contraction and error floor due to the basic properties of linear regression. We then bound the gradient norm of the miss-classified machines and add them to the error floor. We complete the proof by combining the contributions of properly classified and miss-classified worker machines. We can then iteratively apply Theorem 1 and obtain accuracy of the final solution $\widehat{\theta}_j$ in the following corollary.

Corollary 1. *Consider the linear model and assume that Assumptions 1 and 2 hold. By choosing step size* $\gamma = c_1/p$, *with probability at least* $1 - \frac{\log(\Delta/4\varepsilon)}{\mathrm{poly}(m)}$, *after* $T = \log \frac{\Delta}{4\varepsilon}$ *parallel iterations, we have for all* $j \in [k]$, $\|\widehat{\theta}_j - \theta_j^*\| \leq \varepsilon$, *where* $\varepsilon = c_5 \frac{\sigma}{p}\sqrt{\frac{d}{mn'}} + c_6 \exp(-c_4(\frac{\rho}{\rho+1})^2 n')$.

Let us examine the final accuracy. Since the number of data points on each worker machine $n = 2n'T = 2n' \log(\Delta/4\varepsilon)$, we know that for the smallest cluster, there are a total of $2pmn' \log(\Delta/4\varepsilon)$ data points. According to the minimax estimation rate of linear regression [53], we know that *even if we know the ground truth cluster identities*, we cannot obtain an error rate better than $\mathcal{O}(\sigma\sqrt{\frac{d}{pmn' \log(\Delta/4\varepsilon)}})$. Comparing this rate with our statistical accuracy ε, we can see that the first term $\frac{\sigma}{p}\sqrt{\frac{d}{mn'}}$ in ε is equivalent to the minimax rate up to a logarithmic factor and a dependency on p, and the second term in ε decays exponentially fast in n', and therefore, our final statistical error rate is *near optimal.*

Strongly convex loss functions

In this section, we study a more general scenario where the population loss functions of the k clusters are strongly convex and smooth. In contrast to the previous section, our analysis do not rely on any specific statistical model, and thus can be applied to more general machine learning problems. We start with reviewing the standard definitions of strongly convex and smooth functions $F : \mathbb{R}^d \mapsto \mathbb{R}$.

Definition 1. F *is* λ*-strongly convex if* $\forall \theta, \theta'$, $F(\theta') \geq F(\theta) + \langle \nabla F(\theta), \theta' - \theta \rangle + \frac{\lambda}{2}\|\theta' - \theta\|^2$.

Definition 2. F *is* L*-smooth if* $\forall \theta, \theta'$, $\|\nabla F(\theta) - \nabla F(\theta')\| \leq L\|\theta - \theta'\|$.

In this section, we assume that the population loss functions $F^j(\theta)$ are strongly convex and smooth.

Assumption 3. *The population loss function* $F^j(\theta)$ *is* λ*-strongly convex and* L*-smooth,* $\forall j \in [k]$.

We note that we do not make any convexity or smoothness assumptions on the individual loss function $f(\theta; z)$. Instead, we make the following distributional assumptions on $f(\theta; z)$ and $\nabla f(\theta; z)$.

Assumption 4. *For every θ and every $j \in [k]$, the variance of $f(\theta; z)$ is upper bounded by η^2, when z is sampled according to $\mathcal{D}_j$, i.e., $\mathbb{E}_{z\sim\mathcal{D}_j}[(f(\theta; z) - F^j(\theta))^2] \le \eta^2$*

Assumption 5. *For every θ and every $j \in [k]$, the variance of $\nabla f(\theta; z)$ is upper bounded by v^2, when z is sampled according to $\mathcal{D}_j$, i.e., $\mathbb{E}_{z\sim\mathcal{D}_j}[\|\nabla f(\theta; z) - \nabla F^j(\theta)\|_2^2] \le v^2$*

Bounded variance of gradient is very common in analyzing SGD [7]. In this report we use loss function value to determine cluster identity, so we also need to have a probabilistic assumption on $f(\theta; z)$. We note that bounded variance is a relatively weak assumption on the tail behavior of probability distributions. In addition to the assumptions above, we still use some definitions from Section 3.3, i.e., $\Delta := \min_{j\neq j'} \|\theta_j^* - \theta_{j'}^*\|$, and $p = \min_{j\in[k]} p_j$ with $p_j = |S_j^*|/m$. We make the following assumptions on the initialization, n', p, and Δ.

Assumption 6. *Without loss of generality, we assume that $\max_{j\in[k]} \|\theta_j^*\| \lesssim 1$. We also assume that $\|\theta_j^{(0)} - \theta_j^*\| \le \frac{1}{4}\sqrt{\frac{\lambda}{L}}\Delta$, $\forall j \in [k]$, $n' \gtrsim \frac{k\eta^2}{\lambda^2\Delta^4}$, $p \gtrsim \frac{\log(mn')}{m}$, and that $\Delta \ge \widetilde{\mathcal{O}}(\max\{(n')^{-1/5}, m^{-1/6}(n')^{-1/3}\})$.*

Here, for simplicity, the $\widetilde{\mathcal{O}}$ notation omits any logarithmic factors and quantities that do not depend on m and n'. As we can see, again we need to assume good initialization, due to the nature of the mixture model, and the assumptions that we impose on n', p, and Δ are relatively mild; in particular, the assumption on Δ ensures that the iterates stay close to an ℓ_2 ball around θ_j^*.

Theorem 2. *Suppose Assumptions 3-6 hold. Choose step size $\gamma = 1/L$. Then, with probability at least $1 - \delta$, after $T = \frac{8L}{p\lambda}\log\left(\frac{\Delta}{2\varepsilon}\right)$ parallel iterations, we have for all $j \in [k]$, $\|\widehat{\theta}_j - \theta_j^*\| \le \varepsilon$, where*

$$\varepsilon \lesssim \frac{vkL\log(mn')}{p^{5/2}\lambda^2\delta\sqrt{mn'}} + \frac{\eta^2L^2k\log(mn')}{p^2\lambda^4\delta\Delta^4n'} + \widetilde{\mathcal{O}}(\frac{1}{n'\sqrt{m}}).$$

We prove Theorem 2 in the Appendix **??**. Similar to Section 3.3, to prove this result, we first prove a per-iteration contraction

$$\|\theta_j^+ - \theta_j^*\| \le (1 - \frac{p\lambda}{8L})\|\theta_j - \theta_j^*\| + \widetilde{\mathcal{O}}(\frac{1}{\sqrt{mn'}} + \frac{1}{n'} + \frac{1}{n'\sqrt{m}}),\ \forall j \in [k],$$

and then derive the convergence rate. To better interpret the result, we focus on the dependency on m and n and treat other quantities as constants. Then, since $n = 2n'T$, we know that n and n' are of the same scale up to a logarithmic factor. Therefore, the final statistical error rate that we obtain is $\epsilon = \widetilde{\mathcal{O}}(\frac{1}{\sqrt{mn}} + \frac{1}{n})$. As discussed in Section 3.3, $\frac{1}{\sqrt{mn}}$ is the optimal rate even if we know the cluster identities; thus our statistical rate is near optimal in the regime where $n \gtrsim m$. In comparison with the statistical rate in linear models $\widetilde{\mathcal{O}}(\frac{1}{\sqrt{mn}} + \exp(-n))$, we note that the major difference is in the second term. The additional terms of the linear model and the strongly convex case are $\exp(-n)$ and $\frac{1}{n}$, respectively. We note that this is due to different statistical assumptions: in for the linear model, we assume Gaussian noise whereas here we only assume bounded variance.

