Federated Submodel Averaging

Yucheng Ding, Chaoyue Niu, Fan Wu
Shanghai Jiao Tong University
China
{yc.ding, rvince, wu-fan}@sjtu.edu.cn

Shaojie Tang
University of Texas at Dallas
USA
shaojie.tang@utdallas.edu

Chengfei Lv
Alibaba Group
China
chengfei.lcf@alibaba-inc.com

Yanghe Feng
National University of Defense Technology
China
fengyanghe@nudt.edu.cn

Guihai Chen
Shanghai Jiao Tong University
China
gchen@cs.sjtu.edu.cn

Abstract

We study federated learning from the new perspective of feature heat, where distinct data features normally involve different numbers of clients, generating the differentiation of hot and cold features. Meanwhile, each client’s local data tend to interact with part of features, updating only the feature-related part of the full model, called a submodel. We further identify that the classical federated averaging algorithm (FedAvg) or its variants, which randomly selects clients to participate and uniformly averages their submodel updates, will be severely slowed down, because different parameters of the global model are optimized at different speeds. More specifically, the model parameters related to hot (resp., cold) features will be updated quickly (resp., slowly). We thus propose federated submodel averaging (FedSubAvg), which introduces the number of feature-related clients as the metric of feature heat to correct the aggregation of submodel updates. We theoretically prove that due to the dispersion of feature heat, the global objective is ill-conditioned, and FedSubAvg works as a diagonal preconditioner, making the global model easier to converge to local minima and escape saddle points. We also rigorously analyze FedSubAvg’s convergence rate to stationary points. We finally evaluate FedSubAvg over several public and industrial datasets. The evaluation results demonstrate that FedSubAvg significantly outperforms FedAvg and its variants.

1 Introduction

Federated learning (FL) [1] allows a large number of clients (e.g., millions of smartphone users) to collaborate in the training of a global machine learning (ML) model under the coordination of a cloud server without sharing raw data. As a new paradigm of distributed ML, the data characteristics in FL significantly differ from those in the traditional distributed optimization [2][3][4][5]. Different clients have diverse behavior patterns, and their local data are normally non-independent and identically distributed (non-i.i.d.). As a result, the local model trained on a client is biased to the client’s local data distribution. To mitigate the bias when updating the global model, [1] proposed federated averaging (FedAvg) in the seminal work that averages the participating clients’ local models. Later, much effort was devoted to proving the convergence of FedAvg over non-i.i.d. data. One line of work [6][7][8][9][10] established an $O(1/\sqrt{NT})$ convergence, where $N$ denotes the total number of clients and $T$ denotes the total number of iterations. However, these work required all the clients to participate in each round of FL, which is not practical in cross-device FL. Another line of work [11][12][13][14][15][16] allowed partial client participation and proved an $O(1/\sqrt{KT})$ convergence of FedAvg, where $K$ denotes the number of chosen clients in each round. Some other work proposed variants of FedAvg to better
We then consider that a client’s local data involve only part of the full global model, namely, a submodel, which is a subset of the full model. The submodel is denoted by $\xi$. We focus on a general distributed optimization scenario, in which the local updates of the chosen clients, just like FedAvg, are performed for the full global model. However, for some hot items involving nearly 100% of clients (resp., some cold items involving less than 1% of clients), the corresponding item embeddings are involved by a majority (resp., minority) of the clients’ submodels. In expectation, at the end of each round in FedAvg and its variants, the embeddings related to the hot items will be updated approximately by adding the average of the involved clients’ updates, while the embeddings related to the cold items will be updated by adding less than 1% of the average of the involved clients’ updates. The inaccurate aggregation without considering the feature heat dispersion leads to different parameters of the full model being optimized at different speeds.

The existing work on FL has not well studied the issue of feature heat dispersion. However, it will severely deteriorate the performance of FedAvg and its variants. We take an extreme example in RS for illustration. For some hot items involving nearly 100% of clients (resp., some cold items involving less than 1% of clients), the corresponding item embeddings are involved by a majority (resp., minority) of the clients’ submodels. In expectation, at the end of each round in FedAvg and its variants, the embeddings related to the hot items will be updated approximately by adding the average of the involved clients’ updates, while the embeddings related to the cold items will be updated by adding less than 1% of the average of the involved clients’ updates. The inaccurate aggregation without considering the feature heat dispersion leads to different parameters of the full model being optimized at different speeds.

To deal with feature heat dispersion, we propose federated submodel averaging (FedSubAvg), which first averages the local updates of the chosen clients, just like FedAvg, but then multiplies the aggregate update of each model parameter with the ratio between the total number of clients and the number of clients who involve this model parameter. Such a small correction ensures that the expectation of each model parameter’s global update is equal to the average of the local updates of the clients who involve this parameter. We theoretically demonstrate the advantage of FedSubAvg over FedAvg and analyze the convergence of FedSubAvg. We first prove that the global objective is ill-conditioned, leading to the slow convergence of FedAvg, and FedSubAvg works as a diagonal preconditioner, making the global model easier to converge to local minima and escape saddle points. We finally obtain an $O(\sqrt{N/(n_{\text{min}}KT)})$ convergence rate with respect to stationary points, where $n_{\text{min}}$ denotes the minimum of the number of clients who involve each individual model parameter.

We summarize the key contributions: (1) To the best of our knowledge, we are the first to make an in-depth study of FL from the differentiation of hot and cold data features, which is common in practice; (2) we identify the defect of FedAvg and its variants in handling feature heat dispersion and propose a new and efficient algorithm FedSubAvg; (3) we theoretically show that the global objective is ill-conditioned, and FedSubAvg works as a preconditioner for acceleration. We also give the convergence rate of FedSubAvg in the general non-convex case; and (4) using the public MovieLens, Sentiment140, and Amazon datasets, as well as an industrial dataset from Alibaba, we evaluate FedSubAvg and compare its performance with FedAvg, FedProx, and Scaffold. The evaluation results reveal the superiority of FedSubAvg from faster convergence and smaller train loss.

