Tackling Personalized Federated Learning with Label Concept Drift via Hierarchical Bayesian Modeling

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated Learning (FL) is a distributed learning scheme to train a shared model across clients. One fundamental challenge in FL is that the sets of data across clients could be non-identically distributed, which is common in practice. Personalized Federated Learning (PFL) attempts to solve this challenge. Most methods in the literature of PFL focus on the data heterogeneity that clients differ in their label distributions. In this work, we focus on label concept drift which is a large but unexplored area. Firstly, we present a general framework for PFL based on hierarchical Bayesian inference. A global variable is introduced to capture the common trends of different clients and is used to augment the joint distribution of clients' parameters. Then we describe two concrete inference algorithms based on this framework. The first one finds a maximum a posteriori (MAP) solution for this augmented posterior distribution and adds little overhead compared with existing approaches. The second one further considers uncertainties of clients' parameters and different drift pattern across clients. We demonstrate our methods through extensive empirical studies on CIFAR100 and SUN397. Experimental results show our approach significantly outperforms the state of the art PFL when tackling the label concept drift across clients.

1 Introduction

Federated learning (FL) is a collaboration framework, in which clients optimize local models using their private data and share their model updates with the server, while server aggregates all uploaded local updates to provide a global shared model. Since the introduction of FL in [16] in 2016, there has been increasing attention for this distributed learning setting, as it turns out to be a general framework to address the concerns arise from many different areas. For example, medical data are sensitive. FL does not require direct data sharing and therefore helps to avoid conflicts with privacy regulations. FL also enables learning on large scale data, as clients use their local computational resources.

Despite the strengths, there are also many challenges in the application of FL. One of them is the statistical heterogeneity of client data sets. Since clients sit in various environments, their data exhibit certain concepts that correlate with the local environments and deviate from each other [9]. As summarized in [9], there exist different kinds of data heterogeneity. In this work, we focus on the so-called *label concept drift*. That is, the same label can have different features for different clients. A formal definition of *label concept drift* will be given in Section 2. Due to this concept drift, the aggregated global model is not optimal for every individual client.

Personalized federated learning (PFL) methods target data heterogeneity issue and intend to achieve a better local utility via personalized model at the clients' side. Many of them view the learning process of PFL through optimization [5; 6; 8; 17], while a few [1; 4; 2] try to understand PFL as a posterior Bayesian inference problem. However, in the current literature on PFL, the most methods

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

[1; 3; 8; 19; 17; 5; 6] focus on *label distribution skew*, i.e. clients have data corresponding to different labels. In this work we also take a Bayesian perspective, but aim at label concept drift and manage to capture the common trend while allowing each client to specialize in individual concepts. Such a perspective is very general and unifies several existing methods, as we will show later in Section 3,

We summarize the main contributions of this paper in the following:

- A general framework based on hierarchical Bayesian modeling for PFL under label concept drift is proposed.
- Two concrete inference algorithms are presented based on this framework.
- We demonstrate the effectiveness of our proposed approach through intensive empirical experiments.

Necessary background and notation that will be used in this paper are given in the next section. Section 3 presents our proposed framework. Two concrete inference algorithms based on this framework are discussed in Section 3.2 and Appendix C. Experimental results are presented in Section 4. Finally, we conclude in Section 5. Related works are presented in Appendix B.

2 Problem Formulation

This section sets up necessary background for PFL and the non-IID data across clients. Following [9], let the data distribution of client j be $P_j(x, y)$ and rewrite it as $P_j(y|x)P_i(x)$ and $P_j(x|y)P_i(y)$, different kinds of non-IID schemes can be identified. Most existing works focus on *label distribution skew*, that is, the label distributions $P_j(y)$ are varying in different clients but the conditional generating distributions $P(x|y)$ are assumed to be the same. In this work, we instead target *label concept drift*, which is another common type of non-IID scheme but is seldom studied in the current PFL literature. To further clarify *label concept drift*, we present a definition following [9].

Definition 2.1 (Label Concept Drift [9])**.** Let J be the number of clients, label concept drift indicates that the conditional generating distributions $\{P_j(x|y)\}_{j=1}^J$ are different across different clients, but marginal distributions $\{P_j(y)\}_{j=1}^J$ are the same.

A concrete example of *label concept drift* , e.g. due to geographical discrepancy, is that of different species of dogs that look different, such as the Labrador Retriever (most popular dog in the United States) vs. a Border Collie (most popular dog in Europe). Instead of using a single global model as in FL, in PFL we aim to solve the following minimization problem:

$$\min_{\boldsymbol{w}_{1:J} \in \mathbb{R}^d} f(\boldsymbol{w}_{1:J}; \mathcal{D}) := \frac{1}{J} \sum_{j=1}^{J} f_j(\boldsymbol{w}_j; \mathcal{D}_j), \tag{1}$$

where J is the number of clients, $\mathcal{D}_j$ is the set of data available in client j, $\boldsymbol{w}_{1:J}$ is a shorthand for the set of parameters $\{\boldsymbol{w}_1, \cdots, \boldsymbol{w}_J\}$, $\boldsymbol{w}_j$ is the personalized parameter for the j-th client and $f_j(\boldsymbol{w}_j; \mathcal{D}_j)$ is the empirical average loss function of the j-th client:

$$f_j(\boldsymbol{w}; \mathcal{D}_j) = \frac{1}{n_j} \sum_{i=1}^{n_j} l(\boldsymbol{x}_i^{(j)}, y_i^{(j)}; \boldsymbol{w}), \tag{2}$$

where $(\boldsymbol{x}_i^{(j)}, y_i^{(j)}) \in \mathcal{D}_j$ is one data point of client j, $l(\cdot, \cdot|\boldsymbol{w})$ is the loss function using weight parameter $\boldsymbol{w}$ and $n_j := |\mathcal{D}_j|$ is the number of data points on the j-th client.

3 Methods

In this section, firstly we describe our proposed Bayesian framework for PFL under the label concept drift. Then two concrete inference algorithms based on this framework are presented.