3.4 Experiments

In this section, we present our experimental results, which not only validate the theoretical claims in Section 3.3, but also demonstrate that our algorithm can be efficiently applied beyond the regime we discussed in the theory. We emphasize that we *do not* re-sample fresh data points at each iteration. Furthermore, the requirement on the initialization can be relaxed. More specifically, for linear models, we observe that random initialization with a few restarts is sufficient to ensure convergence of Algorithm 1. In our experiments, we also show that our algorithm works efficiently for problems with non-convex loss functions such as neural networks.

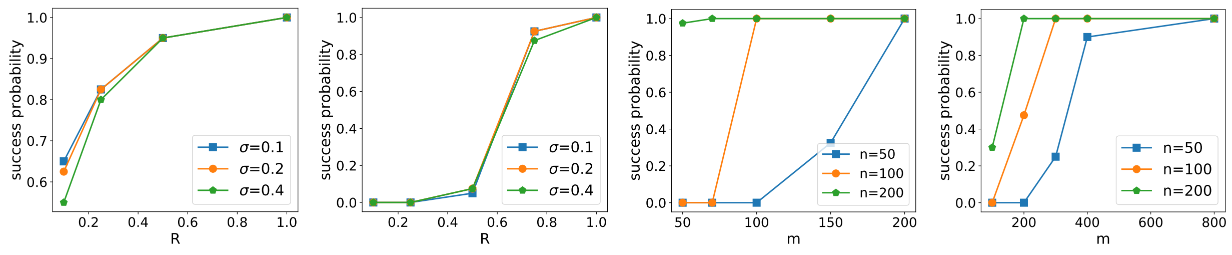
Synthetic data

We begin with evaluation of Algorithm 1 with gradient averaging (option I) on linear models with squared loss, as described in Section 3.3. For all $j \in [k]$, we first generate $\theta_j^* \sim \mathsf{Bernoulli}(0.5)$ coordinate-wise, and then rescale their ℓ_2 norm to R. This ensures that the separation between the θ_j^*'s is proportional to R in expectation, and thus, in this experiment, we use R to represent the *separation* between the ground truth parameter vectors. Moreover, we simulate the scenario where all the worker machines participate in all iterations, and all the clusters contain same number of worker machines. For each trial of the experiment, we first generate the parameter vectors θ_j^*'s, fix them, and then randomly initialize $\theta_j^{(0)}$ according to an independent coordinate-wise Bernoulli distribution. We then run Algorithm 1 for 300 iterations, with a constant step size. For $k = 2$ and $k = 4$, we choose the step size in $\{0.01, 0.1, 1\}$, $\{0.5, 1.0, 2.0\}$, respectively. In order to determine whether we successfully learn the model or not, we sweep over the aforementioned step sizes and define the following distance metric: $\mathsf{dist} = \frac{1}{k}\sum_{j=1}^{k} \|\widehat{\theta}_j - \theta_j^*\|$, where $\{\widehat{\theta}_j\}_{j=1}^{k}$ are the parameter estimates obtained from Algorithm 1. A trial is dubbed *successful* if for a fixed set of θ_j^*, among 10 random initialization of $\theta_j^{(0)}$, at least in one scenario, we obtain $\mathsf{dist} \leq 0.6\sigma$.

In Fig. 3.2 (a-b), we plot the empirical success probability over 40 trials, with respect to the separation parameter R. We set the problem parameters as (a) $(m, n, d) = (100, 100, 1000)$ with $k = 2$, and (b) $(m, n, d) = (400, 100, 1000)$ with $k = 4$. As we can see, when R becomes larger, i.e., the separation between parameters increases, and the problem becomes easier to solve, yielding in a higher success probability. This validates our theoretical result that higher signal-to-noise ratio produces smaller error floor. In Fig. 3.2 (c-d), we characterize the dependence on m and n, with fixing R and d with $(R, d) = (0.1, 1000)$ for (c) and $(R, d) = (0.5, 1000)$ for (d). We observe that when we increase m and/or n, the success probability improves. This validates our theoretical finding that more data and/or more worker machines help improve the performance of the algorithm.

Rotated MNIST and CIFAR

We also create clustered FL datasets based on the MNIST [29] and CIFAR-10 [26] datasets. In order to simulate an environment where the data on different worker machines are generated from different distributions, we augment the datasets using rotation, and create the Rotated



(a) $k = 2$ (b) $k = 4$ (c) $k = 2$ (d) $k = 4$

Figure 3.2: Success probability with respect to: (a), (b) the separation scale R and the scale of additive noise σ; (c), (d) the number of worker machines m and the sample size on each machine n. In (a) and (b), we see that the success probability gets better with increasing R, i.e., more separation between ground truth parameter vectors, and in (c) and (d), we note that the success probability improves with an increase of mn, i.e., more data on each machine and/or more machines.

MNIST [37] and Rotated CIFAR datasets. For **Rotated MNIST**, recall that the MNIST dataset has 60000 training images and 10000 test images with 10 classes. We first augment the dataset by applying $0, 90, 180, 270$ degrees of rotation to the images, resulting in $k = 4$ clusters. For given m and n satisfying $mn = 60000k$, we randomly partition the images into m worker machines so that each machine holds n images *with the same rotation*. We also split the test data into $m_{\text{test}} = 10000k/n$ worker machines in the same way. The **Rotated CIFAR** dataset is also created in a similar way as Rotated MNIST, with the main difference being that we create $k = 2$ clusters with 0 and 180 degrees of rotation. We note that creating different tasks by manipulating standard datasets such as MNIST and CIFAR-10 has been widely adopted in the continual learning research community [18, 24, 37]. For clustered FL, creating datasets using rotation helps us simulate a federated learning setup with clear cluster structure.

For our MNIST experiments, we use the fully connected neural network with ReLU activations, with a single hidden layer of size 200; and for our CIFAR experiments, we use a convolution neural network model which consists of 2 convolutional layers followed by 2 fully connected layers, and the images are preprocessed by standard data augmentation such as flipping and random cropping.

We compare our IFCA algorithm with two baseline algorithms, i.e., the *global model*, and *local model* schemes. For **IFCA**, we use model averaging (option II in Algorithm 1). For MNIST experiments, we use full worker machines participation ($M_t = [m]$ for all t). For LocalUpdate step in Algorithm 1, we choose $\tau = 10$ and step size $\gamma = 0.1$. For CIFAR experiments, we choose $|M_t| = 0.1m$, and apply step size decay 0.99, and we also set $\tau = 5$ and batch size 50 for LocalUpdate process, following prior works [40]. In the **global model** scheme, the algorithm tries to learn single global model that can make predictions from all the distributions. The algorithm does not consider cluster identities, so model averaging step in Algorithm 1 becomes $\theta^{(t+1)} = \sum_{i \in M_t} \widetilde{\theta}_i / |M_t|$, i.e. averaged over parameters from all

Table 3.1: Test accuracies(%) $\pm$ std on Rotated MNIST ($k = 4$) and Rotated CIFAR ($k = 2$)

	Rotated MNIST			Rotated CIFAR
m, n	4800, 50	2400, 100	1200, 200	200, 500
IFCA (ours)	**94.20 $\pm$ 0.03**	**95.05 $\pm$ 0.02**	**95.25 $\pm$ 0.40**	**81.51 $\pm$ 1.37**
global model	86.74 $\pm$ 0.04	88.65 $\pm$ 0.08	89.73 $\pm$ 0.13	77.87 $\pm$ 0.39
local model	63.32 $\pm$ 0.02	73.66 $\pm$ 0.04	80.05 $\pm$ 0.02	33.97 $\pm$ 1.19

the participating machines. In the **local model** scheme, the model in each node performs gradient descent only on local data available, and model averaging is not performed.

