Chapter 4
Personalization in Federated Learning

Mayank Agarwal, Mikhail Yurochkin, and Yuekai Sun

Abstract Typical federated learning (FL) problem formulation requires learning a single model suitable for all parties while prohibiting parties from sharing their data with the aggregator. However, it may not be possible to learn a common single model that is suitable for all parties. For example, consider a sentence completion problem: "I live in the state of ..." The answer clearly depends on the party, and no single model is appropriate here. To handle such situations, various personalization strategies have been proposed in the recent literature. In particular, the problem appears to have a close connection to meta-learning. We review recent FL personalization techniques categorizing them into eight groups and summarize three strategies and corresponding datasets for benchmarking personalization in federated learning. We provide an overview of the statistical challenges of personalization in federated learning. At a high level, personalization leads to an increase in the model complexity, which in turn increases the hardness of the federated learning task. We study when too much personalization can prevent standard approaches to personalized federated learning from learning the common parts of the parties and present alternative approaches that overcome such issues.

M. Agarwal (✉)
IBM Research, Cambridge, MA, USA
e-mail: mayank.agarwal@ibm.com

M. Yurochkin
MIT-IBM Watson AI Lab, IBM Research, Cambridge, MA, USA
e-mail: mikhail.yurochkin@ibm.com

Y. Sun
Department of Statistics, University of Michigan, Ann Arbor, MI, USA
e-mail: yuekai@umich.edu

H. Ludwig, N. Baracaldo (eds.), *Federated Learning*,
https://doi.org/10.1007/978-3-030-96896-0_4

4.1 Introduction

Centralized federated learning aims to learn a global model from individual parties' data while keeping their local data private and localized to their individual machines. This global model has the advantage of utilizing data from all the parties and thus generalizes better to test data across parties. In practical scenarios, however, datasets on individual parties are often heterogeneous (non-IID), thus rendering one global model performance sub-optimal for some parties. On the other hand, if each party trains a local model on their local data, they train on a data distribution similar to what is expected at test time but might fail to generalize due to the paucity of data available on a local party. Personalized federated learning aims to learn a model that has the generalization capabilities of the global model but can also perform well on the specific data distribution of each party.

To illustrate the need for personalization, consider the case of a language model learned in a federated learning setting [10]: if we use a global model to predict the next word for the prompt: "*I live in the state of* . . .," the global model will predict the same token (name of the state) for every party irrespective of their local data distribution. Thus, while a global model might be able to learn the general semantics of language well, it fails to personalize to the individual party.

In addition to the aforementioned qualitative example, we can also quantitatively demonstrate the need for personalization. We set up our experiment using the MNIST dataset[1] divided among 100 parties. We distribute the data among these parties in a heterogeneous manner using a Dirichlet distribution with different concentration parameters (α) [64]. We train a 2-layer fully connected network in two settings to measure the benefits parties gain from participating in federated learning. In the first setting, we train an individual network for each of the 100 parties for 10 epochs using solely the parties' own data and measure the performance of these individual networks on their respective parties' test data ($\text{Acc}_i^{\text{local}}$). In the second setting, all 100 parties participate in training a global model using Federated Averaging (FedAvg) [39] for 100 communication rounds, and we measure the performance of this global model on each of the party's test data ($\text{Acc}_i^{\text{global}}$). Figure 4.1 shows histograms of the differences in the performance of the global model and the local model ($\text{Acc}_i^{\text{global}} - \text{Acc}_i^{\text{local}}$) for each of the parties, under different levels of data heterogeneity. As is evident in the plots, the global model does not benefit each party participating in its training, and this phenomenon is more pronounced when the non-IID characteristics of the data are more severe (smaller α values). This experiment emphasizes the need for personalization of the global model on the parties' local data distribution to ensure that every party benefits from its participation in the learning setup.

In this chapter, we review different personalization techniques proposed in federated learning literature and discuss the connections between federated learning and

[1] http://yann.lecun.com/exdb/mnist/

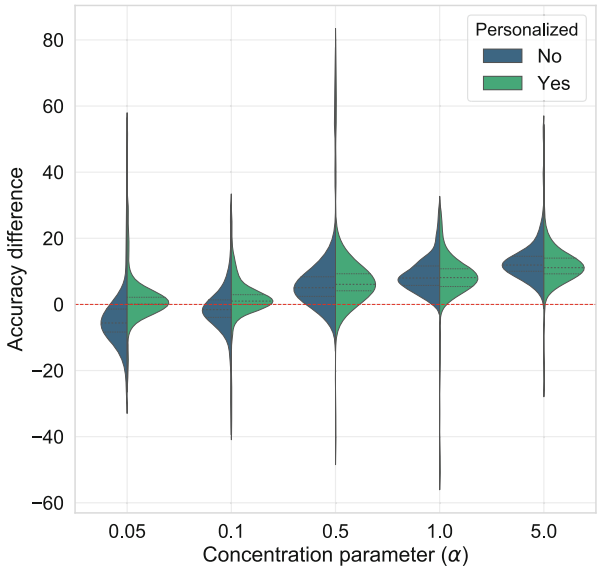

Fig. 4.1 Difference in the accuracies of a global model learned using Federated Averaging and the local models trained solely on the parties' local datasets. Without personalization, under more severe cases of heterogeneity (smaller α), the global model underperforms on a significant number of parties as compared to their local models. With a naive method of personalization by fine-tuning, this effect is attenuated, with the performance improving even in the extreme cases of heterogeneity

first-order meta-learning [18]. We also study the statistical limits of personalization in federated learning. In particular, we show that personalization improves party-specific performance up to a point. After this point, adding more parties to the problem does not lead to improvements in performance.

4.2 First Steps Toward Personalization

In this section, we look at a basic technique that combines federated learning and personalization and explore why this technique is a strong baseline for the personalization task.

4.2.1 Fine-Tuning Global Model for Personalization

A straightforward method to personalize a global model learned using federated learning is to train it further on the local data. This method allows us to control

the level of personalization through the number of local updates performed on the global model—zero local updates retain the global model, while as the number of local updates increases, the model becomes more personalized to the local data.

While this technique might look simple, it is a strong baseline for the personalization task. We study this fine-tuning approach in the experiment presented in Sect. 4.1 and Fig. 4.1. We personalize the global model learned over 100 parties with data distributed in a heterogeneous manner by fine-tuning it for 1 epoch on the local data and then measure the performance of this fine-tuned model on the parties' local test data. As is evident through the results of this experiment, this simple fine-tuning technique considerably improves the performance of the global model as compared to the local models. For the extreme cases of heterogeneity, this method improves the performance for a significant number of parties and also does not negatively impact the performance in less severe cases of heterogeneity. We now aim to understand the reason behind the strong performance of this fine-tuning approach.

4.2.2 Federated Averaging as a First-Order Meta-learning Method

In this section, we try to understand the reason behind the effectiveness of fine-tuning the global model learned using federated averaging. We replicate the derivations of Jiang et al. [29] to show that the updates in Federated Averaging are a combination of federated SGD updates and the first-order MAML (FOMAML) updates.

What is meta-learning and MAML?
While conventional machine learning approaches aim to learn parameters that perform best on a given task, meta-learning or learning to learn [55–57, 59] aims to learn parameters that can be quickly adapted to new tasks. Model-Agnostic Meta-Learning (MAML) [18] is among the most popular meta-learning approaches: its goal is to find model parameters that can be adapted to a new task in few gradient updates. However, to achieve this, the MAML objective requires computing the second-order derivatives, which are computationally expensive. First-order MAML (FOMAML) [43] approximates the MAML objective by considering only the first-order derivatives, thereby reducing the computation demand of MAML. See Sect. 4.3.5 for further discussion of meta-learning and related federated learning personalization strategies.