## 2 Problem Formulation

We focus on a general distributed optimization scenario, in which $N$ clients collaboratively solve the following consensus optimization problem:

$$f (\mathbf{X}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\xi_i \sim D_i} [F (\mathbf{X}, \xi_i)] = \frac{1}{N} \sum_{i=1}^{N} f_i (\mathbf{X}),$$

where $\xi_i \sim D_i$ denotes the local training data of client $i$; $F (\mathbf{X}, \xi_i)$ is the train loss of the full global model $\mathbf{X}$ over the local data $\xi_i$; and $f_i (\mathbf{X})$ is the local generalization error, taking expectation over the randomness of the local data.

We then consider that a client’s local data involve only part of the full global model, namely, a submodel, which is related to the client’s local data features. We use $S = \{1, 2, \cdots, M\}$ to index the parameters of the full model $\mathbf{X} \in \mathbb{R}^M$ and call it the full index set. We let $S(i) \subseteq S$ denote the index set of client $i$’s submodel $\mathbf{X}_{S(i)}$. In other words, after client $i$’s local training, only the submodel $\mathbf{X}_{S(i)}$ will be updated. This further implies that the local loss over the full model is equal to that over the submodel. Therefore, we can rewrite the global objective function in a distributed submodel way:

$$f (\mathbf{X}) = \frac{1}{N} \sum_{i=1}^{N} f_i (\mathbf{X}_{S(i)}).$$
Federated Submodel Averaging

The gradient of $f$ is $\nabla f(X) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(X_{S(i)})$, and the Hessian is $H \triangleq \nabla^2 f(X) = \frac{1}{N} \sum_{i=1}^{N} \nabla^2 f_i(X_{S(i)}) = \frac{1}{N} \sum_{i=1}^{N} H_i$. Note that when doing any operation (e.g., summation) over multiple submodels, gradients, and Hessians, they will be automatically aligned according to the indices.

We finally introduce the metric of feature heat dispersion (resp., the resulting parameter heat dispersion), which is defined as the ratio between the maximum and the minimum of the number of clients who involve each individual feature (resp., parameter). Considering the fact that a parameter may involve one or multiple data features in NLP and RS\footnote{For example, an embedding vector corresponds to a certain word in NLP or an item in RS, while the fully connected layer is related to all the words or items.} the feature heat dispersion is a lower bound of the parameter heat dispersion. In other words, high feature heat dispersion inevitably leads to high parameter heat dispersion. For convenience and clarity in the submodel-level analysis, we mainly take the metric of parameter heat dispersion. We let $n_m$ denote the number of clients involving the parameter with index $m \in S$ and set $n_{\text{max}} = \max_{m \in S} n_m$, $n_{\text{min}} = \min_{m \in S} n_m$. Then, the parameter heat dispersion is $n_{\text{max}}/n_{\text{min}}$.

3 Algorithm Design

In this section, we first show that FedAvg suffers from high parameter heat dispersion in distributed submodel optimization and then propose FedSubAvg to remedy the defect.

3.1 Slow Convergence of FedAvg

Example 1. We consider a special distributed convex optimization problem with two model parameters, denoted as $w_1$ and $w_2$. Each client $i$’s local data $\xi_i \sim D_i$ are with mean $e_i = \mathbb{E}[\xi_i] = 0$. In addition, $w_1$ involves only client 1, while $w_2$ involves all the $N$ clients. Then, the parameter heat dispersion is $n_2/n_1 = N$. We can formulate this learning problem as minimizing the mean square error:

$$f((w_1, w_2)) = \frac{1}{N} \sum_{i=1}^{N} f_i((w_1, w_2)_{S(i)}),$$

where $f_i((w_1, w_2)_{S(i)}) = \mathbb{E}_{\xi_i \sim D_i}[(w_1, w_2) - \xi_i]^2$; and for $i = 2, 3, \ldots, N$, $f_i((w_1, w_2)_{S(i)}) = \mathbb{E}_{\xi_i \sim D_i}[(w_2 - \xi_i)^2] = w_2^2 + \mathbb{E}_{\xi_i \sim D_i}[\xi_i^2 - e_i^2]$.

For this example, the optimal model is $(w_1^*, w_2^*) = (0, 0)$. We leverage FedAvg with only one local iteration and let each client compute the exact (not stochastic) gradient. The learning rate is denoted as $\eta$, and the model is initialized as $(w_1^0, w_2^0)$. After $r$ rounds, the model will become

$$(w_1^r, w_2^r) = \left[1 - \frac{2\eta}{N} 0\right] (w_1^0, w_2^0)^\top.$$

By choosing $\eta = 0.5$, $(w_1^r, w_2^r) = ((1 - 1/N)^r w_1^0, 0)$. We can find that in the FL scenario with high parameter heat dispersion $N$, $w_1$ will converge at a quite low speed.

3.2 Federated Submodel Averaging

To mitigate parameter heat dispersion, we propose FedSubAvg. As shown in Figure\footnote{For example, an embedding vector corresponds to a certain word in NLP or an item in RS, while the fully connected layer is related to all the words or items.} compared with FedAvg, the key principle of FedSubAvg is to further multiply the aggregate update of each model parameter with the ratio between the total number of clients and the number of clients who involve this model parameter (i.e., for the model parameter with index $m$, the correction coefficient is $N/n_m$). We still examine Example\footnote{For example, an embedding vector corresponds to a certain word in NLP or an item in RS, while the fully connected layer is related to all the words or items.} for illustration. FedSubAvg will multiply the aggregate update of $w_1$ and $w_2$ with $N$ and 1, respectively. By correction, the model at the $r$-th round is:

$$(w_1^r, w_2^r) = \left[1 - \frac{2\gamma}{N} 0\right] (w_1^0, w_2^0)^\top,$$

where $\gamma$ is the learning rate of FedSubAvg. Therefore, FedSubAvg converges quickly to the optimal model $(0, 0)$. We also depict the optimization processes of FedSubAvg and FedAvg for Example\footnote{For example, an embedding vector corresponds to a certain word in NLP or an item in RS, while the fully connected layer is related to all the words or items.} in Figure\footnote{For example, an embedding vector corresponds to a certain word in NLP or an item in RS, while the fully connected layer is related to all the words or items.} when the parameter heat dispersion is 100. We can observe that FedSubAvg greatly outperforms FedAvg from convergence speed and loss.
We now present the design details of FedSubAvg in Algorithm 1. In each round $r$ of FL, the cloud server first selects $K$ clients to participate (Line 3), denoted as $C_r$. Each selected client $i \in C_r$ determines its index set $S(i)$ of submodel based on the local training set (Line 12). Then, client $i$ uses $S(i)$ to download the submodel $X_{S(i)}^r$ from the cloud server and initializes the local submodel $x_i^{r,1}$ (Lines 13–14). Client $i$ locally trains its submodel by doing $I$ iterations of stochastic gradient descent (SGD) (Lines 15–17) and uploads the submodel update $\Delta x_i^{r,j}$ (Line 18). After receiving the submodel updates from the selected clients, the cloud server performs aggregation for each index in the union of the participating clients’ index sets and updates the global full model (Lines 7–10). In particular, for the global model parameter with $m$, namely, $X_{\{m\}}$, its expected update, after being corrected with the coefficient $N/n_m$, is

$$
\mathbb{E}_{C_r} \left[ \Delta X_{\{m\}} \right] = \frac{N}{n_m} \left( \frac{1}{N} \sum_{i=1}^{N} \Delta x_i^{r,j} \right) = \frac{1}{n_m} \sum_{\{m \in S(i)\}} \Delta x_i^{r,j},
$$

which is equal to the average of the local updates of the participating clients involving it, as required.

We find the algorithm of FedSubAvg easier to read if we consider the total number of clients $N$, the number of involved clients $n_{inv}$, and the selected clients $K$. If $N = 3$ clients (Alice, Bob, Charlie), then $n_{inv} = 2$ and $K = 3 / 2$. If $N = 100$ clients, then $n_{inv} = 10$ and $K = 10 / 100 = 0.1$. Therefore, the average number of involved clients per round is $\frac{n_{inv}}{K} = \frac{2}{3 / 2} = \frac{4}{3}$.

We now present the design details of FedSubAvg in Algorithm 1. In each round $r$ of FL, the cloud server first selects $K$ clients to participate (Line 3), denoted as $C_r$. Each selected client $i \in C_r$ determines its index set $S(i)$ of submodel based on the local training set (Line 12). Then, client $i$ uses $S(i)$ to download the submodel $X_{S(i)}^r$ from the cloud server and initializes the local submodel $x_i^{r,1}$ (Lines 13–14). Client $i$ locally trains its submodel by doing $I$ iterations of stochastic gradient descent (SGD) (Lines 15–17) and uploads the submodel update $\Delta x_i^{r,j}$ (Line 18). After receiving the submodel updates from the selected clients, the cloud server performs aggregation for each index in the union of the participating clients’ index sets and updates the global full model (Lines 7–10). In particular, for the global model parameter with $m$, namely, $X_{\{m\}}$, its expected update, after being corrected with the coefficient $N/n_m$, is

$$
\mathbb{E}_{C_r} \left[ \Delta X_{\{m\}} \right] = \frac{N}{n_m} \left( \frac{1}{N} \sum_{i=1}^{N} \Delta x_i^{r,j} \right) = \frac{1}{n_m} \sum_{\{m \in S(i)\}} \Delta x_i^{r,j},
$$

which is equal to the average of the local updates of the participating clients involving it, as required.

We now present the design details of FedSubAvg in Algorithm 1. In each round $r$ of FL, the cloud server first selects $K$ clients to participate (Line 3), denoted as $C_r$. Each selected client $i \in C_r$ determines its index set $S(i)$ of submodel based on the local training set (Line 12). Then, client $i$ uses $S(i)$ to download the submodel $X_{S(i)}^r$ from the cloud server and initializes the local submodel $x_i^{r,1}$ (Lines 13–14). Client $i$ locally trains its submodel by doing $I$ iterations of stochastic gradient descent (SGD) (Lines 15–17) and uploads the submodel update $\Delta x_i^{r,j}$ (Line 18). After receiving the submodel updates from the selected clients, the cloud server performs aggregation for each index in the union of the participating clients’ index sets and updates the global full model (Lines 7–10). In particular, for the global model parameter with $m$, namely, $X_{\{m\}}$, its expected update, after being corrected with the coefficient $N/n_m$, is

$$
\mathbb{E}_{C_r} \left[ \Delta X_{\{m\}} \right] = \frac{N}{n_m} \left( \frac{1}{N} \sum_{i=1}^{N} \Delta x_i^{r,j} \right) = \frac{1}{n_m} \sum_{\{m \in S(i)\}} \Delta x_i^{r,j},
$$

which is equal to the average of the local updates of the participating clients involving it, as required.
4 Theoretical Analysis

In this section, we first prove that in FL with high parameter heat dispersion, the global objective \( f \) is ill-conditioned. We then prove that optimizing \( f \) with FedSubAvg approximates to optimizing a preconditioning objective \( \hat{f} \) with gradient descent (GD), thereby remedying ill conditioning. In particular, the diagonal preconditioning matrix comprises of the correction coefficients \( \{N/n_m|m \in S\} \) in FedSubAvg and can be obtained without needing to access any client’s local data and without any expensive computing, which keeps the tenet of FL. Further by analyzing the Hessian of \( \hat{f} \), we demonstrate that the model trained under \( \hat{f} \) is easier to converge to local minima and escape saddle points. We finally obtain the convergence guarantee for FedSubAvg.