For IFCA and the global model scheme, we perform inference in the following way. For every test worker machine, we run inference on all learned models (k models for IFCA and one model for global model scheme), and calculate the accuracy from the model that produces the smallest loss value. For testing the local model baselines, the models are tested by measuring the accuracy on the test data with the same distribution (i.e. those have the same rotation). We report the accuracy averaged over all the models in worker machines. For all algorithms, we run experiment with 5 different random seeds and report the average and standard deviation.

Our experimental results are shown in Table 4.3. We can observe that our algorithm performs better than the two baselines. As we run the IFCA algorithm, we observe that we can gradually find the underlying cluster identities of the worker machines, and after the correct cluster is found, each model is trained and tested using data with the same distribution, resulting in better accuracy. The global model baseline performs worse than ours since it tries to fit all the data from different distributions, and cannot provide personalized predictions. The local model baseline algorithm overfits to the local data easily, leading to worse performance than ours.

Federated EMNIST

We provide additional experimental results on the Federated EMNIST (FEMNIST) [2], which is a realistic FL dataset where the data points on every worker machine are the handwritten digits or letters from a specific writer. Although the data distribution among all the users are similar, there might be ambiguous cluster structure since the writing styles of different people may be clustered. We use the weight sharing technique mentioned in Section 3.2. We use a neural network with two convolutional layers, with a max pooling layer after each convolutional layer, followed by two fully connected layers. We share the weights of all the layers, except the last layer which is trained by IFCA. We treat the number of clusters k as a hyper parameter and run the experiments with different values of k. We compare IFCA with the global model and local model approaches, as well as the one-shot centralized clustering algorithm in [16]. The test accuracies are shown in Table 3.2, with mean and standard deviation computed over 5 independent runs. As we can see, IFCA shows clear advantage over the global model and local model approaches. The results of IFCA and the

Table 3.2: Test accuracies (%) ± std on FEMNIST

IFCA ($k = 2$)	IFCA ($k = 3$)	one-shot ($k = 2$)	one-shot ($k = 3$)	global	local
87.99 ± 0.35	87.89 ± 0.52	87.41 ± 0.39	87.38 ± 0.37	84.45 ± 0.51	75.85 ± 0.72

one-shot algorithm are similar. However, as we emphasized in Section **??**, IFCA does not run a centralized clustering procedure, and thus reduces the computational cost at the center machine. As a final note, we observe that IFCA is robust to the choice of the number of clusters k. The results of the algorithm with $k = 2$ and $k = 3$ are similar, and we notice that when $k > 3$, IFCA automatically identifies 3 clusters, and the remaining clusters are empty. This indicates the applicability of IFCA in real-world problems where the cluster structure is ambiguous and the number of clusters is unknown.

Chapter 4

UIFCA: a generative model-based clustering method based on IFCA

4.1 Problem Setup

We consider a standard data clustering task in a distributed setting, where one central server communicates with n client machines. We assume that total m datapoints are inherently partitioned into K disjoint clusters, $S_1^*, \dots, S_K^*$, and our goal is to find them. We denote j-th datapoint in client i by $x^{i,j}$. Each set S_k^* consists of $\frac{m}{K}$ datapoints coming from the distribution D_k, for $k \in [K]$. We consider unsupervised learning task where the cluster information $S_1^*, \cdots, S_K^*$ is not visible from the learning algorithm. The central server is able to communicate with client machines using predefined secure protocol, such as secure aggregation.

We aim to find an algorithm that finds clusters regardless of whether the client's data is given i.i.d. or not. To quantify the level of heterogeneity in clients, we define heterogeneity level p that measures how much the data's cluster is skewed across the machines. For example, a dataset with $K = 10$ clusters with $p = 0$ refers to the case where data are distributed to clients in purely i.i.d. manner, and in $p = 1$ case, each client holds data from a single distribution. For a client holding s datapoints, the first sp data points are sampled from a single cluster, and remaining $s(1 - p)$ data points are drawn from any clusters at random. Illustrations of different p cases are given in Figure 4.1.

In order to estimate the cluster information of datapoints in clients, we train K different models that capture each cluster's datapoints, following the approach of IFCA [15]. Each model is a generative model that is trained to capture the distribution of a given cluster data. Cluster membership of a datapoint can be evaluated by picking the model that gives the highest likelihood, i.e., selecting the model's distribution that it is the most close to. We define $\{\theta_1, \theta_2, ..., \theta_K\}$ as the model parameters learned for each cluster, and $f_\theta(\cdot)$ as the loss function of a sample evaluated by the model θ. Each client i assigns its each of its datapoint $x^{i,j}$ to one of the cluster sets $\{S_{i,1}, S_{i,1}, ..., S_{i,K}\}$, and runs model update on each parameter

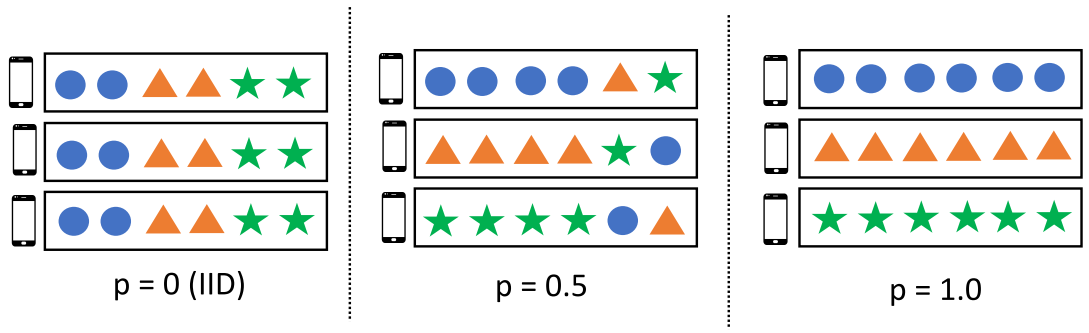


Figure 4.1: An illustration of a client heterogeneity type we consider.

θ_k with set $S_{i,k}$ to optimize the following local objective:

$$\min_{\theta_k} \frac{1}{|S_{i,k}|} \sum_{x^{i,j} \in S_{i,k}} f_{\theta_k}(x^{i,j}).$$

for $k \in [K]$. Then the model updates are aggregated at the central server to optimize the following global objective:

$$\min_{\theta_k} \frac{1}{|S_{*,k}|} \sum_{i=1}^{n} \sum_{x^{i,j} \in S_{i,k}} f_{\theta_k}(x^{i,j}).$$

where $S_{*,k}$ is union of $\{S_{i,k}\}_{i=1}^{n}$. This objective minimizes the loss for all the data assigned to cluster k in all clients, for each cluster $k \in [K]$. We also define $F_\theta(B) = \sum_{x \in B} f_\theta(x)$ to be the loss used in batch gradient update of a data batch B.

We note that our procedure of inferring cluster information is inherently secure, and also beneficial. A client can receive models $\{\theta_1, \theta_2, ..., \theta_K\}$ from the server and *locally* evaluate its datapoints' cluster membership. The central server can access each cluster's content by investigating the model(for example, sampling datapoints with high likelihood), and provide additional beneficial information (such as advertisements) that may be relevant for that cluster. The users have options to choose which additional information they will use, based on the evaluated cluster information, which is not revealed to the central server.

4.2 Algorithms and Models

In this section, we first present a straightforward extension of existing algorithms to a federated learning setting which we will consider as baselines, and discuss potential drawbacks of these methods. Then, we provide details of UIFCA algorithm and the models used.

