Personalized Federated Learning With a Graph

1.摘要、介绍与相关工作

1.1 这篇文章在什么背景下解决了什么问题？现有的方法有什么问题？作者有什么贡献？

背景与问题：文章在联邦学习（Federated Learning，FL）的背景下提出了一种新的解决方案。联邦学习是一种协作式的机器学习框架，它的目标是在保护用户隐私的同时进行模型的训练和更新。然而，传统的FL方法，例如FedAvg，面临着非独立同分布（non-IID）的挑战。目前，已有的研究要么试图在服务端训练一个鲁棒的模型以解决非IID的问题，要么尝试学习多个全局模型或者为每个客户端学习特定的模型。但这些方法在知识共享和模型个性化的实现上有所不足，特别是在非IID程度很高的情况下，全局模型的质量可能较低，进而影响模型个性化的性能。

现有方法的问题：现有的个性化FL方法在模型个性化的改进上做了很多工作，但在知识共享的实现上，它们通常只是简单地聚合所有的本地模型。这样的实现忽视了具有非IID数据的客户之间的图关系。此外，如果非IID的程度非常高，所有客户的聚合将产生一个质量低下的全局模型，最终影响模型个性化的性能。

作者的贡献：作者提出了一种新的结构化联邦学习（Structured Federated Learning，SFL）方法，利用客户间的关系图来增强个性化FL。作者设计了一种细粒度的模型聚合机制，利用每个客户邻居的本地模型。此外，作者提出使用图卷积网络（Graph Convolutional Network，GCN）实现模型聚合功能，从而使得这种解决方案易于实现。作者还将问题描述为一个统一的优化框架，既包括个性化FL，也包括基于图的模型聚合。此外，作者的方法已经扩展到学习客户之间的隐藏关系，并且概念框架可以扩展以与其他模型个性化技术集成。最后，作者在图像和交通数据集上的实验验证了所提方法的有效性。

1.2 在文章的Introduction中提到clients的关系图是保存在server端的，这是否会泄露用户的隐私？

在联邦学习的背景下，客户端的关系图被保存在服务器端并不一定会导致用户隐私的泄露。这主要取决于关系图中包含的具体信息以及如何使用和保护这些信息。

关系图是一种表现不同数据源之间关联性的方式，而并不直接存储用户的个人信息。例如，关系图可能描述的是哪些客户端有相似的数据分布，而不是特定的个人数据。关系图帮助服务器更好地理解数据的结构，并指导模型的训练，以提高模型的性能。

然而，如果关系图包含可以被追溯到个人的敏感信息，或者被不当使用，确实存在泄露用户隐私的风险。因此，创建和使用关系图应遵循数据最小化和隐私保护的原则。例如，可以使用技术手段（如差分隐私或安全多方计算）来保护关系图中的信息，以防止潜在的隐私泄露。在具体实施过程中，也需要考虑适用的法律法规，例如GDPR在欧洲的实施，对于用户数据的处理都有严格的要求。

总的来说，虽然客户端关系图被存储在服务器端可能存在一定的隐私风险，但是通过正确的数据管理和保护手段，可以有效地控制这些风险。

1.3 现有的个性化联邦学习方法主要有哪两种?

现有的个性化联邦学习方法主要有两种：集群式个性化联邦学习（Group-wise PFL）和客户端个性化联邦学习（Client-wise PFL）。

1. **集群式个性化联邦学习（Group-wise PFL）**：这种方法也被称为集群化联邦学习，它假设客户可以被聚类到具有严重非IID特性的不同组中。因此，根据不同的聚类方法和距离度量，可以将集群化的FL进行分类。例如，Kmeans-based集群化FL [Xie et al., 2021] 和 [Mansour et al., 2020; Ghosh et al., 2020] 分别使用模型参数和准确度来度量距离。另外，层次聚类方法（Hierarchical clustering）[Briggs et al., 2020]也被应用到联邦学习中。此外，[Ma et al., 2022] 提出了一种通用的形式来将集群化FL问题模型化为一个双层优化框架，然后利用客户之间的重要贡献来形成一个基于权重的客户端集群化FL框架。
2. **客户端个性化联邦学习（Client-wise PFL）**：这种方法通常假设每个客户的数据分布与其他客户的数据分布不同，因此，每个客户应该在其设备上有一个个性化的模型。一般来说，一个简单的PFL方法可以在FedAvg中训练一个全局模型，然后在每个客户端上进行几步微调 [Cheng et al., 2021]。在这个框架中，知识共享是模型聚合，模型个性化是本地微调。Per-FedAvg [Fallah et al., 2020] 认为微调是全局模型学习目标函数的一个正则化项。Ditto [Li et al., 2021] 被提出作为一个考虑约束局部模型和全局模型之间距离的正则化项的PFL的双层优化框架。此外，有些研究 [Shamsian et al., 2021; Chen et al., 2018] 旨在训练一个全局超网络或元学习器，而不是一个全局模型，然后将其发送给客户进行本地优化。SCAFFOLD [Karimireddy et al., 2020] 提出学习个性化的控制变量来根据需要修正本地模型。逐层个性化 [Arivazhagan et al., 2019; Liang et al., 2020] 和表示层个性化[Tan et al., 2021] 是PFL的两种简单但有效的解决方案。

