Personalized Hyper-parameter Tuning for Federated Learning

Background

超参数在机器学习中具有重要的作用，它们对于训练出优秀的模型至关重要。然而，获得一组良好的超参数往往需要耗费大量的计算资源。而在联邦学习中，参与训练的主体通常是用户的私人设备，它们的计算能力通常较弱，且进行训练所需的代价也相对较高。另一方面，由于本地数据集具有异质性，采用相同的超参数可能无法充分适应每个客户端的特殊情况。相反，为不同客户端提供个性化的超参数将能够更好地缓解数据异质性问题，并显著提高模型的性能。因此，我们需要一种能够个性化调整联邦学习超参数的解决方案。

Policy Gradient based Approach

Environment

在我们的系统中，服务器需要为所有客户端决策最佳超参数，因此可以将其抽象为一个RL问题。

考虑包含 C 个客户端的FL系统。每个客户端的本地模型为 θ_i ，聚合后的全局模型为 $\theta_g = \sum_{i=1}^{C} \frac{D_i}{D} \theta_i$ 。在每一通信轮中，智能体策略 α 需要依次为每个客户端决策个性化的超参数，以最

大化模型在验证集上的精度。这是一个**完全延迟奖励的确定性环境**，即：

1. **完全延迟奖励：**智能体的reward必须等为所有客户端都完成决策以后才能得到，而在决策过程中，智能体不能得到任何奖励的反馈。
2. **确定性环境：**环境的状态转移概率是1，即为确定性环境。在这种设定下，策略 α 为客户端 u_i 做出决策 a_i 的观测值为 $s_i=(a_1,a_2,a_{i-1})$ ，环境的状态转移概率为 $p(s_{i+1}|s_i,a_i)=1$ ，环境不存在不确定性，因此为确定性环境。

Normal Policy Gradient Baseline

对于一个决策轨迹 $\tau=(a_1,a_2,\cdots,a_C)$ ，其出现的概率为：

$$
\begin{aligned}
p_\alpha(\tau)&=p(s_1)p_\alpha(a_1|s_1)p(s_2|s_1,a_1)\cdots p_\alpha(a_C|s_C)p(s_{C+1}|s_C,a_C)\\
&=p(s_1)\prod_{i=1}^{C}p_\alpha(a_i|s_i)\prod_{i=1}^{C}p(s_{i+1}|s_i,a_i)\\
&=\prod_{i=1}^{C}p_\alpha(a_i|s_i)=p_\alpha(a_1)p_\alpha(a_2|a_1)\cdots p_\alpha(a_C|a_1,\cdots,a_{C-1})\\
&=p_\alpha(a_1,a_2,\cdots,a_C)
\end{aligned}
$$

因此对于策略 α ，其所能获得的奖励期望为：

$$
\bar{R}=\mathbb{E}_{\tau\sim p_\alpha(\tau)}[R(\tau)]=\sum_{\tau}R(\tau)p_\alpha(\tau)
$$

这里的 R 为决策轨迹 τ 所获得的奖励，通常设置为 θ_g 在验证集上的精度Acc。为了优化 $\bar{R}$ ，需要求其相对于策略 α 的梯度，有：

$$
\begin{aligned}
&\nabla_\alpha\bar{R}=\sum_{\tau}R(\tau)\nabla_\alpha p_\alpha(\tau)\\
&=\sum_{\tau}R(\tau)p_\alpha(\tau)\nabla_\alpha\log p_\alpha(\tau)\\
&=\mathbb{E}_{\tau\sim p_\alpha(\tau)}[R(\tau)\nabla_\alpha\log p_\alpha(\tau)]
\end{aligned}
$$

因此策略 α 迭代为：

$$
\alpha=\alpha+R(\tau)\nabla_\alpha\log p_\alpha(\tau)
$$

Reward Assignment

由于更新策略的环境是一个完全延迟奖励环境，因此用以上方式更新策略虽然能收敛，但是存在收敛速度慢，收敛不稳定等问题。因此我们希望能够设计一种机制，使得奖励能够根据各个客户端的贡献按需分配。

将最大化reward R=Acc 转换为最小化regret L=Loss，随后通过反向传播来分配奖励。这里的L为 θ_g 在验证集上的损失。

先对训练流程做如下等价转变：

1. 服务器先为每个客户端选择超参数。
2. 客户端根据选择的超参数进行本地训练。
3. 服务器聚合训练结果。

1. 客户端为每一组超参数都训练一个模型。
2. 服务器根据策略 α 为每个客户端选择一个 local model。
3. 服务器聚合选好的 local model。