We start the analysis by defining the update of FedSGD (equation (4.1)). FedSGD operates by taking a single gradient step on each of the N parties, communicating these gradients back to the aggregator, and then aggregating these gradients to update the global model. We use ∇_k^i to denote the kth-step gradient on party i.

$$\nabla_{\text{FedSGD}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}_i(\theta)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\nabla_1^i, \tag{4.1}$$

where θ are the model parameters (e.g., neural network weights) and $\mathcal{L}_i(\theta)$ is the loss of party i. Next, we derive the update of MAML and first-order MAML [18] in similar terms. Assume θ_K^i is the personalized model of party i obtained after taking K steps of the gradient of loss with β as the learning rate:

$$\theta_K^i = \theta - \beta\sum_{j=1}^{K}\frac{\partial \mathcal{L}_i(\theta_j^i)}{\partial \theta}. \tag{4.2}$$

The MAML update is then defined as the gradient of the personalized model θ_K^i with respect to the initial parameters θ, averaged across N parties. Unfortunately, this computation requires higher-order derivatives and is expensive even for $K = 1$. FOMAML ignores the higher-order derivatives and only uses the first-order gradients:

$$\nabla_{\text{FOMAML}}(K) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}_i(\theta_K^i)}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\nabla_{K+1}^i. \tag{4.3}$$

Having computed the updates for FedSGD and FOMAML, we now look at the update of Federated Averaging (FedAvg). The update for FedAvg is the average of party updates, which are the sums of local gradient updates ∇_j^i:

$$\nabla_{\text{FedAvg}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\nabla_j^i = \frac{1}{N}\sum_{i=1}^{N}(\nabla_1^i + \sum_{j=1}^{K-1}\nabla_{j+1}^i) \tag{4.4}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\nabla_1^i + \sum_{j=1}^{K-1}\frac{1}{N}\sum_{i=1}^{N}\nabla_{j+1}^i. \tag{4.5}$$

Rearranging these terms allows us to derive a relation between the updates of FedAvg, FedSGD, and FOMAML:

$$\nabla_{\text{FedAvg}} = \nabla_{\text{FedSGD}} + \sum_{j=1}^{K-1}\nabla_{\text{FOMAML}}(j). \tag{4.6}$$

The FedAvg update for 1-gradient update ($K = 1$) in FedAvg before every communication reduces to the FedSGD setting according to equation (4.6). Increasing the number of gradient updates progressively increases the FOMAML part in the update. According to Jiang et al. [29], models trained with $K = 1$ are hard to personalize, while increasing K increases the personalization capabilities of the models up to a certain point, beyond which the performance of the initial model becomes unstable.

4.3 Personalization Strategies

Personalization in a federated learning setting has gained considerable interest in the research community in recent years. In this section, we look at the various techniques proposed for this problem and classify them into 8 main categories. The classification criteria along with the methods that fall under the respective criteria are summarized in Table 4.1. In the following subsections, we delve deeper into each criterion defined in the table and look at its strengths and weaknesses.

4.3.1 Client (Party) Clustering

The central premise of personalization in federated learning is that one global model might not work for all parties due to the non-IID heterogeneous distribution of data on the parties. Client (party) clustering techniques for personalization operate under a common assumption: among the N parties present in the system, there are $K < N$ distinct data distributions. This assumption enables the techniques to cluster parties into K clusters to alleviate the non-IID data distribution conditions and learns a common global model for each of the K clusters. Thus, under this formulation, the personalization problem is then sub-divided into two sub-problems: (1) Defining a clustering hypothesis to cluster parties together and (2) Aggregating and learning a model for each defined cluster.

Clustered federated learning (CFL) [47] assumes that there exists a partitioning $C = \{c_1, \ldots, c_K\}, \bigcup_{k=1}^{K} c_k = \{1, \ldots, N\}$, such that every subset of parties $c_k \in C$ satisfies the conventional federated learning assumption of a global model minimizing the risk on all the parties' data distributions at the same time. However, instead of identifying the complete clustering C of parties at once, CFL recursively bi-partitions parties into clusters until all the clusters are identified. The algorithm proceeds by training local models until convergence up to a certain limit. These individual party models are then aggregated, and the global model is checked for its congruence, i.e., how well does the global model minimize the risk for each party. If the global model fits a certain stopping criterion for the parties, CFL is terminated. Otherwise, parties are partitioned into two sub-clusters, and CFL is recursively executed on each. Since the bi-partitioning approach works recursively,

Table 4.1 Classification of different methods of personalization in federated learning setting. For each classification, we briefly describe the core idea of this classification along with the methods that fall under this classification criterion

Personalization strategy	Description and methods
Client clustering[a]	Cluster similar parties together to learn models for similar data distributions **Methods:** CFL [47], 3S-Clustering[b] [19], IFCA [20], HypCluster [36]
Client contextualization[a]	Learn contextual features private to the parties to add contextual information to the models along with input features **Methods:** FURL [5], FCF [2]
Data augmentation	Augment local data with data from other parties or global data to increase its diversity and size **Methods:** DAPPER [36], XorMixup [51], global-data-sharing[b] [66]
Distillation	Distill information between local and global models **Methods:** FML [50], FedMD [34]
Meta-learning approach	Formulate the personalization problem as a meta-learning [25, 57] problem **Methods:** FedMeta [9], Per-FedAvg [17], ARUBA [31], FedPer [3]
Mixture of models	Maintain a local model along with a global model and use a combination of the two **Methods:** APFL [15], LG-FedAvg [35], FL+DE [44], MAPPER [36]
Model regularization	Optimize a regularized version of the loss function to balance local model with global model **Methods:** L2GD [23], FedAMP [26], pFedMe [16], Fed+ [61]
Multi-task learning	Use multi-task learning framework [46, 65] for federated learning setting **Methods:** MOCHA [53], VIRTUAL [14]

[a]We use the terms "Client" and "Party" interchangeably here. While the term "Client" is used in some research papers, its analogous term "Party" is the one used in this book
[b]The authors of these methods have not assigned a specific name to their proposed algorithm/technique. We choose to call them so for brevity purposes

the number of clusters K is not required to be known a priori. Additionally, since the clustering mechanism is implemented on the aggregator, parties do not bear the computational burden of the approach. Instead, the aggregator, with its generally greater compute power than parties, can reduce the overhead of clustering.

3S-Clustering [19] also formulates the problem similar to CFL [47], but instead of recursive bi-partitioning of parties, it aims to find the K clusters at once on the aggregator. Once the local models are trained and communicated to the aggregator,

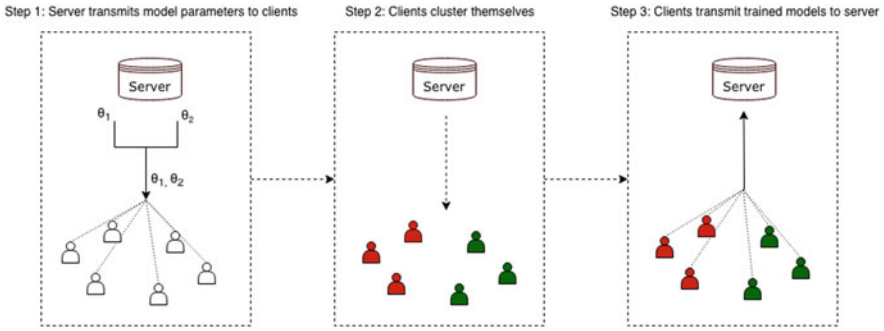

Fig. 4.2 Client (party) clustering strategy for personalization. In this particular instance, the aggregator maintains K separate model parameters and sends these to each individual party, where the parties decide which of the K model parameters they should use

3S-Clustering executes a clustering method—typically, KMeans works, but other clustering methods can also be employed as shown in the original paper where they study this method primarily for byzantine-robust distributed optimization—on the aggregator to find the K clusters. This method, however, is restricted to convex objectives and, hence, does not apply to non-convex objectives such as in the case of deep neural networks.

The aforementioned two methods use the aggregator for party clustering, while the other two methods in this category, IFCA [20] and HypCluster [36], utilize the parties to identify their own cluster memberships (Fig. 4.2). These two methods are pretty similar to each other and operate by the aggregator maintaining K cluster centers and the associated model parameters. At each round, the aggregator broadcasts the cluster parameters to each of the parties, which in turn estimates its cluster identity by choosing the parameters that achieve the lowest loss value. These cluster centers are then used as initializers of the local model, fine-tuned on the local data, and sent back to the aggregator along with the cluster identity for aggregation. The aggregator then aggregates models according to their cluster membership, and the entire process repeats.

4.3.2 Client Contextualization

Learning user-specific contextual features or embeddings has been widely used to improve the personalization of models in problems unrelated to federated learning [1, 22, 27, 38, 58]. Client contextualization utilizes the same approach of learning user embeddings to the task of personalization in federated learning. The rationale behind this approach is that the embeddings for each party capture characteristics specific to the particular party and act as indicators to the global model to utilize this context to adapt its predictions to the specific party.

