4. Method

To solve the challenges of model access, communication cost, and computational cost simultaneously, we propose **Fed**erated **B**lack-box **P**rompt **T**uning method (FedBPT) to train an optimal prompt in a federated fashion by adapting BBT to federated learning. Unlike FL methods communicating model parameters, the clients in FedBPT train and communicate with the server prompts rather than the model parameters, which is communication efficient. To optimize prompts, the clients only need to conduct inference rather than back-propagation, significantly reducing the computational cost and memory usage. The FL server aggregates the local prompts uploaded by the client and is completely agnostic to the employed LLM architecture. During training, the clients can treat the model as a black box: neither the clients nor the server requires access to the PLM parameters.

The key **technical** challenge and contribution of our work are enabling the CMA-ES algorithm to be effective in a distributed/federated setting. Existing CMA-ES algorithms strictly require tracking each search step, which is intractable to the server in the federated learning settings. We have theoretically overcome this limitation, successfully maintaining the efficacy of the CMA-ES algorithm within the context of federated learning.

4.1. Problem Formulation

4.2. Overview of FedBPT

4.3. Server-level CMA-ES Algorithm

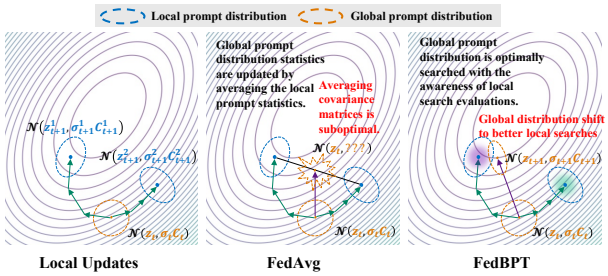

Figure 2. Comparison of aggregation between directly using FedAvg and FedBPT. FedAvg derives the global distribution by directly averaging the local distribution statistics. Mathematically, the arithmetic mean of the covariance matrices is not equivalent to the covariance matrix of our targeted optimal global distribution. In FedBPT, the server applies CMA-ES to derive the global prompt distributions with the awareness of the evaluation results of the uploaded local distributions.

After receiving local CMA-ES parameters, the server conducts aggregation on the server to derive a global search distribution that can guide the clients' search in the next communication round. Directly averaging the models uploaded by the clients following FedAvg is not effective for FedBPT. In FedBPT, the clients locally optimize the CMA-ES parameters parameterized by multivariate normal distribution statistics. Mathematically, the arithmetic mean of the covariance matrices is not equivalent to the covariance matrix of our targeted optimal global distribution, as is empirically shown in Sec. 5.2. In addition, CMA-ES is a random search algorithm that cannot guarantee to achieve a local optimum as with gradient-based optimization algorithms. Directly averaging optimal and inferior local search results makes it difficult to achieve a global optimum. To derive an optimal global search distribution on the server, we adopt a server-level CMA-ES algorithm to update the search distribution statistics based on the local search results. The comparison of aggregations by directly applying FedAvg and FedBPT is shown in Figure 2.

The intuition of the server-level CMA-ES is to consider the local search results as a set of solutions sampled by the server. The server then evaluates these sampled solutions and updates the search distributions for the next communication round. Suppose a set of clients $\mathbb{S}_t$ participate in training in the t-th communication round. The server-level CMA-ES takes the received mean vectors $\{\boldsymbol{z}_{t+1}^k\}_{k\in\mathbb{S}_t}$ as the sampled solutions and the local loss values $\{F^k(\boldsymbol{z}_{t+1}^k)\}_{k\in\mathbb{S}_t}$ as the corresponding search step loss. To update the CMA-ES parameters, the **search step length** is required. **However, the server-level "sampling" is conducted by multiple local search steps, and the server-level search step length σ_t is intractable.** Directly applying a local search step length causes the model to diverge. We provide a theoretical explanation for this divergence later. To solve this problem, we theoretically derive a corrected search step σ_t' on the server formulated as