将客户端 u_i 所有训练后的本地模型表示为 $\Theta_i = [\theta_i^1, \theta_i^2, \cdots, \theta_i^H]^T$ ， H 为候选超参数的个数。则根据策略 α_i 选择后的模型为 $\theta_i = Z_i^T \Theta_i$ ，其中 $Z_i \sim p_{\alpha_i}(Z_i)$ 是一个one-hot的 Hx1 向量，被选中的超参数对应的位置为1。则聚合后的globe model可表示为：

$$\theta_g = \sum_{i=1}^{C} \frac{D_i}{D} \theta_i = \sum_{i=1}^{C} \frac{D_i}{D} Z_i^T \Theta_i$$

则globe model在验证集上的损失为：

$$L = \mathbb{E}_{(x,y)\sim \mathcal{D}_{val}}[CrossEntropy(\theta_g(x), y)]$$

于是Loss相对于策略的偏微分为：

$$\frac{\partial L}{\partial \alpha_i} = \frac{\partial L}{\partial \theta_g} \frac{\partial \theta_g}{\partial Z_i} \frac{\partial Z_i}{\partial \alpha_i}$$

这里由于 Z_i 是根据 α_i 采样得到的，无法进行反向传播。所以我们采用重参数技巧，将 Z_i 定义为：

$$\tilde{Z}_i = [\tilde{Z}_i^1, \tilde{Z}_i^2, \cdots, \tilde{Z}_i^H]^T,$$

$$where \quad \tilde{Z}_i^k = f_{\alpha_i}(G_i^k) = \frac{\exp((\log \alpha_i^k + G_i^k)/\lambda)}{\sum_{j=1}^{H} \exp((\log \alpha_i^j + G_i^j)/\lambda)}$$

这里的 G_i^k 是一个满足gumble分布的随机变量，其可以从均匀分布中抽样获得，即 $G_i^k = -\log(-\log(U_i^k))$ ， U_i^k 为满足均匀分布的随机变量。于是有：

$$\begin{aligned}
\mathbb{E}_{\tilde{Z}_i \sim p_{\alpha_i}(\tilde{Z}_i)}[\frac{\partial \tilde{Z}_i}{\partial \alpha_i}] &= \mathbb{E}_{U_i \sim Uniform}[\frac{\partial f_{\alpha_i}(-\log(-\log(U_i)))}{\partial \alpha_i}] \\
&= \int_0^1 p(U_i) \frac{\partial f_{\alpha_i}(-\log(-\log(U_i)))}{\partial \alpha_i} \mathrm{d}U_i \\
&= \frac{\partial}{\partial \alpha_i} \int p_{\alpha_i}(\tilde{Z}_i) \tilde{Z}_i \mathrm{d}\tilde{Z}_i \\
&= \int p_{\alpha_i}(\tilde{Z}_i) \frac{\partial \log p_{\alpha_i}(\tilde{Z}_i)}{\partial \alpha_i} \tilde{Z}_i \mathrm{d}\tilde{Z}_i \\
&= \mathbb{E}_{\tilde{Z}_i \sim p(\tilde{Z}_i)}[\nabla_{\alpha_i} \log p_{\alpha_i}(\tilde{Z}_i) \tilde{Z}_i]
\end{aligned}$$

所以策略 α 的策略梯度为：

$$\mathbb{E}_{\tilde{Z}_i \sim p_{\alpha_i}(\tilde{Z}_i)}[\frac{\partial L}{\partial \alpha_i}] = \mathbb{E}_{\tilde{Z}_i \sim p_{\alpha_i}(\tilde{Z}_i)}[\frac{\partial L}{\partial \tilde{Z}_i}\frac{\partial \tilde{Z}_i}{\partial \alpha_i}] = \mathbb{E}_{\tilde{Z}_i \sim p_{\alpha_i}(\tilde{Z}_i)}[\nabla_{\alpha_i} \log p_{\alpha_i}(\tilde{Z}_i)\frac{\partial L}{\partial \tilde{Z}_i}\tilde{Z}_i]$$