3.1 The Augmented Joint Distribution

To develop a Bayesian framework, we need to obtain a posterior distribution for parameters which we are interested in. In the context of PFL, the target is the posterior distribution $p(\boldsymbol{w}_j|\mathcal{D})$ of $\boldsymbol{w}_j$ for any

client j. Once we have obtained this posterior distribution, any deductive reasoning can be conducted. For example, to make predictions on the j-th client, we integrate out $\boldsymbol{w}_j$ w.r.t. $p(\boldsymbol{w}_j|\mathcal{D})$:

$$p_j(y_*|\boldsymbol{x}_*, \mathcal{D}) = \int p_j(y_*|\boldsymbol{x}_*, \boldsymbol{w}_j) p(\boldsymbol{w}_j|\mathcal{D}) d\boldsymbol{w}_j, \tag{3}$$

where $\boldsymbol{x}_*$ is a new input, y_* is our prediction for $\boldsymbol{x}_*$, $p_j(y_*|\boldsymbol{x}_*, \boldsymbol{w}_j)$ is the likelihood function and is dependent on the type of data on client j.

One way to obtain this posterior distribution $p(\boldsymbol{w}_j|\mathcal{D})$ is by performing Bayesian inference on the j-th client locally, i.e. $p(\boldsymbol{w}_j|\mathcal{D}) := p(\boldsymbol{w}_j|\mathcal{D}_j)$. Given a vague prior $p(\boldsymbol{w}_j)$, one disadvantage of this approach is the variance of $p(\boldsymbol{w}_j|\mathcal{D}_j)$ could be high if the number of data points on client j is small. However in the context of Bayesian network models, such a vague prior is almost unavoidable [21].

In another way, since all clients are running similar tasks, $\mathcal{D}_{\{1,\cdots,J\}\setminus j}$ should be able to provide information to form the posterior of $\boldsymbol{w}_j$. In a distributed learning scheme like FL, the j-th client has no access to $\mathcal{D}_{\{1,\cdots,J\}\setminus j}$ and it is impossible to obtain the posterior $p(\boldsymbol{w}_j|\mathcal{D})$ directly. To overcome this restriction, we introduce a *center model* $\boldsymbol{w}$ such that all $\boldsymbol{w}_{1:J}$ depend on $\boldsymbol{w}$ and $\boldsymbol{w}$ captures the correlations between different clients, namely the common trend of the task. This also implies the conditional independence between $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$:

$$p(\boldsymbol{w}_i|\boldsymbol{w}) p(\boldsymbol{w}_j|\boldsymbol{w}) = p(\boldsymbol{w}_i, \boldsymbol{w}_j|\boldsymbol{w}). \tag{4}$$

The conditional distribution $p(\boldsymbol{w}_j|\boldsymbol{w})$ enables each client to be specialized in individual concept based on the common trend carried by the center model $\boldsymbol{w}$. Using Bayes' theorem, the augmented posterior distribution of $\{\boldsymbol{w}, \boldsymbol{w}_{1:J}\}$ is proportional to the product of the prior and the likelihood function:

$$\begin{aligned} p(\boldsymbol{w}, \boldsymbol{w}_{1:J}|\mathcal{D}) &\propto p(\boldsymbol{w}, \boldsymbol{w}_{1:J}) p(\mathcal{D}|\boldsymbol{w}, \boldsymbol{w}_{1:J}) \\ &\overset{4}{=} p(\boldsymbol{w}) \prod_{j=1}^{J} p(\boldsymbol{w}_j|\boldsymbol{w}) \exp\left(-f_j(\boldsymbol{w}_j; \mathcal{D}_j)\right), \end{aligned} \tag{5}$$

where $p(\boldsymbol{w})$ is the prior of the introduced center model, $f_j(\boldsymbol{w}_j|\mathcal{D}_j)$ is defined in Equation (2) and is proportional to the negative of the data log-likelihood on client j. From the above augmented joint distribution, we see the introduction of the center model $\boldsymbol{w}$ enables complicated communication across clients. The marginalized joint distribution $p(\boldsymbol{w}_{1:J}|\mathcal{D}) = \int p(\boldsymbol{w}, \boldsymbol{w}_{1:J}|\mathcal{D}) d\boldsymbol{w}$ can thus be flexible. Based on the above augmented model, two inference algorithms could be used:

- **Maximum a Posteriori Probability (MAP):** MAP seeks a maximizer to the unnormalized posterior and avoids the intractable integration of the model evidence. In Appendix C, we will describe an efficient algorithm to optimize $\boldsymbol{w}_{1:J}$. The prediction in Equation (3) is simply $p_j(y_*|\boldsymbol{x}_*, \boldsymbol{w}_j^*)$ in this case, since $p(\boldsymbol{w}_j|\mathcal{D})$ is a unit mass concentrated at the optimized $\boldsymbol{w}_j^*$.
- **Maximize the Marginal Likelihood:** Finding the MAP in PFL is easy to implement and efficient. But there exist two limitations. The first one is it can be difficult to select a set of hyper-parameters for the conditional priors. The second one is that the prediction $p_j(y_*|\boldsymbol{x}_*, \boldsymbol{w}_j^*)$ made on client j does not consider the uncertainties of $\boldsymbol{w}_j$. To mitigate these two limitations, we present an algorithm based on the principle of maximizing the marginal likelihood in Section 3.2. The optimization will be conducted using variational expectation maximization.

3.2 Variational Expectation Maximization

Due to space limitations, our MAP approach is given in Appendix C, where we also discuss the connection between our approach and FedAvg [15] FedProx [12], as well as pFedPer [3]. In MAP, the overall optimization is efficient and easy to implement. However, the assumption $p(\boldsymbol{w}_j|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}_j \mid \boldsymbol{w}, \rho_j^2 \boldsymbol{I})$ arise a group of hyperparameters ρ_j^2 which is hard to set. The point estimation of $\boldsymbol{w}_j$ can also be unreliable, e.g. when the number of data points on the jth client is small and the evidence is not strong enough. To address these issues, we introduce factorized variational approximation $q(\boldsymbol{w}_{1:J}) := \prod_{j=1}^{J} q_j(\boldsymbol{w}_j)$ to the true posterior distribution $p(\boldsymbol{w}_{1:J}|\mathcal{D})$. In this work, the axis-aligned multivariate Gaussian is used as the variational family, that is, $q_j(\boldsymbol{w}_j) = \mathcal{N}(\boldsymbol{w}_j|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ and $\boldsymbol{\Sigma}$ is a

diagonal matrix. To optimize these approximations $\{q_j(\boldsymbol{w}_j)\}_{j=1}^J$, we maximize the evidence lower bound (ELBO) of the marginal likelihood:

$$\text{ELBO}\left(q(\boldsymbol{w}_{1:J}), \rho_{1:J}^2, \boldsymbol{w}\right) = \sum_{j=1}^{J} \mathbb{E}_{q(\boldsymbol{w}_j)}[\log p(\mathcal{D}_j|\boldsymbol{w}_j)] - \text{KL}[q(\boldsymbol{w}_j) \parallel p(\boldsymbol{w}_j|\boldsymbol{w}, \rho_j^2)]. \tag{6}$$