4.1 FedAvg with Ill-Conditioned Global Objective

We analyze the condition number of the global objective \( f \) in the locally convex area. We first make an assumption about the eigenvalues of each client’s Hessian.

Assumption 4.1 (Bounded Hessian). For any model \( X \) and each client \( i \), the Hessian of \( f_i \) satisfies:

\[
-\lambda_2 I \preceq \nabla^2 f_i(X_{S(i)}) \preceq \lambda_2 I,
\]

In addition, if \( X \) is in a locally convex area of \( f \), the Hessian of \( f_i \) satisfies:

\[
\lambda_1 I \preceq \nabla^2 f_i(X_{S(i)}) \preceq \lambda_2 I
\]

with the probability of \( 1 - \alpha \), where \( 0 < \lambda_1 < \lambda_2 \) and \( \alpha \) is a constant with \((1 - \alpha)\lambda_1 - \alpha \lambda_2 > 0 \).

Assumption 4.1 bounds the eigenvalues of \( \nabla^2 f_i(X_{S(i)}) \) and ensures that when \( X \) is in a locally convex area of \( f \), \( X_{S(i)} \) is also in a locally convex area of \( f_i \) for most of the clients, which helps to obtain the bounds of the Hessians in expectation. Under Assumption 4.1, we can obtain the lower bound of the condition number of \( f \).

**Theorem 4.2.** Under Assumption 4.1 for any model \( X \) in a locally convex area, the condition number of the expected Hessian of \( f \) satisfies:

\[
\kappa(\mathbb{E}[H]) \geq \frac{n_{\max}(\lambda_1 - \alpha(\lambda_1 + \lambda_2))}{n_{\min} \lambda_2}.
\]

**Proof.** Please refer to Appendix A.

Theorem 4.2 reveals that when the parameter heat dispersion \( n_{\max}/n_{\min} \) is high, then the global objective is ill-conditioned. Specifically, in the large-scale NLP and RS scenarios, since the fully connected layer involves all the \( N \) clients, while the embedding vector of a cold word or item normally involves a few clients, we have \( \kappa(\mathbb{E}[H]) = \Theta(N) \). This further implies with a large \( N \) in practice, the global objective is extremely ill-conditioned. Therefore, the conventional FedAvg and its variants, which are the approximations to GD, will converge at a quite slow speed, even when the global model is in a locally convex area.

4.2 FedSubAvg as a Preconditioner Better than FedAvg

For the sake of brevity, we introduce \( t \) to denote the global iteration index, thus replacing the round index \( r \) and the local iteration index \( j \), where \( t = (r - 1) \times I + j \). We consider a single iteration of FedSubAvg:

\[
X^{t+1} = X^t - \gamma D \left( \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_i^t) \right) \approx X^t - \gamma D \nabla f(X^t),
\]

where \( X^t \triangleq D(\frac{1}{N} \sum_{i=1}^{N} x_i^t) \) is defined as the global model at iteration \( t \), and \( D = \text{diag}\{N/n_1, N/n_2, \ldots, N/n_M\} \).

By treating \( D \) as a preconditioning matrix \([20]\), we can construct a new objective \( \hat{f} \) with variable \( \hat{X} = D^{-\frac{1}{2}}X \):

\[
\hat{f}(\hat{X}) = f(D^{-\frac{1}{2}}X) = f(X).
\]

The gradient and the Hessian\( ^2 \)of \( \hat{f} \) are:

\[
\nabla \hat{f}(\hat{X}) = D^{\frac{1}{2}} \nabla f(X),
\]

\[
\nabla^2 \hat{f}(\hat{X}) = D^{\frac{1}{2}} HD^{\frac{1}{2}}, \text{ with } H = \nabla^2 f(X).
\]

\([20]\) showed that one SGD update for \( \hat{f} \) corresponds to equation 1. Therefore, we can compare FedSubAvg and FedAvg by comparing the characteristics of \( \hat{f} \) and \( f \).

\(^2\)The gradient and the Hessian of \( \hat{f} \) are with respect to \( \hat{X} \).

---

Federated Submodel Averaging
4.2.1 Convergence to Local Minima

We first analyze the condition number of \( \hat{\eta} \) where \( \hat{\eta} \) corresponds to Theorem 4.4.

**Theorem 4.3.** Under Assumption [4.7] for any model \( X \) in a locally convex area, the condition number of the expected corresponding Hessian, \( \mathbb{E}[\nabla^2 \hat{f}(X)] \), satisfies:

\[
\kappa(\mathbb{E}[\hat{H}]) \leq \frac{\lambda_2}{(\lambda_1 - \alpha(\lambda_1 + \lambda_2))}, \quad \text{with} \quad \hat{H} = \nabla^2 \hat{f}(\hat{X}).
\]

**Proof.** Please refer to Appendix B. \( \square \)

Theorem 4.3 indicates that \( \hat{H} \) is well-conditioned with \( \kappa(\mathbb{E}[\hat{H}]) \leq \Theta(1) \). Therefore, the proposed FedSubAvg, which is an approximation to GD for objective \( \hat{f} \), will converge efficiently if \( X \) is in a locally convex area of \( f \).