Federated collaborative filtering (FCF) [2] proposes a collaborative filtering [48]-based recommender system learned in a federated manner. Collaborative filtering models the interaction between N users and M items through a user–item interaction matrix $\boldsymbol{R} \in \mathbb{R}^{N \times M}$ as a linear combination of the user-factor matrix $\boldsymbol{X} \in \mathbb{R}^{K \times N}$ and the item-factor matrix $\boldsymbol{Y} \in \mathbb{R}^{K \times M}$ as

$$\boldsymbol{R} = \boldsymbol{X}^T \boldsymbol{Y}. \tag{4.7}$$

In each iteration of the FCF algorithm, the aggregator sends the item-factor matrix $\boldsymbol{Y}$ to each of the parties, which in turn use their local data to update the user-factor matrix $\boldsymbol{X}$ and the item-factor matrix $\boldsymbol{Y}$. The updated item-factor matrices are sent back to the aggregator for aggregation, while the user-factor matrices are kept private on the individual parties. This allows each party to learn its own set of user-factor matrices while utilizing the item-factor information from across parties. In experiments comparing FCF with standard collaborative filtering, FCF is shown to closely match the performance of standard collaborative filtering on multiple recommendation performance metrics and for multiple recommendation datasets.

While FCF is specific to collaborative filtering, FURL [5] generalizes this approach by (a) defining private and federated parameters and (b) specifying the independent local training constraint and the independent aggregation constraint. The independent local training constraint specifies that the loss function used by local parties is independent of the private parameters of the other parties, while the independent aggregation constraint specifies that the global aggregation step is independent of the private parameters of the parties. When these conditions are met, FURL guarantees no model quality loss from splitting the parameters into private and federated.

An application of FURL to the task of document classification is shown in Fig. 4.3a. Here, the user embeddings are private to each party, while the parameters of the BiLSTM and the MLP are federated parameters that are shared across all users. Each party trains the private and federated parameters jointly but only shares the federated parameters with the aggregator for aggregation. In experiments, personalization through FURL is shown to significantly improve the performance on the document classification task.

Personalization through FURL, however, has several drawbacks. First, using FURL requires incorporating private parameters into the modeling that might require making changes to the network architecture. Subsequently, incorporating the private parameters into the model increases the number of parameters to be learned on the party, which, given the paucity of data on parties, might make the task more difficult. Finally, the FURL technique suffers from a cold-start problem for new parties. Since the private parameters are specific to each party, new parties joining the framework need to first train their private parameters before they can utilize the power of personalization and might suffer from a degraded performance before that.

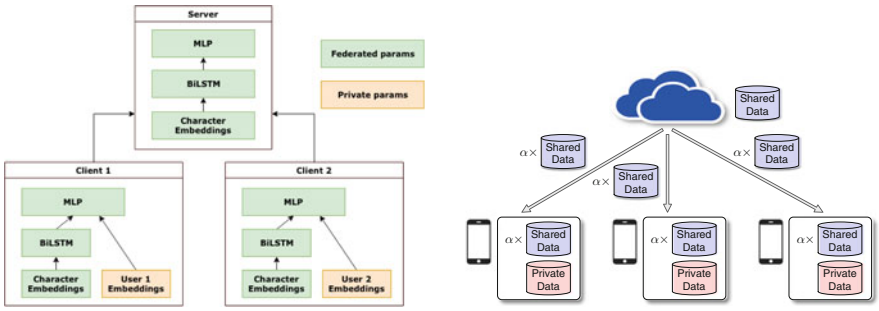

Fig. 4.3 Client contextualization and data augmentation strategies for personalization in federated learning. (**a**) FURL document classification model. Federated parameters (character embeddings) are used along with private parameters (user embeddings) on each party. (**b**) Illustration of data-sharing strategy. Each party uses its private data along with a subset of the global shared data to train its local model. (Image source: original paper [66])

4.3.3 Data Augmentation

Data augmentation techniques have been utilized in standard machine learning problems either to alleviate the problems of class imbalance, non-IID datasets, or to artificially inflate the otherwise lower-sized datasets. Techniques for these range from oversampling under-represented class samples [8] to training GANs to generate augmenting data samples [37]. Readers interested in this area should refer to surveys on data augmentation techniques to gain an understanding of the landscape of the field [40, 52].

Since federated learning also suffers from a paucity of data on the parties, while a significantly large amount of data is available globally, it is natural to ask if the global data (data across all parties) can be used to improve the performance of a particular party. In the same spirit, methods have been proposed either to share a small amount of data globally to help improve performance on parties [66] or to train a Generative Adversarial Network (GAN) in addition to the local model to augment data samples [28].

One straightforward way to augment data is to collect a subset of data from all parties to create a global shared dataset that each party can use to augment their local datasets. DAPPER [36] and global-data-sharing [66] methods fall under this category. Both these methods utilize a global dataset D_G that is indicative of the global data distribution. The global-data-sharing method proposes to initialize the federated learning process by sharing a warm-up model trained on the global dataset, along with a random subset of the dataset (αD_G) to each party. Each party augments its local dataset with the dataset provided by the aggregator to train its local model, which is then transmitted back to the aggregator for aggregation. An illustration of this process is shown in Fig. 4.3b.

DAPPER [36], on other hand, instead of directly augmenting the local dataset with the global dataset, optimizes the objective:

$$\lambda D_{\text{party}} + (1 - \lambda) D_G. \tag{4.8}$$

Each party, at each optimization step, selects the local dataset D_{party} with probability λ and the global dataset D_G with probability $(1 - \lambda)$ for optimization. The rest of the optimization and aggregation steps remain unchanged.

Both DAPPER and global-data-sharing methods show significant improvements over models trained with no personalization, but they require the transfer of parties' data to the global aggregator and to other parties as well. Moving party's data outside their machines violates the privacy guarantees of federated learning, and thus these methods might not be directly implementable in practice.

XorMixup [51] aims to circumvent the privacy concerns associated with transferring party data while still utilizing the personalization power of data augmentation. It proposes to use an XOR encoding scheme to obfuscate data samples to upload to the aggregator. Each party implementing XorMixup first selects data samples from two different class labels and then creates an encoded data sample using an XOR of the two. Each party uploads such encoded samples to the aggregator, which decodes them using a separate data sample from the specified class label, and uses these decoded samples to train a model. These encoded samples are shown to have a high dissimilarity with the original data samples and also show improvement in the performance of models under non-IID conditions.

4.3.4 Distillation

Under the general formulations of federated learning, when the aggregator sends a model to the party, it uses that model as the starting point to train on the local data. Personalization through distillation takes a different approach to the problem. Rather than using the central model parameters as a starting point, distillation-based approaches use knowledge distillation [21, 24] to transfer knowledge between models without explicitly copying parameters. A key advantage of utilizing distillation instead of copying model parameters is that the model architectures need not be the same across parties and the aggregator for the framework to work. For example, parties can choose model architectures more suited to their data and/or hardware constraints. Federated Mutual Learning (FML) [50] and FedMD [34] are two main methods that follow this approach.

What is knowledge distillation?
Model compression [11] is the task of reducing the model size, thereby reducing the memory required to store the model and increasing the speed of inference, while preserving the information in the original neural network. Knowledge distillation [21, 24] is a type of model compression technique that aims to effectively transfer information or knowledge from a larger network to a smaller network. There are 3 main components in knowledge distillation—teacher network, student network, and knowledge. The teacher network is the bigger model that encodes the knowledge, and this knowledge needs to be transferred into the student network that is typically smaller in size. There are several ways to define the knowledge to distill [21]—it can either be outputs of certain layers of the network (such as response-based and feature-based knowledge) or it can be relationships between the different layers or data samples (such as relation-based knowledge).

FML adopts a two-way distillation between the local model and the global model. Parties implementing FML maintain a local model that is continuously trained on their data without sharing the data with the aggregator. At each communication round, the aggregator sends the global model to each party that is updated by the party through a two-way knowledge distillation between the global and the local model. The corresponding objective functions are as follows:

$$\mathcal{L}_{\text{local}} = \alpha \mathcal{L}_{\text{local}} + (1-\alpha) D_{KL}(p_{\text{global}} \| p_{\text{local}}). \tag{4.9}$$

$$\mathcal{L}_{\text{global}} = \beta \mathcal{L}_{\text{global}} + (1-\beta) D_{KL}(p_{\text{local}} \| p_{\text{global}}). \tag{4.10}$$

Since the connection between the local and global models in FML is through the KL divergence of the output probabilities unlike through parameter copying in other federated learning methods, the local and global model architectures can be different. The original work for FML [50] conducts experiments to demonstrate this effect. Using different network architectures on different parties, the authors show improvement over independent training of a global model on the complete dataset under this setting as well.

FedMD [34] also proposes a similar formulation of personalization using distillation as FML. The FedMD framework requires a public dataset that is shared across parties and the aggregator, along with the private datasets that each party maintains. The framework proceeds by parties first training models on the public dataset followed by training on their respective private dataset and then communicating the class scores for each sample in the public dataset to the central aggregator. An aggregation of these class scores across all parties is used as the target distribution that each party learns from using distillation. Similar to FML, FedMD has the advantage of supporting different model architectures across parties. However,

FedMD requires a large public dataset that needs to be shared between parties, thereby increasing the communication cost between parties and aggregator.