The above ELBO can be optimized using Variational Expectation Maximization (VEM) through blockwise coordinate descent. To obtain the variational approximation $q(\boldsymbol{w}_j)$, we only need to maximize $\mathbb{E}_{q(\boldsymbol{w}_j)}[\log p(\mathcal{D}_j|\boldsymbol{w}_j)] - \text{KL}[q(\boldsymbol{w}_j) \parallel p(\boldsymbol{w}_j|\boldsymbol{w}, \rho_j^2)]$ using the local set of data on client j.

After these local approximations have been formed, clients upload their own variational parameters to the server. Given these parameters, the server also tries to optimize the ELBO in Equation (6) by updating the center model $\boldsymbol{w}$. Simplifying Equation (6), we obtain the objective function on the server:

$$\begin{aligned}
\text{ELBO}(\rho_{1:J}^2, \boldsymbol{w}) &= \sum_{j=1}^{J} \mathbb{E}_{q(\boldsymbol{w}_j)}[\log p(\mathcal{D}_j, \boldsymbol{w}_j)] - \mathbb{E}_{q(\boldsymbol{w}_j)}[\log q(\boldsymbol{w}_j)] \\
&\propto \sum_{j=1}^{J} \mathbb{E}_{q(\boldsymbol{w}_j)}[\log p(\mathcal{D}_j|\boldsymbol{w}_j, \boldsymbol{w}) + \log p(\boldsymbol{w}_j|\boldsymbol{w}, \rho_j^2)] \\
&\propto \sum_{j=1}^{J} \mathbb{E}_{q(\boldsymbol{w}_j)}[\log p(\boldsymbol{w}_j|\boldsymbol{w}, \rho_j^2)]. \qquad (7)
\end{aligned}$$

The last line holds because $\log p(\mathcal{D}_j|\boldsymbol{w}_j, \boldsymbol{w}) = \log p(\mathcal{D}_j|\boldsymbol{w}_j)$ by assumption and $\log p(\mathcal{D}_j|\boldsymbol{w}_j)$ does not depend on $\boldsymbol{w}$ and $\rho_{1:J}^2$. Setting the gradient of Equation (7) w.r.t. $\boldsymbol{w}$ and $\rho_{1:J}^2$ to be zero, we obtain closed form for these parameters:

$$\boldsymbol{w}^* = \frac{\sum_{j=1}^{J} \tau_j \boldsymbol{\mu}_j}{\sum_{j=1}^{J} \tau_j}, \tau_j := 1/\rho_j^2; \qquad D \cdot \rho_j^2 = \text{Tr}(\boldsymbol{\Sigma}_j) + \parallel \boldsymbol{\mu}_j - \boldsymbol{w} \parallel^2, \tag{8}$$

where D is the dimension of $\boldsymbol{w}$ and $\text{Tr}(\boldsymbol{\Sigma}_j)$ is the trace of the variational variance-covariance parameter. We summarize the update rules in the following Algorithm 1.

Algorithm 1 Variational Expectation Maximization for PFL (pFedVEM)

Input: T rounds, $\boldsymbol{\mu}_{1:J}^0$, $\boldsymbol{\Sigma}_{1:J}^0$, $\boldsymbol{w}^0$, $\rho_{1:J}$, sampling ratio r
Output $\boldsymbol{w}_{1:J}^T$

```
for t = 0 to T − 1 do
    Server sends w^t to all clients
    ▷ clients update their personalized variational distributions ◁
    for j = 1, ..., J do
        (μ_j^{t+1}, Σ_j^{t+1}) ∈ arg min_{(μ_j, Σ_j)} E_{q(w_j)}[log p(D_j|w_j)] − KL[q(w_j) ∥ p(w_j|w, ρ_j^2)]
        Update ρ_j^{t+1} by ρ_j^2 = (Tr(Σ_j^{t+1}) + ∥ μ_j^{t+1} − w^t ∥^2)/D
    Server select a random subset of clients I_t of size ⌊J × r⌋
    Each client j ∈ I_t sends its updated variational parameters (μ_j^{t+1}, Σ_j^{t+1}) and ρ_j^{t+1} to the server
    ▷ server optimizes the center model ◁
    w^{t+1} = (Σ_{j∈I_t} τ_j^{t+1} μ_j^{t+1}) / (Σ_{j∈I_t} τ_j^{t+1})
```

4 Experiments

To evaluate our methods, we target image classification tasks and compare our methods against baselines under various settings. We present results of the following frameworks: 1) Local: clients are training based on their local data, no collaboration across clients; 2) FedAvg [15]: all clients rely on an aggregated global model; 3) pFedPer [3]: the network consists of representation model

and linear classifier, the representation model participates in collaboration while every client trains its own linear classifier; 4) pFedHN [18]: the server uses a hypernetwork and client-specific latent variable to generate a personal network for each client. 5) pFedMe [20]: bi-level optimization of local model and the global model is conducted using Moreau envelops.[1]

When applying our method, to make the optimization w.r.t. $q(\boldsymbol{w}_j)$ more tractable we follow the treatment in pFedPer, which considers the entire network comprising a representation model and a linear classifier, we only personalize the linear classifier layer while let the representation model to be trained via FedAvg to generate a general representation. Unifying the representation model could have additional profit for the representation model as it has many parameters while individual client has possibly limited data points. We conduct experiments on two variants of our method: 1) pFedMap: our maximum posterior method under non-informative prior of $\boldsymbol{w}$ and normal distributed $p(\boldsymbol{w}_j|\boldsymbol{w})$ with predefined variance ρ_j. We set $\rho_{1:J}$ to be the same to avoid a prohibitive tuning workload. The resulting algorithm looks similar to FedProx, but instead of the center model $\boldsymbol{w}$ personalized linear classifiers $\boldsymbol{w}_{1:J}$ are used for inference. 2) pFedVem: our approach of variational expectation maximization. Every client has an estimation of the conditional distribution $p(\boldsymbol{w}_j|\boldsymbol{w})$ over its personalized linear classifier and the center model $\boldsymbol{w}$ of linear classifiers $\boldsymbol{w}_{1:J}$ are aggregated w.r.t. the estimated distributions $\{p(\boldsymbol{w}_j|\boldsymbol{w})\}_{j=1}^{J}$. Our running algorithms in the experiments are detailed in Appendix E.