1.4 为什么Per-FedAvg方法将微调过程视为全局模型学习目标函数的一个正则化项

在机器学习中，正则化是一种防止模型过拟合的技术，它通过向模型的目标函数添加一个额外的惩罚项来约束模型的复杂度，使模型对训练数据的拟合程度和模型复杂度之间达到某种平衡。这个额外的惩罚项就被称为正则化项。

在Per-FedAvg中，这个正则化项就是微调过程。在这里，微调是指在已经训练好的全局模型基础上，进一步根据每个客户端的本地数据进行优化，以适应每个客户端的特定需求。将微调视为一个正则化项，可以帮助约束全局模型在每个客户端的本地数据上的表现，从而在全局模型和个性化模型之间找到一个平衡。

简单来说，这就是将全局模型的微调过程看作是一个正则化项，从而在全局和个性化之间找到平衡，避免模型在某些客户端数据上过度拟合，同时在其他客户端数据上表现不佳。

1.5 介绍一下GCN

图卷积网络（Graph Convolutional Network，简称GCN）是一种专门处理图结构数据的深度学习模型，是图神经网络（Graph Neural Network，简称GNN）的一种。它可以直接在图上进行高效的前向传播，适用于处理包含丰富关系信息的复杂数据。

在介绍GCN的工作流程之前，先解释一下图结构数据。图是由节点（Node）和边（Edge）组成的，节点代表实体，边代表实体之间的关系。边可以有方向，也可以无方向。边可能带有权重，表示关系的强度或者重要性。

GCN的基本结构主要包括两部分：卷积层和池化层。

1. 卷积层：GCN的卷积层的作用是提取图中节点的特征。这是通过将每个节点的特征与其相邻节点的特征结合起来实现的。具体操作是，对于每个节点，GCN会将其特征与所有相邻节点的特征进行线性变换（例如，加权和），然后通过一个激活函数（例如，ReLU）进行非线性变换。这个过程可以看作是在图上进行卷积操作。
2. 池化层：GCN的池化层的作用是降低图的复杂性。池化层可以对节点进行聚合操作，如最大池化、平均池化或和池化等，从而产生一个更低维度的节点表示。

GCN的工作流程如下：

1. 初始化：给每个节点分配一个初始特征向量。
2. 卷积：对每个节点，将其特征与其邻居节点的特征结合，生成新的特征表示。
3. 非线性变换：通过激活函数对新的特征表示进行非线性变换。
4. 池化：对图中的所有节点进行池化操作，降低特征的维度。
5. 重复步骤2-4：重复卷积、非线性变换和池化的步骤，直到达到预定的层数或满足其他停止条件。
6. 输出：将最后一层的节点特征作为图的表示，输入到分类器或其他任务中。

GCN的优点在于它可以有效地处理图结构数据，能捕获节点之间的复杂关系，并保留节点和图的全局信息。因此，GCN在社交网络分析、生物信息学、交通预测等领域有广泛的应用。

2. Structured Federated Learning

2.1 Structured Federated Learning的问题定义

The optimization problem of **Structured Federated Learning**

$$\min_{v_{1:N};u_{1:N};A} \quad \sum_{i=1}^{N}\left(F_i\left(v_i\right)+\lambda\mathcal{R}\left(v_i,w^*,u_i\right)\right)+\gamma\mathcal{G}\left(A,A^*\right) \tag{1}$$

$$\text{s.t.} \quad w^*\in\arg\min_w G\left(F_1(w),\ldots,F_N(w)\right) \tag{2}$$

$$A_{i,j}^*=S\left(u_i,u_j\right) \tag{3}$$

$$\|A_i\|\leq K \tag{4}$$

1). $F_i(v_i)$ 是第 i个客户端在本地数据集 D_i 训练下得到的本地模型 v_i 的supervised loss

2). $\mathcal{R}\left(v_i, w^*, u_i\right)=\mathcal{R}\left(v_i, w^*\right)+\mathcal{R}\left(v_i, u_i\right)$ 是计算本地模型参数 v_i 与全局模型参数 w^* 或个性化模型参数 u_i 之间的差异。这个差异被用作正则化项，以确保在优化过程中，本地模型参数 v_i 不会偏离全局模型参数 w^* 或个性化模型参数 u_i 太远。注意，v_i和u_i都是客户端 i 的模型。

3). $\mathcal{G}\left(A, A^*\right)$ 是邻接矩阵 A 和基于模型参数关联矩阵 A^* 之间的距离，是约束学习图 A 及其稀疏表示A^* 之间的距离，其中 A^* 是稀疏的,能够保护重要的距离之间的关系的客户。

4). $S\left(u_i, u_j\right)$ 两个关系模型参数的欧几里得距离。

2.2 Structured Federated Learning的工作流程是怎么样的?

结构化联邦学习（Structured Federated Learning, SFL）的工作流程主要包括以下步骤：

1. **本地更新**：在每个通信轮次中，每个客户端i都会更新其本地模型参数v_i。这是通过进行本地模型训练与正则化项来完成的。具体的更新公式为：

   $v_i^{(t)}=v_i^{(t)}-\eta \nabla\left(F_i(v_i^{(t)})+\lambda[R(v_i^{(t)}, w^{(t)})+R(v_i^{(t)}, u_i^{(t)})]\right)$