4.3.5 Meta-learning Approach

Contemporary machine learning models are trained to perform well on a single task. Meta-learning or "learning to learn" [25, 57, 59], on the other hand, aims to learn models that can be rapidly adapted to new tasks with only a handful of examples. There are multiple ways of achieving this—metric-based, model-based, and optimization-based methods [59]. In this section, we focus on optimization-based methods that are better suited for our purposes. Optimization-based techniques for meta-learning aim to learn model parameters that can be quickly modified to new tasks given a handful of examples and within a few gradient updates. Model-Agnostic Meta-Learning (MAML) [18] is a fairly popular method that is applicable to any model that can be learned with gradient-based methods. Instead of training model parameters to minimize the loss on a given task, MAML trains model parameters to minimize the loss on tasks after a few parameter adaptation steps. If we consider each task to be a party in federated learning setting, we can draw a parallel between personalized federated learning and meta-learning. We want to train a global model to act as a good initializer for party models such that it is able to adapt, i.e., personalize, quickly to party data distributions. In Sect. 4.2, we reviewed the connections between the naive personalization baseline, i.e., fine-tuning of FedAvg, and meta-learning. We now review other recent methods based on meta-learning.

ARUBA [31] is a framework that combines meta-learning with multi-task learning techniques to enable meta-learning methods to learn and take advantage of task similarities to improve their performance. One of the motivations behind ARUBA is that in meta-learning models, certain model weights act as feature extractors and are transferable across tasks without much modification, while other weights vary greatly. Having a per-coordinate learning rate allows parameters to be adapted at different rates depending on their transferability across tasks. ARUBA, when tested on the next-character prediction task in a federated learning setting, matches the performance of a fine-tuned FedAvg baseline, but without additional hyperparameter optimization over the fine-tuning learning rates.

FedMeta [9]—proposed concurrently with ARUBA—incorporates standard meta-learning algorithms into federated learning setting. Under this setting, the aggregator aims to maintain an initialization that a party can quickly adapt to its local data distribution. The party trains by executing the inner loop (adaptation steps on the support data) of the meta-learning algorithm locally and returns the gradient of the outer loop (query data) back to the aggregator, which uses this information to update its initialization. While FedMeta incorporates meta-learning into personalization by running the meta-learning step on the parties while aggregating the model initialization on the aggregator, Per-FedAvg [17] shows

that this formulation may not perform well in some cases. Instead, Per-FedAvg assumes that each party takes the global model as initialization and updates it for one gradient step with respect to its own loss function, changing the problem formulation as follows (4.11):

$$\min_{\theta} F(\theta) := \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(\theta - \alpha \nabla \mathcal{L}_i(\theta)). \tag{4.11}$$

Finally, FedPer [3] proposes to separate the global model into a base network that acts as a feature extractor, and a personalization layer. The parties jointly train the base and the personalization layers but only share the base network with the aggregator for aggregation. This allows the system to learn the representation extraction network using information from multiple parties, while learning a personalization layer specific to each party. The connection between this method and meta-learning is not explicitly explored in the original paper, and there is related work in the meta-learning literature, Almost No Inner Loop (ANIL) [45], which proposes separating the network into a body and head network and only adapting the head to a new task in the inner loop of meta-learning.

4.3.6 Mixture of Models

In the standard formulation of federated learning, the local model is trained on local data, while the global model aggregates information from parties to build a global model. The key incentive for a party to participate in federated learning is to utilize the information on other parties to reduce its generalization error as compared to training a model locally. However, there can be cases where the global model performs worse for certain parties in the federated learning system than the individual models that these parties could train locally [62], e.g., see experiment in Fig. 4.1. This motivates the idea of mixing the global and local models by learning a parameter to optimally combine the two models.

FL+DE [44] learns to combine the predicted class probabilities of the local and global models using a mixture of experts technique [63]. Each party maintains a local domain expert (DE) model trained on the local data, while also collaborating with other parties to build a global model. The gating function ($\alpha_i(x)$)—parameterized as a logistic regression model in the original work [44]—is learned along with the federated learning setup to optimally combine the predicted class probabilities of the global model ($\hat{y}_G$) and the local domain expert ($\hat{y}_i$). The gating function thus learns regions of preferences between the two models conditioned on the input. The final prediction for a given data sample x is then a convex combination of the predicted class probabilities of the two models:

$$\hat{y}_i = \alpha_i(x)\hat{y}_G(x) + (1 - \alpha_i(x))\hat{y}_{\text{local}}(x). \tag{4.12}$$

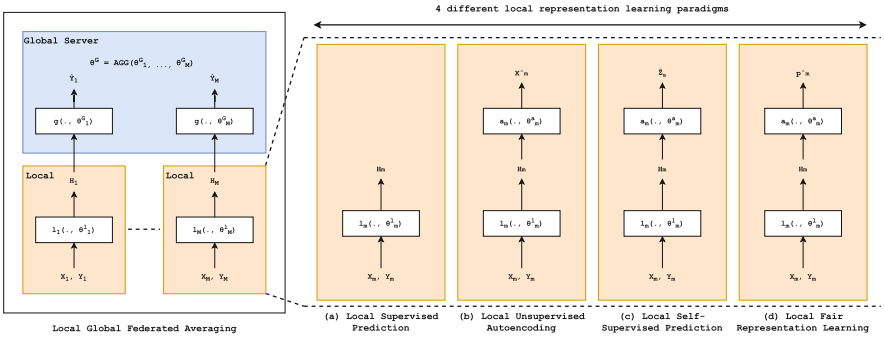

Fig. 4.4 LG-FedAvg—the local models $\ell_i(\,\cdot\,;\theta_i^{\ell})$ learn to extract high-level representations H_i given the local data (X_i, Y_i), while the global model $g(\,\cdot\,;\theta^{g})$ operates solely on the learned representations H_i. Owing to this bifurcation, the local model can be trained using specialized techniques, 4 of which are shown in this figure

$$\alpha_i(x) = \sigma(w_i^T x + b_i). \tag{4.13}$$

Instead of using a mixture of experts technique to combine the output probabilities, the MAPPER [36] and APFL [15] methods learn a mixing parameter (α) to optimally combine the local and global models. In APFL, while the global model is still trained to minimize the empirical risk on the aggregated domain as in traditional federated learning, the local model (h_{local}) is trained to also incorporate part of the global model (h_g) using α (equation (4.14)). The personalized model for the i-th party is a convex combination of the global model (h_g) and the local model (h_{local}) (equation (4.15)).

$$h_{\text{local}} = \arg\min_{h} \hat{\mathcal{L}}_{D_i}(\alpha_i h + (1-\alpha_i)h_g). \tag{4.14}$$

$$h_{\alpha_i} = \alpha_i h_{\text{local}} + (1-\alpha_i)h_g. \tag{4.15}$$

The three methods we have looked at so far maintain both the local and global models. However, both these models are trained for the same task. LG-FedAvg [35] instead proposes to bifurcate the learning task between the local and global models—each party learns to extract high-level representations from raw data, and the global model operates on these representations (rather than the raw data) from all devices. We depict the LG-FedAvg process for an image classification task in Fig. 4.4. Here, the local models are trained to extract high-level representations from raw images, and while the global model is trained using supervised learning, the local models are free to choose the technique to learn these representations. As shown in Fig. 4.4, these can be learned using supervised prediction task (using an auxiliary model to map representations to predictions) or unsupervised or semi-supervised techniques (sub-figures (a) to (c)). The local models can also be trained to learn fair representation through adversarial training against protected attributes

(sub-figure d). This bifurcation has several benefits: (a) Operating the global model on representations rather than raw data reduces the size of the global model, thus reducing the number of parameters and updates that need to be communicated between the aggregator and the parties, (b) it allows the local parties to choose a specialized encoder to extract representations depending on the characteristics of its local dataset rather than using a common global model, and (c) it allows local models to learn fair representations that obfuscate protected attributes thus enhancing the privacy of the local data.

4.3.7 Model Regularization

In conventional supervised federated learning, the system is optimized for the following:

$$\min_{\theta} \left\{ \mathcal{L}(\theta) := \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(\theta) \right\}. \tag{4.16}$$

Here, N is the number of parties, θ are the model parameters, and $\mathcal{L}_i(\theta)$ denotes the loss over the ith party data distribution.