但此时的 $\tilde{Z}_i$ 是一个soft one-hot vector，为了使其变成严格的one-hot，需要令 $\lambda \to 0$ ，则有：

$$\lim_{\lambda \to 0}\left\{\mathbb{E}_{\tilde{Z}_i \sim p_{\alpha_i}(\tilde{Z}_i)}[\nabla_{\alpha_i} \log p_{\alpha_i}(\tilde{Z}_i)\frac{\partial L}{\partial \tilde{Z}_i}\tilde{Z}_i]\right\} = \mathbb{E}_{Z_i \sim p_{\alpha_i}(Z_i)}[\nabla_{\alpha_i} \log p_{\alpha_i}(Z_i)\frac{\partial L}{\partial Z_i^s}]$$

因此策略 α 迭代为：

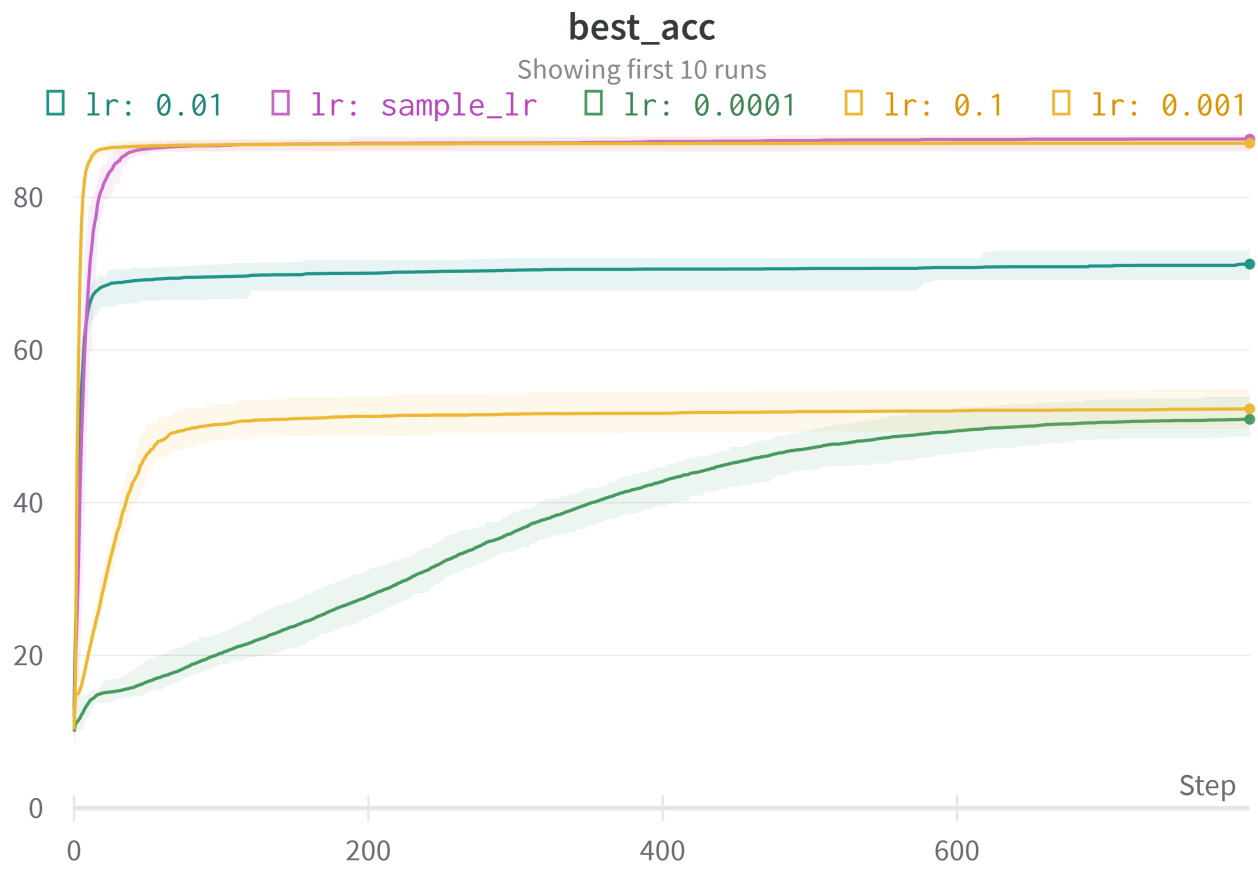
$$\alpha_i = \alpha_i - \nabla_{\alpha_i} \log p_{\alpha_i}(Z_i)\frac{\partial L}{\partial Z_i^s}$$