$$\sigma_t' = 2\sqrt{\sum_{k\in\mathbb{S}_t'}\sum_{j=1}^{I}\left(\sigma_{t,j}^k\right)^2 / \left(|\mathbb{S}_t|\cdot\lambda_k\right)}, \tag{4}$$

where $\mathbb{S}_t'$ is the set of $\frac{|\mathbb{S}_t|}{2}$ clients that upload z_{t+1}^k with the lowest local loss values $F^k(\boldsymbol{z}_{t+1}^k)$. $\sigma_{t,j}^k$ is the step length of client k's j-th local search iteration in communication round t. I is the number of local search iterations, and λ_k is the local search population of client k. Then, the overall algorithm of FedBPT is shown in Algorithm 1.

We then introduce how we derive the corrected search step σ_t'. After the server receives the locally updated prompt vectors $\{z_{t+1}^k\}_{k\in\mathbb{S}_t}$ and the corresponding loss values $\{F^k(\boldsymbol{z}_{t+1}^k)\}_{k\in\mathbb{S}_t}$, the server updates the CMA-ES parameters as shown in Algorithm 2.

Without the loss of generality, we assume that $\left[z_{t+1}^1, ..., z_{t+1}^{|\mathbb{S}_t|}\right]$ is ordered satisfying that $F^1(\boldsymbol{z}_{t+1}^1) < ... < F^{|\mathbb{S}_t|}(\boldsymbol{z}_{t+1}^{|\mathbb{S}_t|})$ as stated in line 1 of Algorithm 2. The

Algorithm 1 Training Algorithm of FedBPT.

Server executes:

1: initialize the projection matrix $\boldsymbol{A}$ and distribute it to the clients
2: initialize the global CMA-ES parameters $\{\boldsymbol{z}_0, \sigma_0, \boldsymbol{C}_0\}$
3: **for** each round $t = 0, 1, \dots$ **do**
4: **for** each client $k \in S_t$ **in parallel do**
5: $\{\boldsymbol{z}_{t+1}^k, \{\sigma_{t,j}^k\}_{j\in[I]}, F^k(\boldsymbol{z}_{t+1}^k)\} \leftarrow$ ClientUpdate $(\boldsymbol{z}_t, \sigma_t, \boldsymbol{C}_t)$
6: **end for**
7: $\sigma_t' = 2\sqrt{\sum_{k\in\mathbb{S}_t'}\sum_{j=1}^{I}\left(\sigma_{t,j}^k\right)^2 / \left(|\mathbb{S}_t| \cdot \lambda_k\right)}$
8: $\{\boldsymbol{z}_{t+1}, \sigma_{t+1}, \boldsymbol{C}_{t+1}\} \leftarrow$ CMA-ES$\left(\left\{\boldsymbol{z}_{t+1}^k, F^k\left(\boldsymbol{z}_{t+1}^k\right)\right\}_{k\in S_t}; \boldsymbol{z}_t, \sigma_t', \boldsymbol{C}_t\right)$
9: **end for**