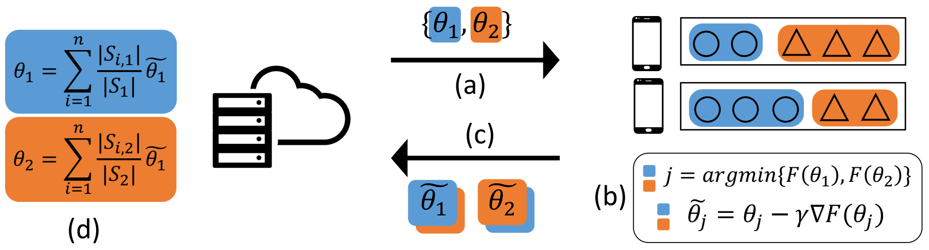

$\theta_1 = \sum_{i=1}^{n} \frac{|S_{i,1}|}{|S_1|} \widetilde{\theta_1}$
$\theta_2 = \sum_{i=1}^{n} \frac{|S_{i,2}|}{|S_2|} \widetilde{\theta_1}$
(d)
$\{\theta_1, \theta_2\}$
(a)
(c)
$\widetilde{\theta_1}$ $\widetilde{\theta_2}$
(b)
$j = argmin\{F(\theta_1), F(\theta_2)\}$
$\widetilde{\theta_j} = \theta_j - \gamma \nabla F(\theta_j)$

Figure 4.2: An overview of UIFCA . (a) Central server broadcast models to n clients. (b) Clients identify their local datapoints' cluster memberships and run local updates with received models. (c) The clients send back the local models to central server. (d) Central server aggregates the models within the same estimated cluster S_j.

Baselines

A natural approach to clustering in a federated environment is to implement a distributed version of k-means algorithm proposed by [8]. Each worker can compute the local estimate of the centroids, and global centroids can be updated by gathering local centroids and running k-means clustering algorithm over these centroids. The client can infer the cluster membership of each datapoint by referring to local centriods' cluster assignment among all local centroids in the server. This approach is summarized in Algorithm 2. We consider this approach as a baseline to compare with UIFCA .

For the data with more complex cluster structures such as real-world images, the k-means based approach may not work well. One may consider deep learning model-based clustering methods, which showed great success in capturing complex features of images and clustering them. The method commonly involves learning a clustering model that infers a cluster label of a sample or a representation that is well separated by the sample's inherent cluster.

In order to adapt these methods into a federated environment, a natural approach is to run distributed model training using secure training methods such as FedAvg algorithm. As a baseline for our experiments, we consider clusterGAN [43] for clustering real image data, combined with FedAvg. For each communication round, each client will update the local replica of clusterGAN model with local data, and updated models from clients will be aggregated at the central server.

However, this approach is problematic when it applies to heterogeneous clients. When client's heterogenity increases, FedAvg's local objective can become different completely different from one another, resulting in increasing difficulty in learning a consensus model that performs well for all distributions [33]. Often, gradient averaging (or minibatch SGD) is proposed as alternative for FedAvg for better convergence behavior in non-i.i.d setting [63, 56, 57], but we do not consider it since its practical use is not common, due to communication being the bottleneck for large models, and FedAvg enables multiple local updates within one communication round while gradient averaging only updates once.

We claim that our method has structural advantage compared to baselines in this sense.

Algorithm 2 k-FED algorithm

```
1: Client $i$ clusters local data $x^{i,j}$ into $S_{i,1}, \dots, S_{i,K}$ based on distance to $\mu_1, \dots, \mu_K$(for all $i \in [n]$).
       $S_{i,k} \leftarrow \{j | k = \arg\min_{k \in [K]} \|\mu_k - x^{i,j}\|\}$
2: Client $i$ computes local centroids $\mu_{i,1}, \dots, \mu_{i,K}$ and send to central server.
3: Central server runs $k$-means algorithm over all local centroids $\mu_{i,j}$(for all $i \in [n]$, $j \in [K]$).
4: Client $i$ assigns local data $x^{i,j}$ to a cluster according to cluster assignment of its local centroids $\mu_{i,1}, \dots, \mu_{i,K}$ obtained from server.
```

Our method captures the datapoints that are likely to be from same distribution, and runs FedAvg with them. This enables learning models in heterogeneous client data, leveraging fast local updates of FedAvg.

UIFCA

IFCA [15] algorithm is a clustering algorithm that clusters clients by its data using deep neural networks in a federated setting. The algorithm recovers optimal clusters by iteratively alternating between estimating cluster identities and optimizing the cluster models. Starting from K randomly initialized models, cluster identities of clients are found by assigning the model that gives best score (usually referring to smallest loss), and models are updated by averaging the model's SGD updates from clients within the same cluster. IFCA [15] was proven to be able to recover correct cluster identities under mild conditions, and was shown to be successful in simple clustering tasks such as grouping the images by rotations by training classifier models as cluster models with supervised data.

We adapt IFCA's training method to our problem to leverage its powerful clustering ability in federated settings. IFCA considers a setting where each client has data drawn i.i.d. from a single distribution, while our problem setting assumes that the data points in a client can come from different clusters. To reflect this change, our algorithm runs client local updates for all K cluster models with data assigned to each corresponding cluster, while in IFCA, each client only updates one cluster model locally. Also, in order to accomodate unsupervised data, we use a generative model as cluster parameter model, to let each model capture each cluster's data distribution.

We now discuss details of our algorithm. The algorithm is formally presented in algorithms 3 to 5 and illustrated in Figure 4.2.

The algorithm starts with K randomly initialized model parameters $\theta_1^{(0)}, \dots, \theta_K^{(0)}$, and initial random cluster assignment $\{\{S_{i,k}^{(0)}\}_{k=1}^K\}_{i=1}^n$. In the t-th cluster round, the center machine broadcasts current model parameters $\theta_1^{(t)}, \dots, \theta_K^{(t)}$ to all the machines.

For each cluster $k \in [K]$, the clients collectively runs FedAvg algorithm with model $\theta_k^(t)$ to capture the distribution of the k-th cluster's data across the clients, using the received model parameters(shown in Algorithm 4). Each client will run batch gradient update M

Algorithm 3 UIFCA

```
1: Input: Client samples $\{x^{i,*}\}_{i=1}^{n}$, number of cluster rounds $T$, initial cluster assignment $\{\{S_{i,k}^{0}\}_{k=1}^{K}\}_{i=1}^{n}$
2: For $t = 1, \cdots, T$ do
3:    (Learning) Fit generative model for each cluster
4:    For cluster $k \in [K]$ in parallel do
5:       $\theta_k^{(t+1)} = \texttt{LearnClusterModel}(S_*^{(t)}, S_{*,k}^{(t)})$
6:    (Assigning) Assign each sample to a cluster:
7:    For client $i \in [m]$ in parallel do
          $$S_{i,k}^{(t)} \leftarrow \{j | k = \arg\min_{k\in[K]} f_{\theta_k^{(t)}}(x^{i,j})\}$$
```

Algorithm 4 `LearnClusterModel`(S)

```
1: Input: Data assigned to cluster $k$ $S = (S_{1,k}, \ldots, S_{n,k})$
2: Choose:  Number of communication rounds $\tau$
3:    Initialize $\theta^0$ randomly
4:    For each round $l = 1, \cdots, \tau$ do
5:       For client $i \in [n]$ in parallel do
6:          $\theta_i^{(l)} = \texttt{ModelUpdate}(\theta^{(l-1)}, S_{i,k})$
7:       $\theta^{(l)} = \sum_{i=1}^{m} \frac{|S_{i,k}|}{|S_{*,k}|} \theta_i^{(l)}$
8: Return:  $\theta^{\tau}$
```

Algorithm 5 `ModelUpdate`(θ, S)

```
1: Input: Initial parameter $\theta$, set $S$
2: Choose:  Step size $\eta$, number $M$ of gradient steps, batch size $N$
3: For $j = 1, \cdots, M$ do
4:       $B_j \leftarrow \texttt{random\_subset}(S, N)$
5:       $\theta^j \leftarrow \theta^{j-1} - \eta \left(\frac{1}{N} \nabla_\theta F_{\theta^{j-1}}(B_j)\right)$
6: Return:  $\theta^M$
```

times for each model and its corresponding cluster set (shown in Algorithm 5). These local model updates are averaged in the central server, weighted by cluster's size. After running τ times of averaging, an optimized cluster models $\theta_1^{(t+1)}, \ldots, \theta_K^{(t+1)}$ are found. These models are then broadcast to all clients. A cluster round ends up with client re-evaluating the cluster identities of local datapoint by finding model parameter with lowest loss(highest likelihood), i.e., $\text{argmin}_{k\in[K]} f_{\theta_k^{(t)}}(x^{i,j})$). The cluster rounds iterate over T times to find optimal cluster structure in the client's data.