Training Process

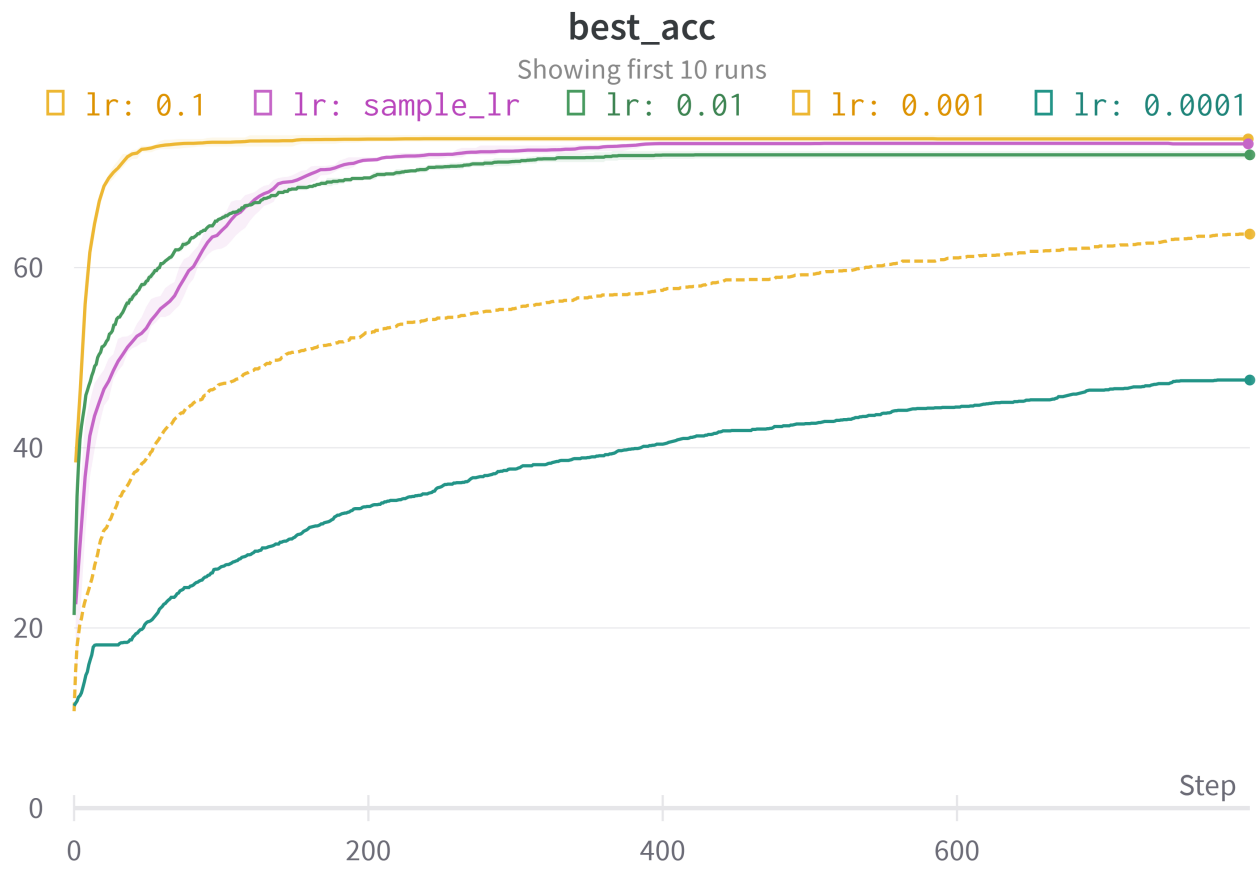
1. 服务器根据策略 α 为每个客户端决策超参数。
2. 各个客户端根据服务器配置的超参数进行本地训练，并上传训练结果 θ_i 。
3. 服务器聚合模型参数 $\theta_g = \sum_{i=1}^{C}\frac{D_i}{D} \cdot 1_i \cdot \theta_i$ 。
4. 服务器用验证集在 θ_g 上前向/反向传播，计算相对于dummy one的梯度： $\frac{\partial L}{\partial 1_i}$ 。
5. 服务器计算梯度： $\nabla_{\alpha_i} \log p_{\alpha_i}(Z_i)$ 。
6. 服务器将前两步计算的梯度相乘，并将其用来更新策略： $\alpha_i = \alpha_i - \nabla_{\alpha_i} \log p_{\alpha_i}(Z_i)\frac{\partial L}{\partial 1_i}$ 。
7. 服务器下发 θ_g ，返回第1步，开启下一轮训练。