ClientUpdate$(\boldsymbol{z}_t, \sigma_t, \boldsymbol{C}_t)$**:**

10: $\boldsymbol{z}_{t,1}^k, \sigma_{t,1}^k, \boldsymbol{C}_{t,1}^k \leftarrow \boldsymbol{z}_t, \sigma_t, \boldsymbol{C}_t$
11: **for** each local iteration j from 1 to $I-1$ **do**
12: Randomly sample a set of binary masks M_j^k with the same shape of X^k with a rate r_p of elements that are zeros
13: Randomly sample a set of tokens $\hat{X}^k$ with the same shape of X^k
14: **for** $i \in \lambda_k$ **do**
15: $\boldsymbol{z}_{t,j,i}^k \sim \mathcal{N}\left(\boldsymbol{z}_{t,j}, \sigma_{t,j}\boldsymbol{C}_{t,j}\right)$
16: $\hat{F}^k(\boldsymbol{z}_{t,j,i}^k) = \frac{\mathcal{L}\left(f(\boldsymbol{A}\boldsymbol{z}_{t,j,i}^k; X^k), Y^k\right)}{\mathcal{L}\left(f(\boldsymbol{A}\boldsymbol{z}_{t,j,i}^k; X^k \odot M_j^k + \hat{X}^k \odot (1-M_j^k)), Y^k\right)}$
17: **end for**
18: $\left\{\boldsymbol{z}_{t,j+1}^k, \sigma_{t,j+1}^k, \boldsymbol{C}_{t,j+1}^k\right\} \leftarrow$ CMA-ES$\left(\left\{\boldsymbol{z}_{t,j,i}^k, \hat{F}^k\left(\boldsymbol{z}_{t,j,i}^k\right)\right\}_{i\in[\lambda_k]}; \boldsymbol{z}_{t,j}^k, \sigma_{t,j}^k, \boldsymbol{C}_{t,j}^k\right)$
19: **end for**
20: $\boldsymbol{z}_{t+1}^k \leftarrow \boldsymbol{z}_{t,I}^k$
21: $F^k(\boldsymbol{z}_{t+1}^k) = \mathcal{L}\left(f(\boldsymbol{A}\boldsymbol{z}_{t+1}^k; X^k), Y^k\right)$
22: return $\{\boldsymbol{z}_{t+1}^k, \{\sigma_{t,j}^k\}_{j\in[I]}, F^k(\boldsymbol{z}_{t+1}^k)\}$ to server

server updates z_{t+1} following

$$z_{t+1} = \sum_{k=1}^{\mu}\frac{1}{\mu}z_{t+1}^k = z_t + \sum_{k=1}^{\mu}\frac{1}{\mu}(z_{t+1}^k - z_t), \tag{5}$$

where we apply *Rank-μ-Update* (Hansen, 2016) and the μ best z_{t+1}^k with lowest z_{t+1}^k are averaged to update z_{t+1}. There is no issue to update z_{t+1} on the server (line 2 in Algorithm 2), but to update σ_{t+1} and $\boldsymbol{C}_{t+1}$, the intermediate coefficients *evolution path* values $p_{\sigma,t+1}$ and $p_{c,t+1}$ for this round should be derived first (line 3&4 in Algorithm 2). For simplicity, we derive σ_t' from the computation of $p_{\sigma,t+1}$, which is also applicable to $p_{c,t+1}$. CMA-ES derives the evolution path $p_{\sigma,t+1}$ following

$$p_{\sigma,t+1} \leftarrow (1-c_\sigma)p_{\sigma,t} + \sqrt{1-(1-c_\sigma)^2}\sqrt{\mu}C_t^{-1/2}\frac{z_{t+1}-z_t}{\sigma_t}, \tag{6}$$

Algorithm 2 CMA-ES update.

CMA-ES$(\{z_i, f_i\}_{i\in[\lambda]}, z, \sigma, C, p_\sigma, p_c)$**:**

1: $z_{1\dots\lambda} \leftarrow z_{s(1)\dots s(\lambda)}$ with $s(i) = \text{argsort}(f_{1\dots\lambda}, i)$
2: $z' \leftarrow \frac{1}{\mu}\sum_{k=1}^{\mu} z_i$
3: $p_\sigma \leftarrow$ Update-$p_\sigma(z', z, C, p_\sigma)$
4: $p_c \leftarrow$ Update-$p_c(z', z, C, p_\sigma, p_c)$
5: $\sigma' \leftarrow \sigma \times \exp(\frac{|p_\sigma|}{E|N(0,I)|} - 1)$
6: $C' \leftarrow (1 - c_1 - c_\mu + c_s)C + c_1 p_c p_c^T + \frac{1}{\mu c_\mu}\sum_{k=1}^{\mu}\frac{z_i - z}{\sigma}(\frac{z_i - z}{\sigma})^T$
7: Return $z', \sigma', C', p_\sigma, p_c$