Using normalizing flow models with UIFCA

For selecting the which generative model to use with UIFCA , we consider normalizing flow model [52]. Normalizing flow models are generative models that model the distribution of input data, by learning an invertible function g that maps from base distribution $p_Z(z)$ to the target distribution $p_X(x)$. With base distribution $p_Z(z)$ given (commonly standard Gaussian), the likelihood of of a sample x can be found by change of variables formula:

$$p_X(x) = p_Z(g^{-1}(x)) \left| \det \frac{\partial g}{\partial x} \right|$$

The model is trained to capture the distribution by maximizing the log-likelihood of the training data with respect to the parameters of the mapping function:

$$\max_g \sum_{i=1}^{n} \log(p_X(x_i))$$

Among many options of generative models that can provide sample likelihood (such as variational autoencoders [22]), normalizing flow models are best fit to our needs, since it explicitly models the data distribution, it can give the most exact estimate of sample likelihood compared to generative models.

4.3 Experiments

In this section, we present our experimental results. We evaluate UIFCA with synthetic datasets and realistic image datasets based on MNIST. Our method correctly recovers clusters for synthetic settings, but does not perform well on MNIST, so we also discuss possible reasons and ways to improve it.

Synthetic experiments

We consider following two types of synthetically generated data.

Gaussian clusters Data samples with dimension $d = 32$ are generated from K Gaussian distributions with same standard deviation $\sigma = 1$ and different centers. To ensure that the clusters have less overlapping data, we generate distribution centers $\theta_k^* \sim \mathsf{Bernoulli}(0.5)$ for all $k \in [kK]$, coordinate-wise, and scale them by R. The R represents minimum separation between each center. For our experiments, we use $R = 5$ for minimal overlap.

Subspace clusters Subspace clustered data [45] is a mixture of distributions that lies in different subspaces. The cluster structure is not easily discoverable using simple clustering algorithms such as k-means. For generating the data, a $d = 32$ dimensional basis set of $d_{\text{subspace}} = 16$ orthonormal basis vectors are sampled for each of k clusters, and clustered

Table 4.1: Cluster accuracies (%) on synthetic datasets.

	p	0.0	0.25	0.5	0.75	1.0
Gaussian	UIFCA			**100**		
	k-FED			**100**		
Subspace	UIFCA			**100**		
	k-FED	11.5	11.7	12.5	15.8	19.3

dataset is achieved by muitlplying random gaussian coefficients to each basis sets of each cluster.

For the model, we use 1-layer planar flow [47] with following linear transformation function:

$$g(z) = w^T z + b$$

with Gaussian prior $z = \mathcal{N}(0,1)$. We initialize the models by first generating random parameters for a single model, and adding small random normal noise to each parameter. This procedure will ensure avoiding initial cluster degeneracy cases, where a particular model initializes a much smaller loss compared to other models, gets most datapoints assigned and fits data regardless of the clusters, while other models cannot learn due to the small number of data points assigned to them. We run Algorithm 3 for $T = 20$ cluster rounds, with each round consisting of $\tau = 100$ communication rounds in Algorithm 4 and $M = 100$ local batch updates. For distributed training with FedAvg, we assume that all clients participate in each communication iteration. For each type of synthetic data, we test our algorithm with $k = 4$ clusters and $n = 4$ clients with each client having 1000 datapoints, resulting $m = 4000$ datapoints in total. We report cluster accuracy, defined as $\frac{1}{m}\sum_c \max_y |S_c \cap S_y|$ which measures purity of each cluster in terms of given true label.

Clustering performance of UIFCA is reported in Table 4.1. As we can see, for our provided synthetic cases, UIFCA is able to fully recover cluster information of individual datapoints inside clients, without compromising security assumptions of federated setting. For Gaussian clusters, each flow model learns the transformation from standard gaussian prior to each cluster distribution. For subspace clusters, the datapoints are given to follow gaussian distribution defined in different sets of basis sets, and our model learns the projection from standard basis to such subspaces. k-FED algorithm does not perform well for subspace clustered data since the data are spread out over different dimensions, distance metric of the algorithm becomes irrelevant to the cluster structure.

To provide in-depth look of UIFCA 's behavior, we plot the cluster accuracy of UIFCA with Gaussian clustered data at each cluster round in Figure 4.3. We can observe cluster accuracy iteratively improving over cluster rounds. Starting from random cluster assignment, each flow model captures the distribution of data assigned to its cluster. The model learns to assign high likelihood to the majority type of the data (We denote 'type' by the ground truth cluster of a data.) leading to grabbing more data of the major type and less of the other type, thus improving the cluster. Note that the cluster accuracy converges faster as client's data heterogeneity(p) increases, due to a FedAvg's weighted averaging behavior. FedAvg's

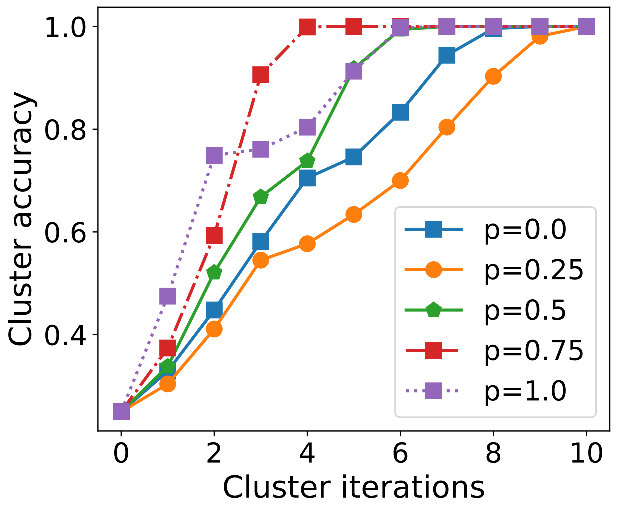


Figure 4.3: Cluster accuracies with respect to cluster rounds for synthetic Gaussian clustered data, for different heterogeneity levels p.

local update would converge faster when the large portion of (same cluster) data points are provided in the same device, performing close to centralized SGD. On the other hand, convergence would be relatively slow when p decreases, due to instability caused by model averaging procedure.

MNIST experiments

We also test the performance of UIFCA with two types of clustered dataset based on MNIST [28] dataset. Then, we discuss limitations of our approach and a possible approach to improve.

Cluster by digits The MNIST dataset consists of $m = 60000$ 28x28 images of 10 digits with each digit having approximately 6000 images. Setting the digit information as a ground truth label for the clustering task, we evaluate unsupervised clustering performance of our algorithm with $K = 10$ clusters. We simulate the setting where the data are distributed clients according to different heterogeneity level p, same as the synthetic experiments.