4.1 Training protocol

To better present label concept drift, i.e. varying $P_j(x|y)$, we use hierarchical datasets CIFAR100 [10] and SUN397 [22], which contain superclasses and subclasses. We set the classification task to be superclass prediction, while for every client the data of each superclass is sampled from a random subclass and hence label concept drift is induced. To get closer to reality, we allow the number of observations by each client to be different. In this work, we only consider balanced data (the amount of data for every label is equal), the details of data sampling are given in Appendix D. When running the experiments the number of communication rounds is set to be 100.[2] We adopt the communication protocol proposed in pFedMe [20], i.e. at the beginning of each round, the server broadcasts the aggregated model to all clients, while only a subset of clients send their parameters back to the server. Each client has a probability of 0.1 to be sampled, we evaluate all frameworks with the number of clients $C \in \{50, 100, 200\}$.

We use a CNN network with five convolutional layers followed by a fully connected layer. In case the framework consists of two speparate components, the convolutional layers block is used as the representation model and the fully connected layer as the linear classifer. For all settings and frameworks, we perform a grid search for hyperparameters. All experiments have been repeated for five times using the same group of five random seeds. We then compute the mean and standard error. We run all experiments on a cluster within the same container environment. More details of experiment setup are given in Appendix F.

4.2 Results

Average performance One important metric to evaluate FL frameworks is the average performance of all clients. We evaluate all the frameworks over various settings, the results are given in Table 1. We observe that: 1) Although the label concept drifts among clients, FedAvg still outperforms Local. When the expected *# Sample/Clients* is large, e.g. on CIFAR100, FedAvg even outperforms some pFed frameworks, implying that although label concept drift across clients using a single global model is still adequate when all participants have sufficient data; 2) Our pFedMAP, which is lightweight and easy to implement, is competitive compared with previous PFL methods; 3) Our pFedVEM always achieves the best accuracy and performs significantly better than baselines over all settings.

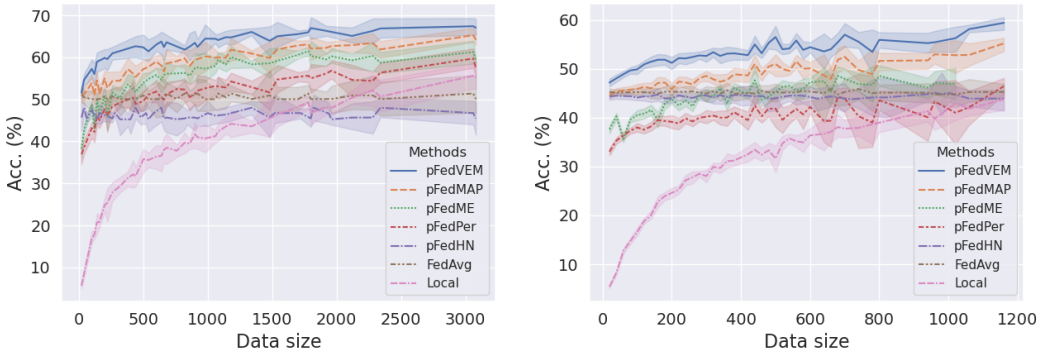
Accuracy over data size FL is a collaboration framework, to attract more clients joining in the collaboration, we need to provide sufficient incentive. Utility gain is a major motivation. However,

[1] We conducted experiments for pFedGP [1] but ran into many errors when clients have only a few data points. So in this work we skip the comparison with this baseline.

[2] 1) When conducting the local framework, we train the model for 100 epochs. 2) pFedHN limits the server to connect a single client at each iteration, so we let it connect with sampled clients in series at each round. Note that in reality this leads to a significant higher time complexity than other frameworks.

	CIFAR100			SUN397		
# Clients	50	100	200	50	100	200
# Samples/Clients	1000	500	250	102	102	102
Local	36.4 ± 0.3	29.2 ± 0.4	22.8 ± 0.3	62.7 ± 0.5	62.6 ± 0.6	61.1 ± 0.7
FedAvg	52.2 ± 0.1	49.4 ± 0.8	45.8 ± 0.1	72.8 ± 0.1	75.4 ± 0.2	76.4 ± 0.1
pFedPer	50.9 ± 0.5	44.6 ± 0.3	38.9 ± 0.4	84.4 ± 0.3	83.9 ± 0.2	83.7 ± 0.2
pFedME	54.9 ± 0.4	48.9 ± 0.5	42.2 ± 0.4	85.1 ± 0.5	87.1 ± 0.1	87.0 ± 0.2
pFedHN	47.6 ± 0.3	46.2 ± 0.2	45.7 ± 0.3	75.9 ± 0.5	74.5 ± 0.3	78.3 ± 0.3
Ours						
pFedMAP	58.8 ± 0.3	53.1 ± 0.3	47.3 ± 0.4	84.0 ± 1.1	86.6 ± 0.4	88.0 ± 0.1
pFedVEM	$\mathbf{62.6 \pm 0.3}$	$\mathbf{57.4 \pm 0.5}$	$\mathbf{51.5 \pm 0.5}$	$\mathbf{87.7 \pm 0.1}$	$\mathbf{88.6 \pm 0.2}$	$\mathbf{88.5 \pm 0.4}$

Table 1: Test accuracy (% $\pm$ SEM) over 50, 100, 200 clients on CIFAR100 and SUN397. # Samples/Clients indicates the *expected data size of a client*.