4.2.2 Escaping Saddle Points

We next study the update of \( X \) in FedAvg when it is near a strict saddle point \( X_0 \) with \( \lambda_{\min}(H_0) < 0 \), where \( H_0 = \nabla^2 f(X_0) \) and \( \lambda_{\min}(\cdot) \) denotes taking the minimum eigenvalue. Around \( X_0 \), \( f \) can be approximated by its second-order Taylor approximation:

\[
f(X) \approx f(X_0) + \frac{1}{2} (X - X_0)^\top H_0 (X - X_0). \tag{3}
\]

Starting from \( X_0^t \) near \( X_0 \), we can approximate the difference between the global model and the saddle point after \( t \)-th iteration by approximating FedAvg to GD:

\[
X_0^{t+1} - X_0 \approx (I - \eta H_0)^t (X_0^t - X_0)
\]

where \( \eta \) is the learning rate of FedAvg. Let \( H_0 = QAQ^\top \) denote the spectral decomposition of \( H_0 \). We have

\[
X_0^{t+1} - X_0 = Q (I - \eta A)^t Q^\top (X_0^0 - X_0).
\]

Generally, \( X \) will escape \( X_0 \) most efficiently in the direction with the greatest negative curvature. Let \( q \) denote the eigenvector corresponding to \( \lambda_{\min}(\nabla^2 f(X_0)) \), and \( X \) escapes \( X_0 \) exponentially in the direction of \( q \) when \( \| q^\top (X_0^t - X_0) \| > 0 \):

\[
d_{\text{FedAvg}}(q) \triangleq \| q^\top (X_0^t - X_0) \| = (1 - \eta \lambda_{\min}(H_0))^t \| q^\top (X_0^t - X_0) \| \tag{4}.
\]

In contrast, for FedSubAvg, when the update of \( X \) is near a strict saddle point \( X_0 \), by equation 2, it corresponds that \( \hat{X} \) is near a strict saddle point \( \hat{X}_0 \). We can obtain the relation between \( \lambda_{\min}(\nabla^2 f(X_0)) \) and \( \lambda_{\min}(\nabla^2 f(X_0)) \).

**Theorem 4.4.** For any strict saddle point \( X_0 \) of \( f \), we have

\[
\lambda_{\min}(H_0) \leq \frac{N}{n_{\max}} \lambda_{\min}(H_0),
\]

where \( \hat{H}_0 = \nabla^2 \hat{f}(\hat{X}_0) \) and \( H_0 = \nabla^2 f(X_0) \).

**Proof.** Please refer to Appendix C. \( \square \)

Similarly, the difference between \( \hat{X} \) and \( \hat{X}_0 \) after \( t \)-th iteration is given by

\[
\hat{X}_0^{t+1} - \hat{X}_0 \approx (I - \gamma \hat{H}_0)^t (\hat{X}_0^t - \hat{X}_0),
\]

which corresponds to

\[
X_0^{t+1} - X_0 = D^{\frac{1}{2}} (X_0^{t+1} - X_0) \approx D^{\frac{1}{2}} (I - \gamma \hat{H}_0)^t D^{-\frac{1}{2}} (X_0^t - X_0).
\]
Let $\hat{H}_0 = \hat{Q}\hat{\Lambda}\hat{Q}^T$ be the spectral decomposition of $\hat{H}_0$ and $\hat{q}$ be the eigenvector corresponding to $\lambda_{\min}(\hat{H}_0)$. We can construct a new unit vector $\hat{q}_1 = D^{-\frac{1}{2}}\hat{q}/\|D^{-\frac{1}{2}}\hat{q}\|$ such that in the direction of $\hat{q}_1$, $X$ escapes $X_0$ exponentially in FedSubAvg when $\|\hat{q}_1^T (X-t - X_0)\| > 0$:
\[
d_{\text{FedSubAvg}}(\hat{q}_1) \triangleq \|\hat{q}_1^T (X-t + X_0)\| = (1 - \gamma\lambda_{\min}(\hat{H}_0))^t \|\hat{q}_1^T (X-t - X_0)\|. \tag{5}
\]
We further consider how to choose $\eta$ and $\gamma$. Generally, the learning rate is set based on the maximum eigenvalue. Under Assumption 4.1, we can obtain the upper bound of the maximum eigenvalue of $H$ and $\hat{H}$.

**Theorem 4.5.** For any model $X$ (resp., $\hat{X}$), the maximum eigenvalue of $H$ (resp., $\hat{H}$) is bounded:
\[
\lambda_{\max}(H) \leq \frac{n_{\max}\lambda_2}{n}, \quad \lambda_{\max}(\hat{H}) \leq \lambda_2,
\]
where $\lambda_{\max}(\cdot)$ denotes taking the maximum eigenvalue.

**Proof.** Please refer to Appendix D

When choosing $\eta = N/(n_{\max}\lambda_2)$ and $\gamma = 1/\lambda_2$, by equations 4 and 5 we have $d_{\text{FedSubAvg}}(\hat{q}_1) \geq \Theta(d_{\text{FedAvg}}(\hat{q}_1))$, which demonstrates that when the global model is near a saddle point, there exists a direction (the direction of $\hat{q}_1$) in which FedSubAvg escapes the saddle point faster than FedAvg, which escapes the saddle point in the direction with the most negative curvature (the direction of $q$).

### 4.3 Convergence Guarantee for FedSubAvg

We analyze the convergence rate of FedSubAvg in the general non-convex case. Since FedSubAvg is an approximation of updating $\hat{f}$ with GD, while the convergence of GD is conventionally proved under the gradient norm, we thus leverage the squared gradient norm of $\hat{f}$, namely, $\|\nabla\hat{f}(\hat{X})\|^2 = \nabla\hat{f}(\hat{X})^T D\nabla\hat{f}(\hat{X})$, to characterize the convergence rate with respect to stationary points.

We next make the following assumptions about the objective functions, as well as the variance and the feasible space of the stochastic gradients.

**Assumption 4.6** (Smoothness), $f_i(\cdot)$ is $L$-smooth if
\[
\forall x, y, f_i(y) \leq f_i(x) + \langle y - x, \nabla f_i(x) \rangle + \frac{L}{2} \|x - y\|^2.
\]

**Assumption 4.7** (Bounded Variance). During local training, the variance of stochastic gradients on each client is bounded by $\sigma^2$:
\[
\forall i, t : \mathbb{E}_{\xi_i \sim D_i} \left[ \| \nabla f_i(x_i^t) - \nabla F(x_i^t, \xi_i) \| \right] \leq \sigma^2.
\]

**Assumption 4.8** (Bounded Gradient Norm). During local training, the expected $l_2$-norm of the stochastic gradients is bounded by a constant $G^2$:
\[
\forall i, t : \mathbb{E}_{\xi_i \sim D_i} \left[ \| \nabla F(x_i^t, \xi_i) \| \right] \leq G^2.
\]

Assumption 4.6 is standard. Assumptions 4.7 and 4.8 were widely made in the literature [7, 8, 10, 21, 12]. Under these assumptions, we can bound the gradient norm.