where c_σ is an artificial hyper-parameter satisfying $c_\sigma \leq 1$ (Hansen, 2016). σ_t is intractable in FedBPT, and we need to derive an estimated σ_t' to conduct CMA-ES on the server correctly. The key to estimating the global search step on the server is to guarantee that the term $\sqrt{\mu}C_t^{-1/2}\frac{z_{t+1}-z_t}{\sigma_t}$ follows a standard normal distribution

$$\sqrt{\mu}C_t^{-1/2}\frac{z_{t+1}-z_t}{\sigma_t} \sim \mathcal{N}(0, I) \tag{7}$$

under neutral selection, which means that the server randomly selects z_{t+1}^k to update the CMA-ES parameters. Based on this rule of estimation, we first derive the distribution of $z_{t+1} - z_t$. From Equation (5), we have

$$z_{t+1} - z_t = \sum_{k=1}^{\mu}\frac{1}{\mu}(z_{t+1}^k - z_t), \tag{8}$$

where z_{t+1}^k is formulated as

$$z_{t+1}^k = z_t + \sum_{j=1}^{J}\sigma_{t,j}^k \times \sum_{i=1}^{\mu_l}\frac{1}{\mu_l}z_{t,j,i}^k \sim \mathcal{N}\left(0, C_{t,j}^k\right), \tag{9}$$

where J is the number of local iterations in one round of training, and μ_l is the rank of local *Rank-μ-Update*. $\sigma_{t,j}^k$ is the search step length of client k's j-th iteration in round t. $z_{t,j,i}^k$ is the i-th sampled point in client k's j-th iteration of search in round t. When the clients conduct limited iterations of local training in one round, we make an assumption that the local covariance matrix $C_{t,j}^k$ in one round will not change significantly, then we have

$$z_{t+1}^k \approx z_t + \sum_{j=1}^{J}\sigma_{t,j}^k \times \sum_{i=1}^{\mu_l}\frac{1}{\mu_l}z_{t,j,i}^k \sim \mathcal{N}\left(0, C_t^k\right). \tag{10}$$

Therefore, we have

$$\begin{aligned}&\frac{C_t^{-1/2}(z_{t+1}-z_t)}{\sqrt{\sum_{k=1}^{\mu}\left(\frac{1}{\mu}\right)^2\sum_{j=1}^{J}\left(\sigma_{t,j}^k\right)^2\sum_{i=1}^{\mu_l}\left(\frac{1}{\mu_l}\right)^2}}\\ =&\sqrt{\mu}\frac{C_t^{-1/2}(z_{t+1}-z_t)}{\sqrt{\sum_{k=1}^{\mu}\sum_{j=1}^{J}\left(\sigma_{t,j}^k\right)^2/(\mu\cdot\mu_l)}} \sim \mathcal{N}(0, I).\end{aligned} \tag{11}$$

Compared with Equation (7), we derive an estimated global search step length as

$$\sigma_t' = \sqrt{\sum_{k=1}^{\mu}\sum_{j=1}^{J}\left(\sigma_{t,j}^k\right)^2 / (\mu \cdot \mu_l)}. \tag{12}$$

In this paper, we set $\lambda_1 = ... = \lambda_K$ and $\mu_l = \frac{\lambda_k}{2}$, where λ_k is the local population size of the k-th client, and we set $\mu = \frac{|\mathbb{S}_t|}{2}$. Then without the restriction on the order of $\left[z_{t+1}^1, ..., z_{t+1}^{|\mathbb{S}_t|}\right]$, we have

$$\sigma_t' = 2\sqrt{\sum_{k\in\mathbb{S}_t'}\sum_{j=1}^{J}\left(\sigma_{t,j}^k\right)^2 / (|\mathbb{S}_t| \cdot \lambda_k)}, \tag{13}$$

where $\mathbb{S}_t'$ is the set of $\frac{|\mathbb{S}_t|}{2}$ clients that upload z_{t+1}^k with the lowest local loss values $F^k(\boldsymbol{z}_{t+1}^k)$.

4.4. Local Black-box Prompt Tuning against Overfitting