(a) 50 clients

(b) 200 clients

Figure 1: Test accuracy (% $\pm$ SEM) over individual data size on CIFAR100. This figure demonstrates how the clients with less local data perform vs. the clients with more data after a collaboration. Clients with different amounts of local data can always obtain the most utility gain using our pFedVEM.

the utility gain could vary for different clients, especially when the clients have different amount of local data. To investigate this, we plot the accuracy over individual data size on CIFAR100 with 50 and 200 clients, see Figure 1. Comparing FL frameworks with Local, we observe: 1) Clients with less data can gain the most by joining collaboration; 2) However, using baseline frameworks, clients with comparatively large data size can not get much benefit from or even loss utility by joining the collaboration. 3) Clients with various data size are motivated to join our pFedMAP and pFedVEM. Specifically, pFedVEM provides significantly more utility gain for all clients. More accuracy over data size plots and posterior distribution estimation plots are available in Appendix G.

5 Conclusions

In this paper, we addressed the problem of personalized federated learning when label concept distributions differ between clients. We have developed an evaluation framework for PFL with concept drift, and proposed a general framework for PFL via hierarchical Bayesian modeling. Our method pFedVEM using variational inference yields consistently superior performance to main competing frameworks.

References

[1] Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized federated learning with gaussian processes. *Advances in Neural Information Processing Systems*,

2021.

[2] Maruan Al-Shedivat, Jennifer Gillenwater, Eric Xing, and Afshin Rostamizadeh. Federated Learning via Posterior Averaging: A New Perspective and Practical Algorithms. *arXiv:2010.05273 [cs, stat]*, January 2021.

[3] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated Learning with Personalization Layers. *arXiv:1912.00818 [cs, stat]*, December 2019.

[4] Luca Corinzia, Ami Beuret, and Joachim M. Buhmann. Variational Federated Multi-Task Learning. *arXiv:1906.06268 [cs, stat]*, February 2021.

[5] Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. Personalized Federated Learning with Moreau Envelopes. *arXiv:2006.08848 [cs, stat]*, March 2021.

[6] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. URL `http://arxiv.org/abs/2002.07948`.

[7] Filip Hanzely and Peter Richtárik. Federated Learning of a Mixture of Global and Local Models. *arXiv:2002.05516 [cs, math, stat]*, February 2021.

[8] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. URL `http://arxiv.org/abs/1909.12488`.

[9] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and Open Problems in Federated Learning. *arXiv:1912.04977 [cs, stat]*, March 2021.

[10] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[12] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization in Heterogeneous Networks, April 2020.

[13] Christos Louizos, Matthias Reisser, Joseph Soriaga, and Max Welling. An Expectation-Maximization Perspective on Federated Learning. *arXiv:2111.10192 [cs, stat]*, November 2021.

[14] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three Approaches for Personalization with Applications to Federated Learning. *arXiv:2002.10619 [cs, stat]*, July 2020.

[15] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[16] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv:1602.05629 [cs]*, February 2017.

[17] Sotirios Nikoloutsopoulos, Iordanis Koutsopoulos, and Michalis K. Titsias. Personalized Federated Learning with Exact Stochastic Gradient Descent. *arXiv:2202.09848 [cs]*, February 2022.

[18] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[19] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized Federated Learning using Hypernetworks. *arXiv:2103.04628 [cs]*, March 2021.

[20] Canh T. Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. In *Advances in Neural Information Processing Systems*, 2020.

[21] Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How Good is the Bayes Posterior in Deep Neural Networks Really? *arXiv:2002.02405 [cs, stat]*, February 2020.

[22] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

A Ethics statement

There are generally positive societal impacts of personalized federated learning due to increased privacy. Potential negative impacts may result from the application of such models to e.g. social network preference models if it would result in e.g. bubble effects. This, however, is rather a result of the way the method would be applied than the method itself, and the ethical concerns are likely the same as for any machine learning model.

B Related Work

Federated Learning. Since the introduction of *FedAvg*, which is a method for FL using deep neural networks, many methods [16; 7; 12] have been proposed to improve it from different perspectives. In the literature of FL, most existing approaches formulate the training of FL as an optimization problem. *FedAvg* [16] updates the shared model by averaging client updates and these updates come from local Stochastic Gradient Descent (SGD). *L2GD* [7] learns a mixture of the global shared model and local models by formulating FL as a quadratic penalized optimization model. A similar solution is also used by *FedProx* [12], which adds a proximal term to a local objective. Recently, there have been several works [2; 13] which view the federated training process through the lens of Bayesian posterior inference. *FedPA* [2] shows there is an equivalence between the inference of the posterior mode and the federated optimization under the uniform prior. [13] views the federated optimization as a hierarchical latent model and it has been shown that *FedAvg* is a specific instance under this viewpoint.

Personalized Federated Learning. One fundamental challenge in FL is how to deal with non-IID data [9]. Although it has been empirically shown that *FedAvg* works well in moderate non-IID situations [16], it could diverge when the data distribution is highly skewed [17; 12]. A standard way to cope with this data heterogeneity is to learn a personalized model per client [3; 14; 5; 19; 4]. *pFedPer* [3] introduces a personalization layer for each client. To learn a good image representation, all clients share a base model. [14] describes a systematic way to study personalization in FL and presents several algorithms. *pFedMe* [5] uses the Moreau envelope to regularize a client's loss function. This approach shares some ideas with [12; 7], but targets PFL instead of FL. Another work on PFL is *pFedHN* [19] and hypernetworks are used to generate a set of personalized models. Instead of viewing PFL as an optimization process, *VIRTUAL* [4] approaches this task using a Bayesian method and learns a client model through approximated variational inference.

C Maximum a Posteriori

To find the MAP solution for the unknown set of parameters $\{\boldsymbol{w}, \boldsymbol{w}_{1:J}\}$, we need to maximize the RHS of Equation (5). Before an update scheme for $\{\boldsymbol{w}, \boldsymbol{w}_{1:J}\}$ can be derived, it is necessary to specify the concrete forms for both the prior distribution $p(\boldsymbol{w})$ and the conditional density $p(\boldsymbol{w}_j|\boldsymbol{w})$. In this work, we assume an isotropic Gaussian conditional prior $p(\boldsymbol{w}_j|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}_j \mid \boldsymbol{w}, \rho_j^2 \boldsymbol{I})$, where ρ_j^2 is the variance of this distribution. A Gaussian conditional implies all clients' parameters are close to this center model $\boldsymbol{w}$. In PFL, this is a reasonable assumption since all clients are running similar tasks. Besides, this enables a closed form for updating $\boldsymbol{w}$. In Section 4, it will be shown that the isotropic Gaussian assumption works well in practice. To make the following notation clear, we also use an alternative parameterization for this conditional $p(\boldsymbol{w}_j|\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}_j \mid \boldsymbol{w}, \tau_j^{-1} \boldsymbol{I})$, where $\tau_j = 1/\rho_j^2$ is the precision parameter. In the following, we consider two different priors of $\boldsymbol{w}$ and the derive corresponding MAP updating rules for $\boldsymbol{w}$.