**Theorem 4.9.** Under Assumptions 4.6, 4.7, and 4.8 we can bound the expected average of the squared gradient norm
\[
\mathbb{E} \left[ \frac{1}{T} \sum_{i=1}^{T} \nabla f(X_i^t)^T D\nabla f(X_i^t) \right] \leq \frac{2(f(X^1) - f(X^*))}{\gamma T} + \frac{2\gamma L\sigma^2}{n_{\min}} + \frac{4N\gamma^2 F^2 G^2 L^2}{n_{\min}} + \frac{2\gamma N I^2 G^2 L}{n_{\min} K}.
\]

When $T > n_{\min}K^3/N$ and $\gamma = \Theta\left(\sqrt{\frac{n_{\min}K}{N}}\right)$, we have
\[
\mathbb{E} \left[ \frac{1}{T} \sum_{i=1}^{T} \nabla f(X_i^t)^T D\nabla f(X_i^t) \right] \leq O\left(\sqrt{\frac{N}{n_{\min}KT}}\right).
\]

**Proof.** Please refer to Appendix E

Theorem 4.9 indicates that the convergence rate of FedSubAvg improves substantially as the number of selected clients per round $K$ increases. In addition, without parameter heat dispersion (i.e., each feature or parameter involves all the clients), then $n_{\min} = N$, and the convergence result would reduce to $O(1/\sqrt{KT})$ in existing work.
Federated Submodel Averaging

![Graphs depicting train losses or test AUCs of FedSubAvg and the baselines on different datasets.](image)

**Figure 3**: Train losses or test AUCs of FedSubAvg and the baselines on different datasets.

**Table 1**: Statistics of four datasets.

|                       | # Clients | # Samples     | # Samples Per Client | Feature Heat Dispersion |
|-----------------------|-----------|---------------|----------------------|-------------------------|
| MovieLens             | 6,040     | 1,000,209     | 165                  | 4,331                   |
| Sent140               | 1,473     | 79,050        | 54                   | 1,451                   |
| Amazon                | 1,870     | 123,147       | 66                   | 232                     |
| Alibaba               | 49,023    | 16,864,641    | 344                  | 3,142                   |

5 Evaluation

In this section, we extensively evaluate the performance of FedSubAvg over several datasets with different feature heat dispersion. We also introduce FedAvg and its variants, including FedProx and Scaffold, as baselines for comparison.

5.1 Experimental Setups

We choose the following datasets, tasks, and models for evaluation. The statistics about clients and samples, as well as the feature heat dispersion are shown in Table 1. The parameter heat dispersion is equal to the total number of clients, because there exists network layer(s) in the model involving all the clients, and meanwhile, there exist data feature(s) (i.e., movies, words, or items) involving one client.

**LR for rating classification**: We perform a rating classification task over the MovieLens-1M dataset [22], which contains 6,040 users, 3,883 movies, and 1,000,209 samples. We preprocess the dataset to be suitable for binary classification. In particular, the original user ratings of movies range from 0 to 5. We label the samples with the ratings of 4 and 5 to be positive and label the rest to be negative. We randomly select 20% of the samples as the test dataset and leave the remaining 80% as the training dataset for FL. The task is to predict whether users will rate a given movie to be positive based on the user’s gender and age and on the movie ID. We first encode gender, age, movie, gender cross movie, and age cross movie, based on the one-hot encoder. Next, the features are input into a logistic regression (LR) model to predict the label.

**LSTM for sentiment analysis**: We perform a text sentiment classification task on the Sentiment140 dataset [23], which comprises 1,600,000 tweets collected from 659,775 twitter users. In this task, we use a two-layer long short-term memory (LSTM) network with 100 hidden units and an embedding layer as the binary classifier, where the embedding dimension is set to 25. We naturally partition this dataset by letting each Twitter account correspond to a client. We keep only the clients who hold more than 40 samples and get 1,473 clients in total. We randomly select 20% of the samples as the test dataset and leave the remaining 80% as the training set for FL.

**DIN for CTR prediction**: We perform a click-through rate (CTR) prediction task on the Amazon electronics dataset and an Alibaba industrial dataset. The Amazon dataset contains 1,689,188 reviews contributed by 192,403 users for 63,001 items. The ratings range from 0 to 5. We label the samples with the rating of 5 to be positive and label the rest to be negative. We naturally partition this dataset by letting each Amazon user correspond to a client. We keep only the clients who hold more than 40 samples and get 1,473 clients in total. We randomly select 20% of the samples as the test dataset and leave the remaining 80% as the training set for FL.
We use the following four baselines for comparison.

- **FedAvg** averages the local model updates from the participating clients to update the global model.
- **FedProx** is the first variant of FedAvg. The main difference from FedAvg is that FedProx adds a quadratic proximal term to explicitly limit the local model updates. We set the coefficient of the proximal term to 0.01 throughout the evaluation.
- **Scaffold** is another important variant of FedAvg. The key difference from FedAvg is that each client keeps a variate to control the local model updates in Scaffold. However, the size of the control variate is equal to size of the full model, which is prohibitively inefficient for the learning tasks with large-scale full models. Therefore, for the CTR prediction tasks, we make an approximation to Scaffold. In particular, the cloud server performs the controlled update step every round by weighted averaging the historical updates. Please refer to Appendix F.2 for details.
- **CentralSGD** runs the standard SGD algorithm to train the global model using the whole dataset, sets the number of iterations in each round as $K$, and sets the batch size to the sum of the selected clients’ local batch sizes in each round. This ensures the same amount of data per round with the distributed algorithms.