Preliminary results

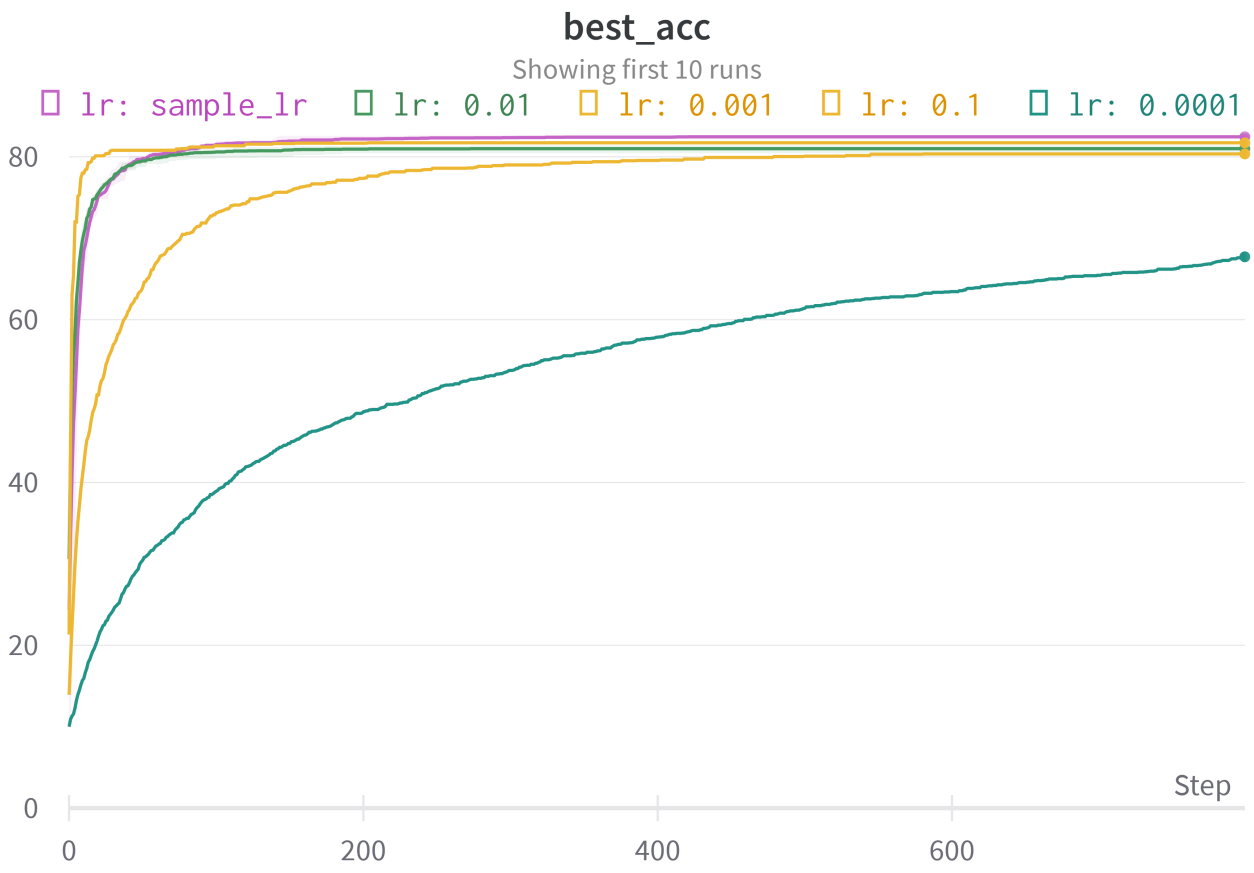
ResNet18


best_acc
Showing first 10 runs
lr: 0.01
lr: sample_lr
lr: 0.0001
lr: 0.1
lr: 0.001
80
60
40
20
0
Step
0
200
400
600

MLP


best_acc
Showing first 10 runs
lr: 0.1
lr: sample_lr
lr: 0.01
lr: 0.001
lr: 0.0001
60
40
20
0
Step
0
200
400
600

Conv


best_acc
Showing first 10 runs
lr: sample_lr
lr: 0.01
lr: 0.001
lr: 0.1
lr: 0.0001
80
60
40
20
0
Step
0
200
400
600