Regularization-based personalization techniques, on the other hand, optimize for a regularized version of the standard objective. Loopless Local Gradient Descent (L2GD) [23], Federated Attentive Message Passing (FedAMP) [26], pFedME [16], and Fed+ [61] are all instantiations of the regularization technique, differing primarily in their definition of the regularization objective.

L2GD [23] defines the regularization objective to be the L2 norm of the difference between the local model parameters (θ_i) and the average parameters across parties ($\bar{\theta}$), and the entire system optimizes the objective defined in equations (4.17) and (4.18). To optimize for this objective, L2GD proposes a non-uniform SGD method with convergence analysis over the required number of communication rounds between the parties and the aggregator. The method views the objective as a 2-sum problem, sampling either $\nabla\mathcal{L}$ or $\nabla\psi$ to estimate ∇F, and defines an unbiased estimator of the gradient as in equation (4.19). At each time step, either the local models take a local gradient step with probability $1 - p$ or the aggregator shifts the local models toward the average with probability p.

$$\texttt{L2GD}: \min_{\theta_1,\ldots,\theta_N} \left\{ F(w) := \mathcal{L}(\theta) + \lambda\psi(\theta) \right\}. \tag{4.17}$$

$$\mathcal{L}(\theta) := \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(\theta_i), \quad \psi(\theta) := \frac{1}{2N} \sum_{i=1}^{N} \|\theta_i - \bar{\theta}\|^2. \tag{4.18}$$

$$G(\theta) := \begin{cases} \frac{\nabla\mathcal{L}(\theta)}{1-p} & \text{with probability } 1-p, \\ \frac{\lambda\nabla\psi(\theta)}{p} & \text{with probability } p. \end{cases} \tag{4.19}$$

L2GD is applicable to and provides guarantees for convex loss functions and thus is not directly applicable to non-convex loss functions typically encountered in neural network models.

pFedMe [16] models the personalization problem as a bi-level optimization problem through the objective defined as follows:

$$\texttt{pFedMe} : \min_{\theta} \left\{ F(\theta) = \frac{1}{N} \sum_{i=1}^{N} F_i(\theta) \right\}. \tag{4.20}$$

$$F_i(\theta) = \min_{\theta_i} \left\{ \mathcal{L}_i(\theta_i) + \frac{\lambda}{2} \|\theta_i - \theta\|^2 \right\}. \tag{4.21}$$

Here, θ_i is the ith party's personalized model, trained on its local data distribution while maintaining a bounded distance from the global model parameters θ at the inner level. The optimal personalized model for a party is then defined as

$$\hat{\theta}_i(\theta) := \text{prox}_{\mathcal{L}_i/\lambda}(\theta) = \arg\min_{\theta_i} \left\{ \mathcal{L}_i(\theta_i) + \frac{\lambda}{2} \|\theta_i - \theta\|^2 \right\}. \tag{4.22}$$

Similar to FedAvg, a system implementing pFedMe sends the global model weights to the parties at each communication round and performs model aggregation using weights returned by parties after a certain number of local rounds. Unlike FedAvg, party locally minimizes equation (4.21), which is a bi-level optimization problem. At each local round, the party first solves equation (4.22) to find the optimal personalized party parameters $\hat{\theta}_i(\theta_{i,r}^t)$. Here, $\theta_{i,r}^t$ is the local model of party i at global round t and local round r, where $\theta_{i,0}^t = \theta^t$. Thereafter, at the outer level, the party updates the local model $\theta_{i,r}^t$ using gradients with respect to F_i in equation (4.21).

FedAMP [26] proposes the following objective for personalization:

$$\min_{\theta} \left\{ F(\theta) = \sum_{i=1}^{N} \mathcal{L}_i(\theta_i) + \lambda \sum_{i<j}^{N} A(\|\theta_i - \theta_j\|^2) \right\}. \tag{4.23}$$

The second part of the objective defines an attention-inducing function $A(\|\theta_i - \theta_j\|^2)$, which measures the similarity between party parameters in a non-linear manner, and aims to improve the collaboration between parties. The attention-inducing function can take any form; however, in the proposed work, the authors use the negative exponential function $A(\|\theta_i - \theta_j\|^2) = 1 - e^{-\|\theta_i - \theta_j\|^2/\sigma}$. To optimize for the objective, FedAMP adopts an alternate-optimization strategy, first optimizing $\sum_{i<j}^{N} A(\|\theta_i - \theta_j\|^2)$ on the aggregator through weights collected from all parties, and then optimizing $\mathcal{L}_i(\theta_i)$ on the corresponding parties using their local datasets.

Fed+ [61] argues that robust aggregation allows better handling of the heterogeneity of data across parties and accommodates it through a model regularization

approach to personalization. Fed+ introduces a convex penalty function ϕ and constants α, μ as follows:

$$\min_{\theta,z,\bar{\theta}} \frac{1}{N}\left\{F_{\mu,\alpha}(\theta,z,\bar{\theta}) = \sum_{i=1}^{N} \mathcal{L}_i(\theta_i) \;+\; \frac{\alpha}{2}\|\theta_i - z_i\|_2^2 \;+\; \mu\phi(z_i - \bar{\theta})\right\} \tag{4.24}$$

and proposes a robust combination of the current local and global models obtained by minimizing (4.24) with respect to z_i, keeping θ_i and $\bar{\theta}$ fixed. Setting $\phi(\cdot) = \|\cdot\|_2$ gives a ρ-smoothed approximation of the Geometric Median, a form of robust aggregation:

$$\begin{aligned} z_i \;&\leftarrow \bar{\theta} + \text{prox}_{\phi/\rho}(\theta_i - \bar{\theta}), \quad \text{where} \quad \rho := \mu/\alpha, \;\; \text{hence} \\ z_i \;&=\; (1-\lambda_i)\,\theta_i \;+\; \lambda_i\,\bar{\theta}, \quad \text{where} \quad \lambda_i \;:=\; \min\left\{1,\; \rho/\|\theta_i - \bar{\theta}\|_2\right\}. \end{aligned}$$

To compute the robust $\bar{\theta}$ from $\{\theta_i\}$, the aggregator runs the following two-step iterative procedure initialized with $\bar{\theta} = \theta_{\text{mean}} := \texttt{Mean}\{\theta_i\}$ until $\bar{\theta}$ converges:

$$\begin{aligned} v_i \;&\leftarrow\; \max\left\{0,\; 1 - (\rho/\|\theta_i - \bar{\theta}\|_2)\right\}(\theta_i - \bar{\theta}), \\ \bar{\theta} \;&\leftarrow\; \theta_{\text{mean}} - \texttt{Mean}\{v_i\}. \end{aligned}$$

4.3.8 Multi-task Learning

Traditional machine learning approaches typically optimize a model for a single task. Multi-task learning (MTL) [4, 7, 54] extends this traditional approach to learn multiple tasks jointly, thus exploiting the commonalities and differences between the tasks to potentially enhance the performance on individual tasks. Since these methods can learn relationships between non-IID and unbalanced datasets, they are well suited to apply to the federated learning setting [53]. Readers interested in multi-task learning should refer to the following survey papers [46, 65] to gain an overview of the field.

While MTL methods are appealing in the federated learning context, they do not account for the communication challenges such as fault tolerance and stragglers in the framework. MOCHA [53] was the first framework for federated learning using multi-task learning that factored in fault tolerance and stragglers during training. MTL approaches generally formulate the problem as follows:

$$\min_{w,\Omega}\left\{\sum_{i=1}^{N} \mathcal{L}_i(\theta_i) + R(\Omega)\right\}. \tag{4.25}$$

Here, N is the total number of tasks, and $\mathcal{L}_i(\theta_i)$ and θ_i are the loss function and parameters for task i. The matrix $\Omega \in \mathbb{R}^{N \times N}$ models relationships among tasks and is either known a priori or is estimated while simultaneously learning task models. MTL approaches differ in their formulation of R that promotes suitable structure among the tasks through the Ω matrix. MOCHA optimizes this objective in the federated learning setting using the objective's distributed primal–dual optimization formulation. This allows it to separate computation across nodes by only requiring data available on the party to update the local model parameters. MOCHA showed the applicability of multi-task learning approaches to federated learning setting and showed improved performance as compared to global and local models trained on the experimental datasets. It, however, is designed for only convex models and is therefore inapplicable to non-convex deep learning models.