Regarding the experimental settings, we choose mini-batch SGD as the optimization algorithm. For the tasks of rating classification and sentiment analysis, $K = 50$ clients are randomly chosen per round as default; and for the CTR prediction tasks, $K$ is set to 100 as default. The settings of the other hyperparameters are deferred to Appendix F.3.

### 5.2 Evaluation Results

We first present the results of FedSubAvg and the baselines under the default $K$. We then vary $K$ to show its impact.

**FedSubAvg vs. Baselines:** For the rating classification on the MovieLens dataset and the sentiment analysis on the Sent140 dataset, we plot the train loss in Figure 3(a) and Figure 3(b) and for the CTR prediction on Amazon and the Alibaba dataset, we plot the test area under the curve (AUC) in Figure 3(c) and 3(d). In addition, we measure the convergence rates of different algorithms by counting the communication rounds to reach a target train loss or test AUC. We set the target loss in the rating classification (resp., the sentiment analysis) to be the minimum loss of CentralSGD, which is 0.325 (resp., 0.380); and we set the target test AUC to be 0.6 in two CTR prediction tasks. The results are listed in Table 2.

From Figure 3 and Table 2 we observe that FedSubAvg consistently outperforms FedAvg and FedProx. Specifically, (1) in the rating classification, FedSubAvg always has the smallest train loss during FL and reaches the target at the 100-th communication round, $1.7 \times$ faster than FedAvg and FedProx, and $1.8 \times$ faster than Scaffold and CentralSGD; (2) in the sentiment analysis, FedSubAvg still has the smallest train loss during FL and reaches the target at the 260-th

---

3The positive and negative samples in CTR datasets (especially the Alibaba dataset) are extremely uneven. Even if all the samples are predicted to be negative (or positive), the train loss is very small. As a result, the train losses of different algorithms are hard to distinguish, and we choose to plot the test AUCs instead.
Federated Submodel Averaging

Table 2: Number of communication rounds to reach the target train loss or test AUC with different algorithms. #+ indicates that the target was not reached even after # rounds.

|                  | CentralSGD | FedAvg   | FedProx  | Scaffold | FedSubAvg |
|------------------|------------|----------|----------|----------|-----------|
| MovieLens        | 180        | 170      | 170      | 180      | 100       |
| Sent140          | 980        | 1,000+   | 1,000+   | 1,000+   | 260       |
| Amazon           | 20         | 200+     | 200+     | 200+     | 53        |
| Alibaba          | 265        | 5,000+   | 5,000+   | 5,000+   | 610       |

Table 3: Number of communication rounds for FedSubAvg to reach the target train loss or test AUC with the varying number of selected clients per round $K$.

|                  | MovieLens | Sent140 |
|------------------|-----------|---------|
| $K$              | 10        | 30      | 50      |
| Rounds           | 100       | 110     | 100     |
|                  | Amazon    | Alibaba |
| $K$              | 20        | 60      | 100     |
| Rounds           | 95        | 57      | 53      | 3,469    | 1,030 | 610    |

round, $3.77 \times$ faster than CentralSGD, while FedAvg, FedProx, and Scaffold cannot reach the target even in 1,000 rounds; (3) in the CTR prediction on the Amazon dataset, FedSubAvg performs the best among all the FL algorithms. FedSubAvg achieves the highest AUC of 0.641 in 200 rounds, decreasing by 0.013 in terms of AUC compared with the ideal CentralSGD. In contrast, FedAvg achieves the highest AUC of 0.523, FedProx achieves the highest AUC of 0.519, and Scaffold achieves the highest AUC of 0.514. In addition, FedSubAvg reaches the target test AUC at the 53-th round, while the other FL algorithms cannot reach the target even after 200 rounds; and (4) in the CTR prediction on the Alibaba dataset, FedSubAvg still outperforms all the other FL algorithms. FedSubAvg achieves the highest AUC of 0.626 in 5,000 rounds, decreasing by 0.016 compared with CentralSGD. In contrast, FedProx achieves the highest AUC of 0.514, FedAvg achieves the highest AUC of 0.509, and Scaffold achieves the highest AUC of 0.507. Moreover, FedSubAvg reaches the target test AUC at the 610-th round, whereas the other FL algorithms cannot reach the target even after 5,000 rounds.

**Impact of participating clients:** We next evaluate the impact of the number of selected clients $K$ per round on FedSubAvg. We set $K = 10, 30, 50$ for the rating classification and the sentiment analysis, and set $K = 20, 60, 100$ for the CTR prediction. We plot the results in Figure 4 and record the minimum number of rounds to reach the target in Table 3. We observe that FedSubAvg with a larger $K$ generally converges much faster, which validates the speedup with respect to $K$. (1) In the rating classification, FedSubAvg with different $K$ behaves somewhat uniformly. This is because a larger $K$ improves the convergence by reducing the variance of the global model update, while for such a simple convex optimization scenario, the variance is already small enough with $K = 10$; (2) in the sentiment analysis, FedSubAvg with $K = 50$ reaches the target train loss $1.62 \times$ faster than FedSubAvg with $K = 10$; (3) in the CTR prediction on the Amazon dataset, FedSubAvg with $K = 100$ reaches the target test AUC $1.79 \times$ faster than FedSubAvg with $K = 20$; and (4) in the CTR prediction on the Alibaba dataset, FedSubAvg with $K = 100$ reaches the target test AUC $5.69 \times$ faster than FedSubAvg with $K = 20$.

6 Conclusion

In this work, we studied federated submodel optimization over non-i.i.d. data with feature heat dispersion. We proposed FedSubAvg, which ensures the expectation of the global update of each model parameter is equal to the average of the local updates of the clients who involve it. We also proved that FedSubAvg works as a preconditioner to improve collaborative training and thoroughly analyzed the convergence. Empirical studies demonstrated the remarkable superiority of FedSubAvg over FedAvg and its variants.
Federated Submodel Averaging

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*, pages 1273–1282, 2017.