   其中，$F_i(v_i^{(t)})$是客户端i的本地损失函数，λ是正则化系数，$R(v_i^{(t)}, w^{(t)})$和$R(v_i^{(t)}, u_i^{(t)})$是正则化项，η是学习率。
2. **基于结构的聚合**：在服务器端，使用图卷积网络（Graph Convolutional Network, GCN）进行模型聚合。首先，将所有客户端的本地模型参数v_i作为输入，然后通过GCN更新每个客户端的个性化模型参数u_i。具体的更新公式为：

   $u_i^{(t+1)}=\mathrm{GCN}(A, \{u_i^{(t+1)}\}_{i=1}^N)$

   其中，A是邻接矩阵，表示客户端之间的关系。

   然后，通过GCN的read-out操作来更新全局模型参数w。具体的更新公式为：

   $w^{(t+1)}=\text{GCN readout}(\{u_i^{(t+1)}\}_{i=1}^N)$
3. **（可选）结构学习**：如果邻接矩阵A不存在，我们将添加一个结构学习步骤。具体的更新公式为：

   $A=\text{Structure learn}(\{v_i^{(t+1)}, u_i^{(t+1)}, w^{(t+1)}\}_{i=1}^N)$

这个过程会在多个通信轮次中迭代进行，直到达到预定的通信轮次数或满足其他停止条件。

2.3 什么是GCN的read-out operator?

在图卷积网络（Graph Convolutional Network, GCN）中，read-out操作是一种特殊的操作，它的目标是从图的所有节点特征中提取全局信息。这通常涉及到对所有节点的特征进行某种形式的聚合，例如求和、平均或最大化。

在结构化联邦学习（Structural Federated Learning）的上下文中，read-out操作被用来更新全局模型参数w。具体来说，它通过聚合所有个性化模型参数ui来更新全局模型参数w。这样，全局模型参数w就能够反映所有客户端的本地模型的共享信息。

总的来说，GCN的read-out操作是一种从图的全局结构中提取信息的方法，它在处理图形数据时非常重要，因为图形数据的全局结构通常包含了重要的信息。

2.4 在文章开源的代码中GCN模型结构是怎么样的?

在这个GitHub仓库的代码中，并没有明确地定义一个图卷积网络（GCN）的模型结构。相反，它使用了一种基于图的参数聚合方法，这种方法的思想与GCN的思想类似，但并没有显式地定义一个GCN模型。

具体来说，这个代码中的graph_dic函数实现了一种基于图的参数聚合方法。在每个通信轮次中，它将每个客户端的模型参数看作一个节点，然后使用邻接矩阵和参数矩阵进行矩阵乘法操作，得到新的参数矩阵。这个过程可以看作是GCN的前向传播过程，其中邻接矩阵代表了图的结构，参数矩阵代表了节点的特征。

然而，这个过程并没有显式地定义一个GCN模型，也没有进行GCN的反向传播和参数优化。这是因为在联邦学习中，每个客户端的模型参数是在本地进行更新的，然后再使用这种基于图的参数聚合方法进行聚合。因此，这个过程更像是在使用GCN的思想，而不是在使用一个明确的GCN模型。

在开源的代码中：

```
aggregated_param = torch.mm(A, param_metrix)
    for i in range(args.layers - 1):
        aggregated_param = torch.mm(A, aggregated_param)
    new_param_matrix = (args.serveralpha * aggregated_param) + ((1 - args.serveralpha)
* param_metrix)
```

是使用GCN来聚合本地模型。

首先，aggregated_param = torch.mm(A, param_metrix)这行代码是在进行矩阵乘法操作，将邻接矩阵A和参数矩阵param_metrix相乘。在这个上下文中，邻接矩阵A表示客户端之间的连接关系，参数矩阵param_metrix的每一行表示一个客户端的模型参数向量。因此，这个矩阵乘法操作的结果是一个新的参数矩阵，其中的每一行是原始参数矩阵的行（即客户端的模型参数向量）与邻接矩阵的对应行（即与该客户端连接的所有客户端的连接强度）的加权和。

然后，for i in range(args.layers - 1): aggregated_param = torch.mm(A, aggregated_param)这两行代码是在进行多层的图卷积操作。在每一层中，都使用邻接矩阵A和当前的参数矩阵aggregated_param进行矩阵乘法操作，得到新的参数矩阵。这样，每一层的图卷积操作都会使模型参数的更新考虑到更远的邻居。

最后，new_param_matrix = (args.serveralpha * aggregated_param) + ((1 - args.serveralpha) * param_metrix)这行代码是在进行线性组合操作，将聚合后的参数矩阵aggregated_param和原始参数矩阵param_metrix进行线性组合，得到新的参数矩阵。这里的args.serveralpha是一个权重参数，用于控制聚合后的参数和原始参数在新的参数中的比例。