Under the non-informative prior of $p(\boldsymbol{w})$, it is easy to see that the MAP w.r.t. $\{\boldsymbol{w}, \boldsymbol{w}_{1:J}\}$ is the solution to the following quadratic penalized optimization problem:

$$\arg\min_{\boldsymbol{w}, \boldsymbol{w}_{1:J}} \sum_{j=1}^{J} f_j(\boldsymbol{w}_j; \mathcal{D}_j) + \frac{1}{2}\tau_j \left\| \boldsymbol{w} - \boldsymbol{w}_j \right\|^2 . \tag{9}$$

Such an optimization problem can be solved through the blockwise coordinate descent method.

- Given $\boldsymbol{w}$, clients optimize their personalized parameters $\boldsymbol{w}_{1:J}$ using their private data set. This process can be done in parallel.

- Given a set of clients' parameters $\boldsymbol{w}_{1:J}$, a central server optimizes the introduced center model $\boldsymbol{w}$. This does not need any private data.

To optimize the personalized parameter of the j-th client, we simply minimize $f_j(\boldsymbol{w}_j|\mathcal{D}_j) + \frac{1}{2\rho_j^2}\|\boldsymbol{w} - \boldsymbol{w}_j\|^2$ since the all terms in Equation (9) are uncorrelated given $\boldsymbol{w}$. Setting the gradient of the objective function Equation (9) w.r.t. $\boldsymbol{w}$ to zero, a closed form for $\boldsymbol{w}$ can be obtained:

$$\boldsymbol{w}^* = \frac{\sum_{j=1}^{J} \tau_j \boldsymbol{w}_j}{\sum_{j=1}^{J} \tau_j}. \tag{10}$$

We see the optimized center model $\boldsymbol{w}^*$ is a weighted average of clients' personalized parameters, and this set of weights is equal to the set of precision parameters of our conditional priors. This solution for $\boldsymbol{w}$ can be interpreted intuitively. The more we trust a client, the larger influence it will have on the final solution. This also gives us a way to set the hyper-parameters $\tau_{1:J}$. the MAP method unifies several previous works, in particular:

FedAvg If we let a variable $\tau \to 0$, while set $\tau_j = \tau \cdot n_j$, the resulting algorithm approximates FedAvg, since $\tau \cdot n_j(\boldsymbol{w} - \boldsymbol{w}_j) \approx 0$ and $\boldsymbol{w}^* = \sum_{j=1}^{J} n_j \boldsymbol{w}_j / \sum_{j=1}^{J} n_j$.

pFedPer [3] pFedPer regards a network as a representation model attached with a personalized linear classifier. The representation model is updated via FedAvg while for the linear classifer MLE is used which is same as setting $\tau_{1:J} \to 0$ and abandon the use of $\boldsymbol{w}$.

FedProx [12] If we set a hyperparameter τ and let $\tau_{1:J} = \tau$, the resulting objective is FedProx. We note that for FedProx clients use $\boldsymbol{w}$ instead of $\boldsymbol{w}_j$ for inference, naturally the local performance is expected to be worse than using $\boldsymbol{w}_j$ respectively.

The following Algorithm 2 summarizes the optimization steps under the non-informative prior.

Algorithm 2 Maximum a Posteriori for PFL (pFedMAP)

Input: T rounds, $\boldsymbol{w}_{1:J}^0$, $\boldsymbol{w}^0$, $\tau_{1:J}$, sampling ratio r
Output $\boldsymbol{w}_{1:J}^T$
for $t = 0$ to $T-1$ **do**
  Server sends $\boldsymbol{w}^t$ to all clients
  ▷ *clients update their personalized parameters* ◁
  for $j = 1, \ldots, J$ **do**
    $\boldsymbol{w}_j^{t+1} \in \arg\min_{\boldsymbol{w}_j} f_j(\boldsymbol{w}_j; \mathcal{D}_j) + \frac{\tau_j}{2}\|\boldsymbol{w}^t - \boldsymbol{w}_j\|^2$
  Server select a random subset of clients $\mathcal{I}_t$ of size $\lfloor J \times r \rfloor$
  Each client $j \in \mathcal{I}_t$ sends its updated $\boldsymbol{w}_j^t$ to the server
  ▷ *server optimizes the center model* ◁
  $\boldsymbol{w}^{t+1} = \frac{\sum_{j \in \mathcal{I}_t} \tau_j \boldsymbol{w}_j^t}{\sum_{j=1}^{J} \tau_j}$

D Data partition and preprocessing

PFL considers that in reality clients sit in various environments. As a result their data are expected to contain different submodes or namely concept drift, resulting in non-identical data distribution among clients. Previous works split the dataset by labels and distribute data corresponding to a subset of labels to different clients causing varying $p(y_j)$. However, such a method cannot simulate concept drift among clients where $\forall j = 1 \ldots J$, $p(y_j) = p(y)$ while $p(x_j|y)$ varies.

To better present the personalization setting with concept drift, we use hierarchical datasets, namely datasets having superclasses and subclasses. The task performed by the clients is superclass prediction, while for every client the data of each superclass is sampled from a random subclass. Moreover, we allow the number of observations by each client to be different. In this work, we only consider balanced data (the amount of data for every label is equal). We use two datasets as described below.

CIFAR100 [10] This dataset has 20 superclasses, each contains five subclasses, e.g. "trees" includes "pine", "maple", etc.. It consists of 50,000 training data and 10,000 test data. We partition all training data by random slicing into J parts and distribute them to clients, thus the collection of data participated in the collaboration is a fixed set, we can summarize the results of different partitions and evaluate the performance of the framework. When conducting our evaluation, clients will be tested on the 20 subclasses it observes.