Cluster by rotation To simulate data coming from different distributions, we also create a clustered dataset by applying rotations. We select one of the 10 digits and generate dataset of $k = 4$ clusters by applying $0, 90, 180, 270$ degree of rotation, resulting in unsupervised dataset of apprixmately $24,000$ images. Mixing two or more digits is not considered, in order to ensure the cluster is formed by rotation, not digits. We consider digits $2, 3, 4, 5$, which are some of the digits that does not confuse rotation recognition(such as 8). We run clustering algorithm for each of the digits and report average cluster accuracy.

Cluster representations Since many pre-trained models are publicly available, it is often more practical to embeddings(representations) of user data for clustering rather than

	Default	RB	CM	RB+CM
ACC(%)	50.7	95.09	**99.0**	98.6

Table 4.2: Ablation study of two methods: randomizing the background of digit images(RB), and changing masking pattern of in RealNVP(CM).

clustering the raw images. We test our method for clustering image representations from a pre-trained network. For the pre-trained model, we train a RotNet [17] that predicts image rotations, trained using rotated MNIST images in centralized settings. Each client will have access to this pre-trained model, and is able to obtain the representations of its local data from this model. For the network, we use Alexnet [27] and extract activations from last convolutional layer.

Using initialization obtained from baseline We often find our method performing bad due to initializations. To improve this situation, we also test an additional method that starts UIFCA method with initial cluster assignment obtained from k-FED (named k-FED + UIFCA). We provide experimental result of this method for clustering representations.

Issues with RealNVP We consider RealNVP [11] as the flow model to use with our algorithm, which is one of the common types of normalizing flow model targeted for images. However, our preliminary experiments showed that using standard RealNVP architecture with UIFCA cannot cluster at all. We observed that the likelihoods of the samples from cluster set used for training the model, were indistinguishible from samples outside the cluster set, which makes likelihood-based cluster assignment completely fail. We reason this failure by referring to an observation from [23], stating that a typical RealNVP model captures graphical styles rather than features. A RealNVP model typically consists of multiple stacks of affine coupling transformation function. In each function, input is split into two parts, and the function is optimized to model the transformation between the two. The typical way to split the image input is to apply a pixel-level checkerboard pattern (often called checkerboard masking) . This architecture can be good at producing realistic images, but may assign high likelihood to images outside the training set, if the graphical style matches the training set. For our task of clustering MNIST, digit images consist of similar graphical images with a large portion of black background and a pattern of strokes, thus clustering based on likelihoods would have failed.

We consider two different solutions to tackle this issue. First, we consider randomizing the background of the digit images, as shown in Figure 4.5. For black pixels with value less than 0.01, we randomly set pixel value from $[0, 1]$. Because the background does not have a pattern, we expect models can focus more to the strokes of the digits. Second, we apply one of the methods proposed to address this issue in [23], which changes the masking pattern of the RealNVP models from the checkerboard type to the cyclic one in each coupling function. For details, we refer to [23]. We adapt their proposed architectural change to UIFCA from the author's code repository, and observe improved clustering performance compared to standard RealNVP models.

In order to select best configuration, we conduct an ablation study of these two solutions.

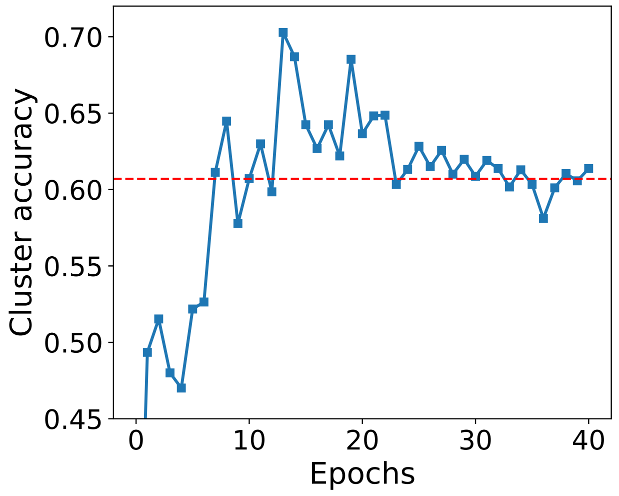


Figure 4.4: A sample of cluster accuracy measured at each epoch in single cluster iteration. The red horizontal line refers to the cluster accuracy of the given cluster set.

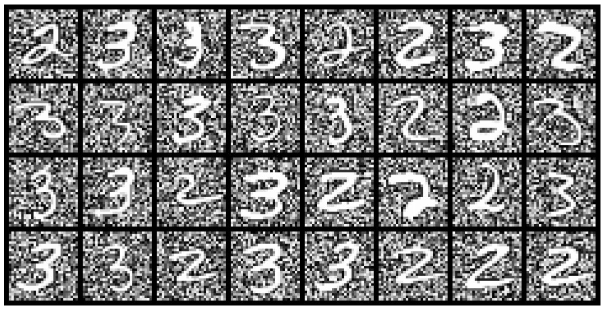
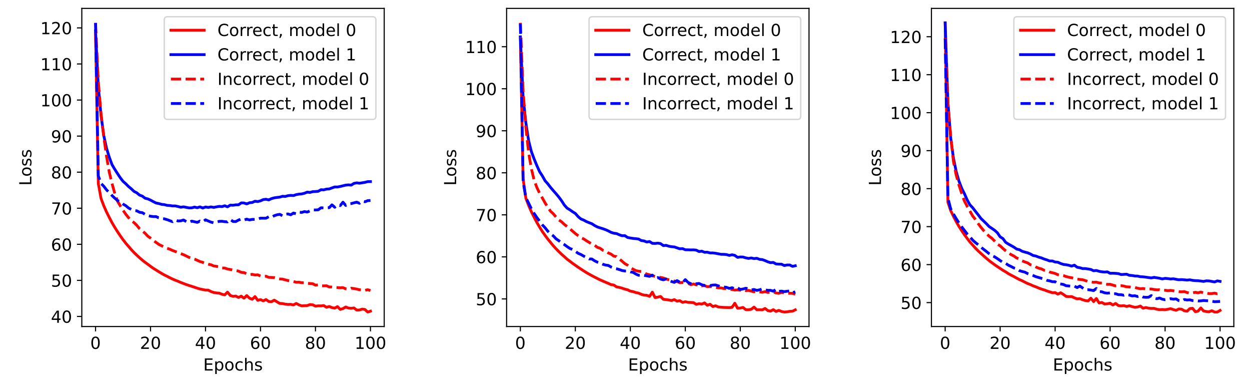


Figure 4.5: Randomizing the background of MNIST digit images.

We evaluate sample cluster assignment of two RealNVP flow models based on its likelihood, each trained with digit 2 and 3 respectively. The experiment measures the ability of distinguishing the samples in training set and the sample outside the training set, which is similar to Out-of-distribution detection. The results are shown in Table 4.2. We found changing masking pattern by itself is most effective. Randomizing background helps for the standard RealNVP model, but does not improve when the model's masking pattern is changed. Hence, we conclude that using changed masking pattern is best option for our setting.

For experiments of clustering rotations and digits of MNIST, we run Algorithm 3 for $T = 20$ cluster rounds, each having $\tau = 40$ communication rounds and $M = 100$ local updates. We use the SGD optimizer with a learning rate 1×10^{-4}, combined with the FedAvg algorithm. For clusterGAN [43], we use the model imported from the author's code repository, and use SGD optimizer with learning rate 5×10^{-3} for local batch updates. We set the number of



(a) $m = 800$ (b) $m = 2400$ (c) $m = 4000$

Figure 4.6: Mean loss of correctly-clustered samples and misclustered samples, measured with $k = 2$ models, in single cluster iteration.