VIRTUAL (variational federated multi-task learning) [14] extends the multi-task learning framework to non-convex models through the usage of variational inference methods. Given N parties, each with datasets D_i, local model parameters θ_i, and the central model parameters θ, VIRTUAL computes the posterior distribution

$$p(\theta, \theta_1, \ldots, \theta_N | D_{1:N}) \propto \frac{\prod_{i=1}^{N} p(\theta, \theta_i | D_i)}{p(\theta)^{N-1}}. \tag{4.26}$$

This posterior distribution assumes: (1) that party data is conditionally independent given aggregator and party parameters, $p(D_{1:N}|\theta, \theta_1, \ldots, \theta_N) = \prod_{i=1}^{N} p(D_i|\theta, \theta_i)$, and (2) a factorization of the prior as $p(\theta, \theta_1, \ldots, \theta_N) = p(\theta) \prod_{i=1}^{N} p(\theta_i)$. Since the posterior distribution defined in equation (4.26) is intractable, the algorithm proposes an expectation propagation like algorithm [41] to approximate the posterior.

4.4 Benchmarks for Personalization Techniques

In this section, we review datasets suitable for benchmarking methods for personalization in federated learning. We consider datasets with non-IID party data distributions, i.e., each party's data is sampled from a different distribution. We review datasets used in the prior works, as well as other datasets that might fit the personalization problem setting.

Broadly classified, prior works in this domain have used one of the following types of datasets: (a) Synthetic datasets, where a generative process for the dataset is defined to generate data samples for parties, (b) Simulating federated datasets by partitioning commonly available datasets such as MNIST or CIFAR-10 [32] according to some hypothesis, and (c) Utilizing datasets that have a natural partitioning such as data collected from multiple parties. We now look at each of these types in detail.

4.4.1 Synthetic Federated Datasets

A fairly common way of generating synthetic federated dataset is to follow the method proposed by Shamir et al. [49], with some added modifications to inject heterogeneity among parties. While the exact way of generating the dataset varies between proposed methods, the underlying process is as follows: for each device k, samples (X_k, Y_k) are generated according to the model $y = \arg\max(\text{softmax}(Wx + b))$. The model parameters W_k and b_k are controlled by a parameter α and are sampled as: $u_k \sim \mathcal{N}(0, \alpha)$, $W_k \sim \mathcal{N}(u_k, 1)$, and $b_k \sim \mathcal{N}(u_k, 1)$. The generation of X_k is controlled by the second parameter β and is sampled as: $B_k \sim \mathcal{N}(0, \beta)$, $v_k \sim \mathcal{N}(B_k, 1)$, Σ is a diagonal covariance matrix with $\Sigma_{j,j} = j^{-1.2}$, and $x_k \sim \mathcal{N}(v_k, \Sigma)$.

This synthetic dataset `Synthetic(α, β)` has two parameters: α and β. Here, α controls how much local models differ from each other, and β controls how much the local data on each device differs from data on other parties.

4.4.2 Simulating Federated Datasets

A common way of simulating federated datasets is to use commonly available datasets and partitioning them across parties according to a hypothesis. Prior works have generally utilized datasets such as MNIST,[2] CIFAR-10, and CIFAR-100 [32] for the task. Since these datasets have no specific natural feature that can be used to partition them across parties, we need to partition them according to a hypothesis. One way to partition is to sample data points for each party from a particular subset of classes, to ensure that parties do not see data from all classes and thus do not have features representative of all the classes in the dataset. Another way of partitioning includes using a probabilistic allocation of data samples across parties—such as sampling $p_k \sim \text{Dir}_N(\alpha)$, and allocating $p_{k,i}$ proportion of instances of class k to party i [64].

Synthetically created federated datasets have the advantage of allowing control over the amount of heterogeneity in the dataset; however, they are limited by the number of parties they can support. Prior works have set up their experiments with the number of parties in the order of tens. While this is suitable for federated learning in an enterprise setting where typically the number of parties does not scale too much, this setting does not consider the scale typically encountered in smartphones or IoT type of applications.

[2] http://yann.lecun.com/exdb/mnist/

Table 4.2 Federated learning datasets

Dataset	Task	No. of parties	Total samples	Samples per device	
				Mean	Std
FEMNIST	Image classification	3,550	805,263	226.83	88.94
CelebA	Image classification	9,343	200,288	21.44	7.63
Shakespeare	Language modeling	1,129	4,226,158	3,743.28	6, 212.26
Reddit	Language modeling	1,660,820	56,587,343	34.07	62.95
Sent140	Sentiment analysis	660,120	1,600,498	2.42	4.71

4.4.3 Public Federated Datasets

Besides synthetic and simulated federated datasets, there are datasets available, which support natural partitioning of data among parties. LEAF [6] is a popular benchmark for federated learning methods that provides multiple datasets for image classification, language modeling, and sentiment analysis tasks. These datasets are the preferred datasets for benchmarking personalization techniques due to their proximity to real-life non-IID characteristics of data, and the scale of parties they support. Some of the datasets included in LEAF are:

- **FEMNIST:** Extended MNIST (EMNIST) [13] is a dataset containing handwritten samples of digits, uppercase, and lowercase characters for a total of 62 class labels. The EMNIST dataset is partitioned by the original writers of the handwritten samples to create the FEMNIST dataset with over 3,500 parties.
- **Shakespeare dataset:** Intended for language modeling task, this dataset is built from *The Complete Works of William Shakespeare*,[3] by considering each speaking character in each play as a separate party.

Details about the datasets available in LEAF along with their aggregate and party-level statistics can be found in Table 4.2.

In addition to the datasets provided in LEAF, there are other datasets that have the characteristics required for personalization in federated learning task. Some of these datasets are:

- **Google Landmarks Dataset v2 (GLDv2):** The GLDv2[4] is a large-scale fine-grained dataset intended for instance recognition and image retrieval tasks [60]. It is composed of approximately 5 million images of human-made and natural landmarks across 246 countries, with approximately 200,000 distinct labels for these images. This dataset can be utilized in the federated learning setting by either partitioning according to the geographical location of the landmarks or the

[3] http://www.gutenberg.org/ebooks/100

[4] https://github.com/cvdfoundation/google-landmark

landmark categories, or the author. Given its scale and diversity, this dataset is a strong test bed for the personalization task.

- **MIMIC-III:** The Medical Information Mart for Intensive Care III (MIMIC-III) [30] is a large-scale de-identified health-related data of over 40,000 patients who stayed at the critical care units of a hospital in Boston, Massachusetts between 2001 and 2012. It includes information such as demographics, vital sign measurements, laboratory test results, procedures, medications, caregiver notes, imaging reports, and mortality (both in and out of hospital), for over 60,000 critical care unit stays. While the scale of this dataset is limited in comparison to the GLDv2, it is one of the largest available medical datasets and thus provides an important benchmark for evaluating personalization in federated learning.

4.5 Personalization as the Incidental Parameters Problem

There is a possible theoretical explanation for the limitations of personalization in federated learning: the incidental parameters problem. We consider a general model of personalization in federated learning: the aggregator and the parties aim to solve an optimization problem of the form

$$\min_{\theta,\theta_1,\ldots,\theta_N} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}_i(\theta,\theta_i), \tag{4.27}$$

where θ are the shared model parameters, the θ_i's are the party-specific parameters, and $\mathcal{L}_i$ is the empirical risk on the i-th party. In most federated learning settings, the sample size per party is limited, so the party-specific parameters θ_i are only estimated up to a certain accuracy that depends on the sample size per party. We may hope that it is possible to estimate the shared parameters θ more accurately, but this is only possible up to a point.