However, since we require that the individual data size is random while for every client, each label contains the same amount of data, the intersection of any two clients' data is empty (sampling without replacement) and all data in the training set need to be consumed, the partition turns out to be an integer programming problem which is NP-complete. To work around this and efficiently generate the clients' data, we take the following sampling procedure, which satisfies all the requirements with some loss of randomness:

1. Divide clients into G groups so that one group has five clients, each group contains all 100 subclasses, and each client within the group randomly samples 20 subclasses without replacement.
2. Randomly generate a integer l_j for the group j satisfying $\sum_{j=1}^{G} l_j = 500$.
3. For group $j = 1..G$, every client randomly samples l_j data points without replacement for each of its subclasses.

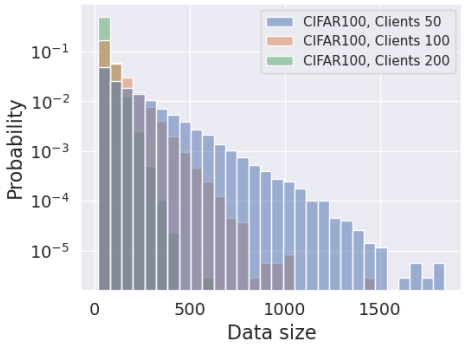


Figure 2: Data size distribution of CIFAR100, generated by Monte Carlo simulation.

The resulting data size distribution thus approximates an exponential distribution, as illustrated by Figure 2. Note that with more clients the expected amount of data per client reduces.

SUN397 [22] This scene category dataset is arranged in a 3-level tree, we use the first level for prediction, which contains three classes: "indoor", "manmade outdoor", "nature outdoor". Each first level node contains a variable number of leaf nodes and data points. There are in total 908 leaf nodes with more than 100,000 data points. We split 30 data points from each leaf node for testing and sample the training data for each client w.r.t. a uniform distribution $U(1, 67)$ from the rest, so the minimal data size is 3 for three superclasses and the maximal data size is 201. Note that, with more clients the expected data per client remains unchanged while with fewer clients less total data will be involved in collaboration. . The hierarchy has duplicated leaf nodes due to confusion in classification, e.g. "hayfield" could be considered as nature or manmade, we remove such data in our experiment. When conducting our evaluation, clients will be tested on the three leaf nodes it observes.

When generating client data, we use the following sampling procedure:

1. For the client j, we sample a number l_j w.r.t. a uniform distribution $U(0, 67)$ to determine the data per class.
2. For every class in client j, we randomly choose a leaf node that has more than l_j data and sample l_j data without replacement.

Based on this sampling procedure, the collection of all data participated in the collaboration varies for different random seeds and could result in different data distribution, so the achieved accuracy cannot be compared with each other. In our experiment on the dataset SUN397, we set the random seed of data sampling to be 42, while introducing randomness to parameter initialization and others.

E Running algorithms in the experiments

We present pseudo-code of our running pFedMAP in Algorithm 3, and running pFedVEM in Algorithm 4.

Algorithm 3 Running pFedMAP

Input: T rounds, linear classifiers $\boldsymbol{w}_{1:J}^0$, $\boldsymbol{w}^0$, representation model $\boldsymbol{\theta}^0$, penalty term λ, sampling ratio r

Output $\boldsymbol{w}_{1:J}^T$; $\boldsymbol{w}^T$; $\boldsymbol{\theta}^T$

for $t = 0$ to $T-1$ **do**
  Server sends $\boldsymbol{w}^t, \boldsymbol{\theta}^t$ to all clients
  ▷ *clients update their personalized parameters* ◁
  for $j = 1, \ldots, J$ **do**
    $\boldsymbol{w}_j^{t+1} \in \arg\min_{\boldsymbol{w}_j} f_j(\boldsymbol{w}_j; \boldsymbol{\theta}^t; \mathcal{D}_j) + \lambda \left\| \boldsymbol{w}^t - \boldsymbol{w}_j \right\|^2$
  ▷ *clients update the representation model* ◁
  for $j = 1, \ldots, J$ **do**
    $\boldsymbol{\theta}_j^{t+1} \in \arg\min_{\boldsymbol{\theta}_j} f_j(\boldsymbol{w}_j^{t+1}; \boldsymbol{\theta}_j; \mathcal{D}_j)$
  Server select a random subset of clients $\mathcal{I}_t$ of size $\lfloor J \times r \rfloor$
  Each client $j \in \mathcal{I}_t$ sends its updated $\boldsymbol{w}_j^t$ to the server
  ▷ *server optimizes the center model* ◁
  $\boldsymbol{w}^{t+1} = \frac{\sum_{j \in \mathcal{I}_t} \boldsymbol{w}_j^{t+1}}{|\mathcal{I}_t|}$; $\boldsymbol{\theta}^{t+1} = \frac{\sum_{j \in \mathcal{I}_t} n_j \boldsymbol{\theta}_j^{t+1}}{\sum_{j \in \mathcal{I}_t} n_j}$

Algorithm 4 Running pFedVEM

Input: T rounds, $\boldsymbol{\mu}_{1:J}^0$, $\boldsymbol{\Sigma}_{1:J}^0$, $\boldsymbol{w}^0$, $\rho_{1:J}^0 := 1$, representation model $\boldsymbol{\theta}^0$, sampling ratio r

Output $\boldsymbol{w}^T$, $\boldsymbol{\theta}^T$, $\boldsymbol{\mu}_{1:J}^T$, $\boldsymbol{\Sigma}_{1:J}^T$