Table 4.3: Cluster accuracies(%) of MNIST based datasets over different value of client heterogeneity level p.

	p	0.0	0.25	0.5	0.75	1.0
MNIST Rotated	UIFCA	63.5	60.8	63.4	79.5	79.6
	ClusterGAN	**95.7**	78.9	49.1	36.4	28.8
	k-FED	92.2	**87.1**	**80.5**	**78.3**	**100**
MNIST Digits	UIFCA	34.2	31.3	33.7	38.9	43.1
	ClusterGAN	**71.3**	57.4	39.4	27.5	16.8
	k-FED	57.0	**59.1**	**53.3**	**51.6**	**62.3**
MNIST Representations	k-FED	**86.0**	80.2	63.1	57.6	96.8
	UIFCA	43.6	56.4	78.2	79.5	99.5
	k-FED + UIFCA	85.1	**85.4**	**80.1**	**81.8**	**99.6**

clients n to be same number as clusters k.

We report cluster accuracies for $p \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ in Table 4.3. For raw images(Rotated, Digits), k-FED performs overall best. As expected, clusterGAN works best at i.i.d. case ($p = 0.0$) but fails as moving towards high heterogeneity case ($p = 1.0$), due to FedAvg's bad convergence under heterogeneous environment. We observe that ours reach higher cluster accuracy for high heteogeneity level cases compared to clusterGAN, but yields overall very low cluster accuracy. For clustering representations, we find that UIFCA often works better than the k-FED approach. The k-FED + UIFCA approach gives overall best performance, showing that generative models can often find better structure in distributions than k-means based approaches, and that UIFCA can also benefit from giving good initializations, especially when the cluster accuracies are low.

One of the key reason that UIFCA perform bad is that the cluster accuracy often worsens if model training gets long. We plot a sample trace of the cluster accuracies measured at each epoch in single cluster iteration of Algorithm 3 in Figure 4.4. The horizontal line is the cluster accuracy of assignments from previous cluster round, which is the cluster set that the models are trained with. We can observe that cluster accuracy does not always increase and converge as training progresses. It decreases after certain point, toward original cluster accuracy. Under the hood, we observe that cluster assignments are becoming similar to assignments of previous iteration.

Main cause of this phenomenon overfitting of each cluster model to its cluster set. To see this issue in detail, we reproduce the same issue in synthetic data setting of 2 gaussian clusters. For each cluster, we define correct samples as samples that are the majority type of the cluster's set, and all other samples as incorrect. In Figure 4.6 we plot the mean loss of correct and incorrect samples in cluster 1, evaluated with two cluster models θ_1, θ_2, for $m = 800, 2400, 4000$ cases. For the correct sample x in cluster 1, trained the model parameters results in $f_{\theta_1}(x) < f_{\theta_2}(x)$. For the incorrect sample x', we expect $f_{\theta_1}(x') > f_{\theta_2}(x')$ so that x' can move to cluster 2 at the next cluster round. However, in Figure 4.6a, we observe this can happen only in early epochs. However, as the training proceeds, we observe $f_{\theta_1}(x') < f_{\theta_2}(x')$ again, due to x' becoming overfitted to model 1. The sample x' is assigned to cluster 1 again as a result, making no improvement in cluster iterations. Note that this issue gets reduced when number of data m increases, as we see Figure 4.6b and Figure 4.6c. Figure 4.6c shows the plot of the same configuration with $m = 4000$, and shows that two model loss $f_{\theta_1}(x), f_{\theta_2}(x)$, are not crossing, meaning that the misclustered sample will be correctly clustered at any timestep of model training.

One may consider heuristics like applying early stopping based on validation stats. We find this method often works well at synthetic settings, but fails when training with real-world noisy data. The model should be trained to a level where correctly clustered data are well fit, and at the same time incorrectly clustered data are not overfit, therefore determining the optimal stopping point would be difficult without access to the ground truth cluster label. Finding the solution for this issue remains open.

Chapter 5

Conclusions and Future Works

5.1 Future work

Our framework has simple structure that clusters by loss given by the generative models trained with each cluster set. For the future research direction, we consider augmenting our method to involve more information. An interesting direction would be using more information than loss statistics for the cluster assignment stage, such as involving different aspects of the model (such as sample's activation on specific layer). Another way of improvement would be explicitly feeding supervision signal in training the model, such as maximizing the loss for the samples outside the cluster, which can help models in distinguishing samples inside the cluster from outside. We believe applying these methods improve our framework in model training and clustering, and paritally reduce the issue of cluster assignment converging.

5.2 Conclusions

In this report, we address clustering problems in a heterogeous federated learning setting. We propose an iterative algorithm and obtain convergence guarantees for strongly convex and smooth functions. In experiments, we achieve this via random initialization with multiple restarts, and we show that our algorithm works efficiently beyond the convex regime. We also apply our algorithm to settings with varying heterogeniety by training multiple generative models that captures each cluster, and assigns data a cluster membership by comparing the model's likelihoods, and evaluated its performance with MNIST and synthetic data.

Bibliography

[1] Sivaraman Balakrishnan, Martin J Wainwright, and Bin Yu. "Statistical guarantees for the EM algorithm: From population to sample-based analysis". In: *The Annals of Statistics* 45.1 (2017), pp. 77–120.

[2] Sebastian Caldas et al. "Leaf: A benchmark for federated settings". In: *arXiv preprint arXiv:1812.01097* (2018).

[3] Mathilde Caron et al. "Deep Clustering for Unsupervised Learning of Visual Features". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[4] Rich Caruana. "Multitask learning". In: *Machine learning* 28.1 (1997), pp. 41–75.

[5] Fei Chen et al. "Federated meta-learning with fast convergence and efficient communication". In: *arXiv preprint arXiv:1802.07876* (2018).

[6] Constantinos Daskalakis, Christos Tzamos, and Manolis Zampetakis. "Ten steps of EM suffice for mixtures of two Gaussians". In: *arXiv preprint arXiv:1609.00368* (2016).

[7] Ofer Dekel et al. "Optimal distributed online prediction using mini-batches". In: *Journal of Machine Learning Research* 13.Jan (2012), pp. 165–202.

[8] Don Kurian Dennis, Tian Li, and Virginia Smith. *Heterogeneity for the Win: One-Shot Federated Clustering*. 2021. arXiv: 2103.00697 [cs.LG].

[9] Wayne S DeSarbo and William L Cron. "A maximum likelihood methodology for clusterwise linear regression". In: *Journal of Classification* 5.2 (1988), pp. 249–282.

[10] Nat Dilokthanakul et al. *Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders*. 2017. arXiv: 1611.02648 [cs.LG].

[11] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. *Density estimation using Real NVP*. 2017. arXiv: 1605.08803 [cs.LG].

[12] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. "Personalized federated learning: A meta-learning approach". In: *arXiv preprint arXiv:2002.07948* (2020).

[13] James R Fienup. "Phase retrieval algorithms: a comparison". In: *Applied optics* 21.15 (1982), pp. 2758–2769.

[14] Avishek Ghosh and Kannan Ramchandran. "Alternating Minimization Converges Super-Linearly for Mixed Linear Regression". In: *arXiv preprint arXiv:2004.10914* (2020).

[15] Avishek Ghosh et al. “An Efficient Framework for Clustered Federated Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 19586–19597. URL: `https://proceedings.neurips.cc/paper/2020/file/e32cc80bf07915058ce90722ee17bb71-Paper.pdf`.