The population version of the problem is

$$\min_{\theta,\theta_1,\ldots,\theta_N} \frac{1}{N}\sum_{i=1}^{N} \mathcal{R}_i(\theta,\theta_i), \tag{4.28}$$

where $\mathcal{R}_i$ is the (population) risk on the i-th party: $\mathcal{R}_i(\cdot) \triangleq E\big[\mathcal{L}_i(\cdot)\big]$. Let $(\widehat{\theta},\widehat{\theta}_1,\ldots,\widehat{\theta}_N)$ and $(\theta^*,\theta_1^*,\ldots,\theta_N^*)$ be the argmin of (4.27) and (4.28), respectively. The estimates of the shared parameters satisfy the *score equations*:

$$0 = \frac{1}{N}\sum_{i=1}^{N} \partial_\theta \mathcal{L}_i(\widehat{\theta},\widehat{\theta}_1,\ldots,\widehat{\theta}_N). \tag{4.29}$$

Expanding the score equations around $(\theta^*, \theta_1^*, \ldots, \theta_N^*)$ (and dropping the higher-order terms), we have

$$0 = \frac{1}{N}\sum_{i=1}^{N} \partial_\theta \mathcal{L}_i(\theta^*, \theta_1^*, \ldots, \theta_N^*) + \partial_\theta^2 \mathcal{L}_i(\theta^*, \theta_1^*, \ldots, \theta_N^*)(\widehat{\theta} - \theta^*) + \partial_{\theta_i}\partial_\theta \mathcal{L}_i(\theta^*, \theta_1^*, \ldots, \theta_N^*)(\widehat{\theta_i} - \theta_i^*).$$

We rearrange to isolate the estimation error in the shared parameters

$$\widehat{\theta} - \theta^* = (\frac{1}{N}\sum_{i=1}^{n} \partial_\theta^2 \mathcal{L}_i(\theta^*, \theta_1^*, \ldots, \theta_N^*))^{-1} \left(\begin{array}{l} \frac{1}{N}\sum_{i=1}^{N} \partial_\theta \mathcal{L}_i(\theta^*, \theta_1^*, \ldots, \theta_N^*) \\ \quad + \partial_{\theta_i}\partial_\theta \mathcal{L}_i(\theta^*, \theta_1^*, \ldots, \theta_N^*)(\widehat{\theta_i} - \theta_i^*) \end{array} \right).$$

We see that the estimation error in the party-specific parameters affects the estimation error of the shared parameters through the average of the terms $\partial_{\theta_i}\partial_\theta \mathcal{L}_i(\theta^*, \theta_1^*, \ldots, \theta_N^*)(\widehat{\theta_i} - \theta_i^*)$. This average is generally not mean zero, so it does not converge to zero even as the number of parties grows. For this average to converge to zero, one of two things must occur:

1. $\widehat{\theta_i} - \theta_i^* \xrightarrow{p} 0$: the estimation errors of the personalized parameters converge to zero. This is only possible if the sample size per party grows. Unfortunately, computational and storage constraints on the parties preclude this scenario in most federated learning problems.
2. $\partial_{\theta_i}\partial_\theta \mathcal{L}_i(\theta^*, \theta_1^*, \ldots, \theta_N^*)$ is the mean zero. This is equivalent to the score equations (4.29) satisfying a certain orthogonality property [12, 42]. If the score equations satisfy this property, then the estimation errors of the party-specific parameters do not affect (to first-order) the estimates of the shared parameters. Although this is highly desirable, orthogonality only occurs in certain special cases.

In summary, the estimation error in the shared parameters is generally affected by the estimation errors in the party-specific parameters, and it does not converge to zero in realistic federated learning settings in which the number of parties grows but the samples size per party remains bounded. Taking a step back, this is to be expected from a degree-of-freedom point of view. As we grow the number of parties, although the total sample size increases, the total number of parameters we must learn also increases. In other words, personalization in federated learning is a high-dimensional problem. Such problems are known to be challenging, and estimation errors generally do not converge to zero unless the parameters exhibit special structure (e.g., sparsity, low rank). Unfortunately, this is typically not the case in federated learning.

Practically, this means that if we incorporate personalization in a federated learning problem, it is profligate to increase the number of parties beyond a certain

point without increasing the sample size per party. Whether we are beyond this point can be ascertained by checking whether the quality of the shared parameters estimates is improving as more parties are added. If we are beyond this point, the estimation error in the party-specific parameters is dominating the estimation error in the shared parameters, so there is no benefit to parameter sharing. This is known as the *incidental parameters* problem, and it has a long history in statistics. We refer to [33] for a review of the problem.

4.6 Conclusion

In this chapter, we motivate the need for personalization by demonstrating that native federated learning does not necessarily help all parties train a better model as compared to them training the models locally. To alleviate this problem, different techniques for personalization have been proposed. We group these techniques into eight major categories based on the type of personalization strategy they employ. In addition to a review of personalization strategies, we also provide an overview of the statistical challenges of personalization in federated learning. We conclude this chapter by providing recommendations for practical considerations while implementing or utilizing personalization strategies, and future research directions, and open problems in theoretical understanding of personalization in federated learning.

Practical considerations. Choosing a personalization strategy for your application is closely tied to the properties of the parties and the aggregator participating in the federated learning setup. Specifically, the questions that will help in making an informed choice are: (1) Do all parties have the same model architecture? (2) Is there a data-sharing mechanism available to augment local data? (3) What are the compute capabilities available on the participating parties and the aggregator? and (4) How much data do you expect to be present on each party?

If not all parties have the same model architecture, or if it is preferable to have different architectures for different parties, then either distillation-based approaches (Sect. 4.3.4) or the LG-FedAvg [35] method should be explored. These techniques support and have been experimentally shown to work with different party and global model architectures. Another important consideration is if there is global data available to augment the local data. While sharing party data violates the core tenet of federated learning, if it is possible to collect a shared dataset, data augmentation techniques for personalization (Sect. 4.3.3) can be powerful candidates in these scenarios.

The party and aggregator compute capabilities also play an important role in selecting a personalization strategy. Specifically for parties, if the compute and memory capabilities are sufficiently available, then a mixture of models approach (Sect. 4.3.6) can be explored. Because the mixture of models approach maintains a local and a global model on the parties and uses a combination of the two for

inference, it significantly increases the memory and compute requirements for the participating parties. On similar lines, if the aggregator has enough memory capacity to allow for maintaining multiple model parameters, then client (party) clustering approaches (Sect. 4.3.1) might be helpful.

The final question that will help in making an informed decision is regarding the amount of data available on each party. This is an important consideration to apply contextualization approaches. Client (party) contextualization (Sect. 4.3.2) increases the number of parameters to learn from the local data, and if sufficient data is available on parties, then it might be possible to learn these contextual parameters to help in personalization.

Lastly, meta-learning (Sect. 4.3.5) and model regularization (Sect. 4.3.7) approaches are applicable irrespective of whether the aforementioned conditions are met or not and should always be considered when choosing a personalization strategy.

Advancing personalization in federated learning. With the growing number of personalization algorithms proposed in the literature, an important next step in our opinion is to establish benchmarks and performance metrics to effectively and reliably measure the performance of the proposed techniques. This requires the availability of datasets that mimic the conditions typically encountered in practical deployments. While some datasets exist for this purpose, there is a need for more datasets in a broader range of application domains. Benchmarking on standardized datasets will allow for better interpretation of the capabilities and limitations of the proposed techniques and will also enable easy comparison across the techniques. In addition to datasets, there is a need for a standardized evaluation setup for personalization. The typical way of evaluating federated learning techniques is to measure the performance of the global model, and this technique has been ported over to the personalization problem as well. However, as we saw in the motivating example for personalization, measuring the global model accuracy does not necessarily provide a complete picture of the performance of each party. Thus, defining an evaluation setup that considers the performance of each party will serve as an important contribution for effectively evaluating personalization techniques.

Theoretical understanding of personalization. As we saw, there is a scalability issue with personalization due to the incidental parameters problem. This issue is distinguished from most scalability issues in machine learning by its statistical nature. Overcoming this issue is a requirement for scaling personalization to large party clouds. Unfortunately, a general solution to the underlying incidental parameters problems has eluded the statistics community for the better part of a century, so it is unlikely that there is a general way to perform personalization at scale. However, it may be possible to develop solutions tailored to the particular model/application, and this is a rich area of future work.