for $t = 0$ to $T-1$ **do**
  Server sends $\boldsymbol{w}^t$ to all clients
  ▷ *clients update their personalized variational distributions* ◁
  for $j = 1, \ldots, J$ **do**
    Expectation is taken by sampling five times:
    $(\boldsymbol{\mu}_j^{t+1}, \boldsymbol{\Sigma}_j^{t+1}) \in \arg\min_{(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \mathbb{E}_{q(\boldsymbol{w}_j)}[\log p(\mathcal{D}_j | \boldsymbol{w}_j)] - \mathrm{KL}[q(\boldsymbol{w}_j) \parallel p(\boldsymbol{w}_j | \boldsymbol{w}, \rho_j^2)]$
    Update ρ_j^{t+1} by $\rho_j^2 = (\mathrm{Tr}(\boldsymbol{\Sigma}_j^{t+1}) + \parallel \boldsymbol{\mu}_j^{t+1} - \boldsymbol{w}^t \parallel^2)/D$
  ▷ *clients update the representation model* ◁
  for $j = 1, \ldots, J$ **do**
    $\boldsymbol{\theta}_j^{t+1} \in \arg\min_{\boldsymbol{\theta}_j} f_j(\boldsymbol{\mu}_j^{t+1}; \boldsymbol{\theta}_j; \mathcal{D}_j)$
  Server select a random subset of clients $\mathcal{I}_t$ of size $\lfloor J \times r \rfloor$
  Each client $j \in \mathcal{I}_t$ sends its updated variational parameters $(\boldsymbol{\mu}_j^{t+1}, \boldsymbol{\Sigma}_j^{t+1})$ and ρ_j^{t+1} to the server
  ▷ *server optimizes the center model* ◁
  $\boldsymbol{w}^{t+1} = \frac{\sum_{j \in \mathcal{I}_t} \tau_j^{t+1} \boldsymbol{\mu}_j^{t+1}}{\sum_{j \in \mathcal{I}_t} \tau_j^{t+1}}$; $\boldsymbol{\theta}^{t+1} = \frac{\sum_{j \in \mathcal{I}_t} n_j \boldsymbol{\theta}_j^{t+1}}{\sum_{j \in \mathcal{I}_t} n_j}$

F Network and hyperparameters

Previous works use a LeNet-based network [11]. However, it performs equally bad and cannot reflect the difference between methods in our evaluation framework, which has a more complicated task. In this work, we adopt a CNN network with five convolutional layers followed by a fully connected layer, as illustrated in Figure 3.

As we proposed a new evaluation framework, for a fair comparison we perform a grid search for the hyperparameters of every method. The variables and corresponding ranges are summarized in Table 2. In particular: 1) For methods with two components/loops, i.e. pFedMAP, pFedVEM, pFedPer and pFedHN, the learning rate and epochs are tuned respectively for each component/loop; 2) For pFedHN, epochs for outer loop is fixed to 1 as required. Other hyperparameters, which are not listed in Table 2, are adopted from previous works; 3) To comply with our theoretical analysis, pFedVEM use data size as batch size when training the classifier, and adopts "sum" loss reduction. To adapt the magnitude of gradient, we scale the range of learning rate for the classifier down with a factor of 0.1; 3) For the Local method, we let the clients with more than 1000 data scale the learning

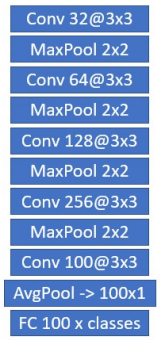


Figure 3: Network architecture, every convolutional layer is followed by a ReLU activation function. In case the method has two components, convolutional layers are used as a representation model while the last fully connected layer is used as a classifier.

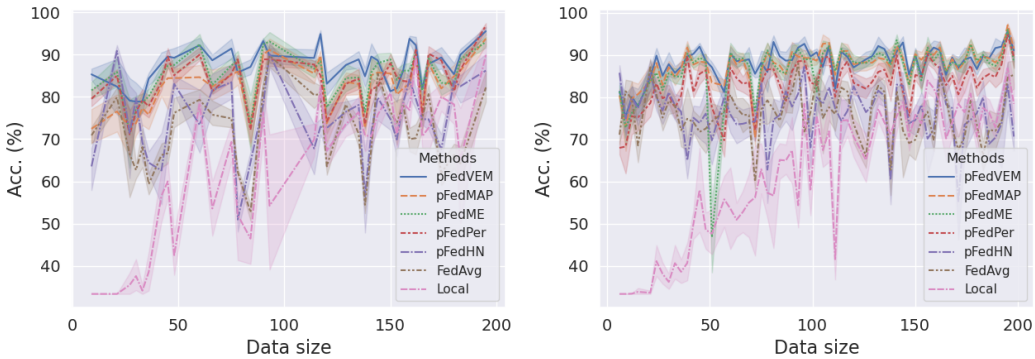
Hyperparameters	Range
Learning rate	$\{0.1,\ 0.01,\ 0.001\}$
Batch size	$\{10,\ 50,\ 100\}$
Epochs	$\{5,\ 10,\ 20\}$
λ	$\{0.1,\ 1,\ 10\}$

Table 2: Grid search for hyperparameters, λ denotes the scalar for the penalty term in loss function.

rate down with a factor of 0.1 on the basis of the learning rate of other clients, otherwise clients with more data may end up with a lower accuracy.

G Additional results

Accuracy over data size An additional plot of accuracy over individual data size on SUN397 is presented in Figure 4



(a) 50 clients (b) 200 clients

Figure 4: Test accuracy (%) over individual data size on SUN397.

Posterior distribution estimation In pFedVEM, every client will approximate a posterior distribution of the linear classifier based on their local data. This estimated distribution can automatically adjust the trade-off between the dependency on the center model and the confidence of the local data. Intuitively, clients having more data will have a higher confidence on their local data, therefore their

estimation of the parameter distribution should have a low variance, also their parameters would have a comparative large distance to the center model due to the low dependency on the center model. Figure 5 verifies this tendency, which indicates that pFedVEM works properly as designed.

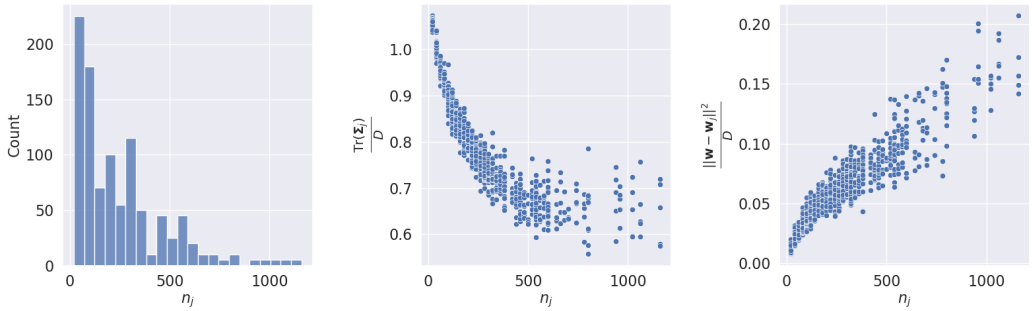


Figure 5: Clients' estimation of the posterior parameter distribution in pFedVEM. The result is a summary of five runs over 200 clients setting on CIFAR100.