[16] Avishek Ghosh et al. “Robust federated learning in a heterogeneous environment”. In: *arXiv preprint arXiv:1906.06629* (2019).

[17] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. *Unsupervised Representation Learning by Predicting Image Rotations*. 2018. arXiv: `1803.07728 [cs.CV]`.

[18] Ian J Goodfellow et al. “An empirical investigation of catastrophic forgetting in gradient-based neural networks”. In: *arXiv preprint arXiv:1312.6211* (2013).

[19] Andrew Hard et al. “Federated learning for mobile keyboard prediction”. In: *arXiv preprint arXiv:1811.03604* (2018).

[20] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. “Low-rank matrix completion using alternating minimization”. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 2013, pp. 665–674.

[21] Yihan Jiang et al. “Improving federated learning personalization via model agnostic meta learning”. In: *arXiv preprint arXiv:1909.12488* (2019).

[22] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: `1312.6114 [stat.ML]`.

[23] Polina Kirichenko, Pavel Izmailov, and Andrew G Wilson. “Why Normalizing Flows Fail to Detect Out-of-Distribution Data”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 20578–20589. URL: `https://proceedings.neurips.cc/paper/2020/file/ecb9fe2fbb99c31f567e9823e884dbec-Paper.pdf`.

[24] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the National Academy of Sciences* 114.13 (2017), pp. 3521–3526.

[25] Jakub Konečnỳ et al. “Federated optimization: distributed machine learning for on-device intelligence”. In: *arXiv preprint arXiv:1610.02527* (2016).

[26] Alex Krizhevsky and Geoffrey Hinton. “Learning multiple layers of features from tiny images”. In: *Technical Report* (2009).

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[28] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[29] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[30] Dar-Shyang Lee. "Effective Gaussian mixture learning for video background subtraction". In: *IEEE transactions on pattern analysis and machine intelligence* 27.5 (2005), pp. 827–832.

[31] Liping Li et al. "RSA: Byzantine-Robust Stochastic Aggregation Methods for Distributed Learning from Heterogeneous Datasets". In: *arXiv preprint arXiv:1811.03761* (2018).

[32] Mu Li et al. "Scaling distributed machine learning with the parameter server". In: *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*. 2014, pp. 583–598.

[33] Tian Li et al. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (May 2020), pp. 50–60. ISSN: 1558-0792. DOI: `10.1109/msp.2020.2975749`. URL: `http://dx.doi.org/10.1109/MSP.2020.2975749`.

[34] Xiang Li et al. "On the convergence of fedavg on non-iid data". In: *arXiv preprint arXiv:1907.02189* (2019).

[35] Steven Liu et al. *Diverse Image Generation via Self-Conditioned GANs*. 2020. arXiv: `2006.10728 [cs.CV]`.

[36] S. Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: `10.1109/TIT.1982.1056489`.

[37] David Lopez-Paz and Marc'Aurelio Ranzato. "Gradient episodic memory for continual learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6467–6476.

[38] Yishay Mansour et al. "Three approaches for personalization with applications to federated learning". In: *arXiv preprint arXiv:2002.10619* (2020).

[39] Brendan McMahan and Daniel Ramage. *Federated Learning: Collaborative Machine Learning without Centralized Training Data*. `https://research.googleblog.com/2017/04/federated-learning-collaborative.html`. 2017.

[40] H Brendan McMahan et al. "Communication-efficient learning of deep networks from decentralized data". In: *arXiv preprint arXiv:1602.05629* (2016).

[41] Rick P Millane. "Phase retrieval in crystallography and optics". In: *JOSA A* 7.3 (1990), pp. 394–411.

[42] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. "Agnostic federated learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4615–4625.

[43] Sudipto Mukherjee et al. "ClusterGAN: Latent Space Clustering in Generative Adversarial Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 4610–4617. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/4385`.

[44] Praneeth Netrapalli, Prateek Jain, and Sujay Sanghavi. "Phase retrieval using alternating minimization". In: *Advances in Neural Information Processing Systems*. 2013, pp. 2796–2804.

[45] Lance Parsons, Ehtesham Haque, and Huan Liu. "Subspace Clustering for High Dimensional Data: A Review". In: *SIGKDD Explor. Newsl.* 6.1 (June 2004), pp. 90–105. ISSN: 1931-0145. DOI: `10.1145/1007730.1007731`. URL: `https://doi.org/10.1145/1007730.1007731`.

[46] Benjamin Recht et al. "Hogwild: A lock-free approach to parallelizing stochastic gradient descent". In: *Advances in Neural Information Processing Systems*. 2011, pp. 693–701.

[47] Danilo Jimenez Rezende and Shakir Mohamed. *Variational Inference with Normalizing Flows*. 2016. arXiv: `1505.05770 [stat.ML]`.

[48] Anit Kumar Sahu et al. "On the convergence of federated optimization in heterogeneous networks". In: *arXiv preprint arXiv:1812.06127* 3 (2018).

[49] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. "Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints". In: *arXiv preprint arXiv:1910.01991* (2019).

[50] Felix Sattler et al. "Robust and communication-efficient federated learning from non-iid data". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2019), pp. 3400–3413.

[51] Virginia Smith et al. "Federated multi-task learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4424–4434.

[52] E. G. Tabak and Cristina V. Turner. "A Family of Nonparametric Density Estimation Algorithms". In: *Communications on Pure and Applied Mathematics* 66.2 (2013), pp. 145–164. DOI: `https://doi.org/10.1002/cpa.21423`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpa.21423`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.21423`.

[53] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge University Press, 2019.

[54] Irène Waldspurger. "Phase retrieval with random gaussian sensing vectors by alternating projections". In: *IEEE Transactions on Information Theory* 64.5 (2018), pp. 3301–3312.

[55] Michel Wedel and Wagner A Kamakura. "Mixture regression models". In: *Market segmentation*. Springer, 2000, pp. 101–124.

[56] Blake Woodworth, Kumar Kshitij Patel, and Nathan Srebro. *Minibatch vs Local SGD for Heterogeneous Distributed Learning*. 2020. arXiv: `2006.04735 [cs.LG]`.

[57] Blake Woodworth et al. *Is Local SGD Better than Minibatch SGD?* 2020. arXiv: `2002.07839 [cs.LG]`.

[58] Lei Xu and Michael I Jordan. "On convergence properties of the EM algorithm for Gaussian mixtures". In: *Neural Computation* 8.1 (1996), pp. 129–151.

[59] Bowei Yan, Mingzhang Yin, and Purnamrita Sarkar. "Convergence of gradient EM on multi-component mixture of Gaussians". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6956–6966.

[60] Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. "Alternating minimization for mixed linear regression". In: *International Conference on Machine Learning*. 2014, pp. 613–621.

[61] Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. "Solving a mixture of many random linear equations by tensor decomposition and alternating minimization". In: *arXiv preprint arXiv:1608.05749* (2016).

[62] Dong Yin et al. "Learning mixtures of sparse linear regressions using sparse graph codes". In: *IEEE Transactions on Information Theory* 65.3 (2018), pp. 1430–1451.

[63] Chulhee Yun, Shashank Rajput, and Suvrit Sra. *Minibatch vs Local SGD with Shuffling: Tight Convergence Bounds and Beyond*. 2021. arXiv: `2110.10342 [cs.LG]`.

[64] Yue Zhao et al. *Federated Learning with Non-IID Data*. 2018. arXiv: `1806.00582 [cs.LG]`.

[65] Martin Zinkevich et al. "Parallelized stochastic gradient descent". In: *Advances in Neural Information Processing Systems*. 2010, pp. 2595–2603.