References

1. Amir S, Wallace BC, Lyu H, Silva PCMJ (2016) Modelling context with user embeddings for sarcasm detection in social media. arXiv preprint arXiv:160700976
2. Ammad-Ud-Din M, Ivannikova E, Khan SA, Oyomno W, Fu Q, Tan KE, Flanagan A (2019) Federated collaborative filtering for privacy-preserving personalized recommendation system. arXiv preprint arXiv:190109888
3. Arivazhagan MG, Aggarwal V, Singh AK, Choudhary S (2019) Federated learning with personalization layers. arXiv preprint arXiv:191200818
4. Baxter J (2000) A model of inductive bias learning. J Artif Intell Res 12:149–198
5. Bui D, Malik K, Goetz J, Liu H, Moon S, Kumar A, Shin KG (2019) Federated user representation learning. arXiv preprint arXiv:190912535
6. Caldas S, Duddu SMK, Wu P, Li T, Konečnỳ J, McMahan HB, Smith V, Talwalkar A (2018) Leaf: a benchmark for federated settings. arXiv preprint arXiv:181201097
7. Caruana R (1997) Multitask learning. Mach Learn 28(1):41–75
8. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority oversampling technique. J Artif Intell Res 16:321–357
9. Chen F, Luo M, Dong Z, Li Z, He X (2018) Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:180207876
10. Chen M, Suresh AT, Mathews R, Wong A, Allauzen C, Beaufays F, Riley M (2019) Federated learning of n-gram language models. In: Proceedings of the 23rd conference on computational natural language learning (CoNLL), pp 121–130
11. Cheng Y, Wang D, Zhou P, Zhang T (2017) A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:171009282
12. Chernozhukov V, Chetverikov D, Demirer M, Duflo E, Hansen C, Newey W, Robins J (2017) Double/debiased machine learning for treatment and causal parameters. arXiv:160800060 [econ, stat] 1608.00060
13. Cohen G, Afshar S, Tapson J, Van Schaik A (2017) EMNIST: extending MNIST to handwritten letters. In: 2017 international joint conference on neural networks (IJCNN). IEEE, pp 2921–2926
14. Corinzia L, Beuret, A, Buhmann JM (2019) Variational federated multi-task learning. arXiv preprint arXiv:190606268
15. Deng Y, Kamani MM, Mahdavi M (2020) Adaptive personalized federated learning. arXiv preprint arXiv:200313461
16. Dinh CT, Tran NH, Nguyen TD (2020) Personalized federated learning with Moreau envelopes. arXiv preprint arXiv:200608848
17. Fallah A, Mokhtari A, Ozdaglar A (2020) Personalized federated learning: a meta-learning approach. arXiv preprint arXiv:200207948
18. Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning. PMLR, pp 1126–1135
19. Ghosh A, Hong J, Yin D, Ramchandran K (2019) Robust federated learning in a heterogeneous environment. arXiv preprint arXiv:190606629
20. Ghosh A, Chung J, Yin D, Ramchandran K (2020) An efficient framework for clustered federated learning. arXiv preprint arXiv:200604088
21. Gou J, Yu B, Maybank SJ, Tao D (2020) Knowledge distillation: a survey. arXiv preprint arXiv:200605525
22. Grbovic M, Cheng H (2018) Real-time personalization using embeddings for search ranking at Airbnb. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 311–320
23. Hanzely F, Richtárik P (2020) Federated learning of a mixture of global and local models. arXiv preprint arXiv:200205516
24. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. arXiv preprint arXiv:150302531

25. Hospedales T, Antoniou A, Micaelli P, Storkey A (2020) Meta-learning in neural networks: a survey. arXiv preprint arXiv:200405439
26. Huang Y, Chu L, Zhou Z, Wang L, Liu J, Pei J, Zhang Y (2021) Personalized cross-silo federated learning on non-IID data. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, pp 7865–7873
27. Jaech A, Ostendorf M (2018) Personalized language model for query auto-completion. arXiv preprint arXiv:180409661
28. Jeong E, Oh S, Kim H, Park J, Bennis M, Kim SL (2018) Communication-efficient on-device machine learning: federated distillation and augmentation under non-IID private data. arXiv preprint arXiv:181111479
29. Jiang Y, Konečnỳ J, Rush K, Kannan S (2019) Improving federated learning personalization via model agnostic meta learning. arXiv preprint arXiv:190912488
30. Johnson AE, Pollard TJ, Shen L, Li-Wei HL, Feng M, Ghassemi M, Moody B, Szolovits P, Celi LA, Mark RG (2016) MIMIC-III, a freely accessible critical care database. Sci Data 3(1):1–9
31. Khodak M, Balcan MF, Talwalkar A (2019) Adaptive gradient-based meta-learning methods. arXiv preprint arXiv:190602717
32. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images
33. Lancaster T (2000) The incidental parameter problem since 1948. J Econ 95(2):391–413. https://doi.org/10.1016/S0304-4076(99)00044-5
34. Li D, Wang J (2019) FedMD: Heterogenous federated learning via model distillation. arXiv preprint arXiv:191003581
35. Liang PP, Liu T, Ziyin L, Allen NB, Auerbach RP, Brent D, Salakhutdinov R, Morency LP (2020) Think locally, act globally: federated learning with local and global representations. arXiv preprint arXiv:200101523
36. Mansour Y, Mohri M, Ro J, Suresh AT (2020) Three approaches for personalization with applications to federated learning. arXiv preprint arXiv:200210619
37. Mariani G, Scheidegger F, Istrate R, Bekas C, Malossi C (2018) BAGAN: data augmentation with balancing GAN. arXiv preprint arXiv:180309655
38. McGraw I, Prabhavalkar R, Alvarez R, Arenas MG, Rao K, Rybach D, Alsharif O, Sak H, Gruenstein A, Beaufays F et al (2016) Personalized speech recognition on mobile devices. In: 2016 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, pp 5955–5959
39. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR, pp 1273–1282
40. Mikołajczyk A, Grochowski M (2018) Data augmentation for improving deep learning in image classification problem. In: 2018 international interdisciplinary PhD workshop (IIPhDW). IEEE, pp 117–122
41. Minka TP (2013) Expectation propagation for approximate Bayesian inference. arXiv preprint arXiv:13012294
42. Neyman J (1979) C(α) tests and their use. Sankhyā: Indian J Stat Ser A (1961–2002) 41(1/2):1–21
43. Nichol A, Achiam J, Schulman J (2018) On first-order meta-learning algorithms. arXiv:180302999 [cs] 1803.02999
44. Peterson D, Kanani P, Marathe VJ (2019) Private federated learning with domain adaptation. arXiv preprint arXiv:191206733
45. Raghu A, Raghu M, Bengio S, Vinyals O (2019) Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In: International conference on learning representations
46. Ruder S (2017) An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:170605098
47. Sattler F, Müller KR, Samek W (2020) Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans Neural Netw Learn Syst 1–13. https://doi.org/10.1109/TNNLS.2020.3015958

48. Schafer JB, Frankowski D, Herlocker J, Sen S (2007) Collaborative filtering recommender systems. In: Brusilovsky P, Kobsa A, Nejdl W (eds) The adaptive web: methods and strategies of web personalization. Springer, pp 291–324. https://doi.org/10.1007/978-3-540-72079-9_9
49. Shamir O, Srebro N, Zhang T (2014) Communication-efficient distributed optimization using an approximate Newton-type method. In: International conference on machine learning. PMLR, pp 1000–1008
50. Shen T, Zhang J, Jia X, Zhang F, Huang G, Zhou P, Wu F, Wu C (2020) Federated mutual learning. arXiv preprint arXiv:200616765
51. Shin M, Hwang C, Kim J, Park J, Bennis M, Kim SL (2020) XOR mixup: privacy-preserving data augmentation for one-shot federated learning. arXiv preprint arXiv:200605148
52. Shorten C, Khoshgoftaar TM (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):1–48
53. Smith V, Chiang CK, Sanjabi M, Talwalkar A (2017) Federated multi-task learning. arXiv preprint arXiv:170510467
54. Thrun S (1996) Is learning the n-th thing any easier than learning the first? In: Advances in neural information processing systems. Morgan Kaufmann Publishers, San Mateo, pp 640–646
55. Thrun S (1998) Lifelong learning algorithms. In: Learning to learn. Springer, Boston pp 181–209
56. Vanschoren J (2018) Meta-learning: a survey. arXiv preprint arXiv:181003548
57. Vilalta R, Drissi Y (2002) A perspective view and survey of meta-learning. Artif Intell Rev 18(2):77–95
58. Vu T, Nguyen DQ, Johnson M, Song D, Willis A (2017) Search personalization with embeddings. In: European conference on information retrieval. Springer, pp 598–604
59. Weng L (2018) Meta-learning: learning to learn fast. lilianwenggithubio/lil-log. http://lilianweng.github.io/lil-log/2018/11/29/meta-learning.html
60. Weyand T, Araujo A, Cao B, Sim J (2020) Google Landmarks Dataset v2-a large-scale benchmark for instance-level recognition and retrieval. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2575–2584
61. Yu P, Kundu A, Wynter L, Lim SH (2021) Fed+: a unified approach to robust personalized federated learning. 2009.06303
62. Yu T, Bagdasaryan E, Shmatikov V (2020) Salvaging federated learning by local adaptation. arXiv preprint arXiv:200204758
63. Yuksel SE, Wilson JN, Gader PD (2012) Twenty years of mixture of experts. IEEE Trans Neural Netw Learn Syst 23(8):1177–1193
64. Yurochkin M, Agarwal M, Ghosh S, Greenewald K, Hoang N, Khazaeni Y (2019) Bayesian nonparametric federated learning of neural networks. In: International conference on machine learning. PMLR, pp 7252–7261
65. Zhang Y, Yang Q (2017) A survey on multi-task learning. arXiv preprint arXiv:170708114
66. Zhao Y, Li M, Lai L, Suda N, Civin D, Chandra V (2018) Federated learning with non-IID data. arXiv preprint arXiv:180600582