[2] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, pages 583–598. USENIX Association, 2014.

[3] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 3049–3058. PMLR, 2018.

[4] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. D^2: Decentralized training over decentralized data. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 4855–4863. PMLR, 2018.

[5] Hao Yu and Rong Jin. On the computation and communication complexity of parallel SGD with dynamic batch sizes for stochastic non-convex optimization. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7174–7183. PMLR, 2019.

[6] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. First analysis of local GD on heterogeneous data. *CoRR*, abs/1909.04715, 2019.

[7] Sebastian U. Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication. *CoRR*, abs/1909.05350, 2019.

[8] Sebastian U. Stich. Local SGD converges fast and communicates little. In *Proceedings of ICLR*, 2019.

[9] Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. *CoRR*, abs/2010.01243, 2020.

[10] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of AAAI*, pages 5693–5700, 2019.

[11] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *CoRR*, abs/2010.01243, 2020.

[12] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *ICLR*. OpenReview.net, 2020.

[13] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for federated learning. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 2020.

[14] Dmitrii Avdiukhin and Shiva Prasad Kasiviswanathan. Federated learning under arbitrary communication patterns. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 425–435. PMLR, 2021.

[15] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 3407–3416. PMLR, 2021.

[16] Xinran Gu, Kaixuan Huang, Jingzhao Zhang, and Longbo Huang. Fast federated learning in the presence of arbitrary device unavailability. *CoRR*, abs/2106.04159, 2021.

[17] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys*. mlsys.org, 2020.

[18] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: stochastic controlled averaging for federated learning. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 2020.

[19] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. Billion-scale federated learning on mobile clients: a submodel design with tunable privacy. In *MobiCom*, pages 31:1–31:14. ACM, 2020.

[20] Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *NIPS*, pages 1504–1512, 2015.

[21] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum SGD for distributed non-convex optimization. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7184–7193. PMLR, 2019.
Federated Submodel Averaging

[22] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2016.

[23] Tapan Sahni, Chinmay Chandak, Naveen Reddy Chedeti, and Manish Singh. Efficient twitter sentiment classification using subjective distant supervision. In *COMSNETS*, pages 548–553. IEEE, 2017.

[24] Guorui Zhou, Xiaojiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *KDD*, pages 1059–1068. ACM, 2018.
A Proof of Theorem 4.2

A.1 Proof Sketch

We first introduce the following lemma:

**Lemma A.1.** For matrices $A_1, A_2, B_1, B_2 \in \mathbb{M}_n$, if $A_1 \preceq B_1$ and $A_2 \preceq B_2$, then we have $A_1 + A_2 \preceq B_1 + B_2$.

By Lemma A.1 when $X$ is in a locally convex area, for each client’s Hessian $H_i$, we have
\[
E[H_i] \geq (1 - \alpha)\lambda_1 I + \alpha(-\lambda_2)I_i = (\lambda_1 - \alpha(\lambda_1 + \lambda_2))I_i,
\]
where $I_i$ is an identity matrix with the same shape as the Hessian of $H_i$.

By Lemma A.1 we have
\[
E[H] = \frac{1}{N} \sum_{i=1}^{N} E[H_i] \geq \frac{1}{N} \sum_{i=1}^{N} (\lambda_1 - \alpha(\lambda_1 + \lambda_2))I_i = (\lambda_1 - \alpha(\lambda_1 + \lambda_2)) = M_1,
\]
where (a) follows from the align operation when doing summation over local Hessians.

Similarly, we have
\[
H = \frac{1}{N} \sum_{i=1}^{N} H_i \leq \frac{1}{N} \sum_{i=1}^{N} \lambda_2I_i = \frac{\lambda_2}{N} \begin{bmatrix} n_1 & 0 & \cdots & 0 \\ 0 & n_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & n_M \end{bmatrix} = M_2.
\]

Thus, $E[H] \preceq M_2$. We next introduce Lemma A.2 about eigenvalue.

**Lemma A.2.** For matrices $A, B \in \mathbb{M}_n$, if $A \preceq B$, then we have $\lambda_{\min}(A) \leq \lambda_{\min}(B)$ and $\lambda_{\max}(A) \leq \lambda_{\max}(B)$.

By Lemma A.2 we have $\lambda_{\max}(E[H]) \geq \lambda_{\max}(M_1) = n_{\max}(\lambda_1 - \alpha(\lambda_1 + \lambda_2))/N$, and $\lambda_{\min}(E[H]) \leq \lambda_{\min}(M_2) = n_{\min}\lambda_2/N$. Therefore, we have the lower bound of the condition number of $E[H]$:
\[
\kappa(E[H]) \geq \frac{n_{\max}(\lambda_1 - \alpha(\lambda_1 + \lambda_2))/N}{n_{\min}\lambda_2} = \frac{n_{\max}}{n_{\min}} \frac{(\lambda_1 - \alpha(\lambda_1 + \lambda_2))}{n_{\min}} = \Theta \left( \frac{n_{\max}}{n_{\min}} \right).
\]

A.2 Proof of Lemmas

**Proof of Lemma A.1.** If $A_1 \preceq B_1$ and $A_2 \preceq B_2$, for any $x \in \mathbb{R}^n$, we have
\[
x^\top A_1 x \leq x^\top B_1 x, \quad x^\top A_2 x \leq x^\top B_2 x.
\]

Thus, $\forall x \in \mathbb{R}^n$, we have
\[
x^\top (A_1 + A_2) x \leq x^\top (B_1 + B_2) x,
\]
and we further have $(A_1 + A_2) \preceq (B_1 + B_2)$.

**Proof of Lemma A.2.** For any matrix $P \in \mathbb{M}_n$, we have
\[
\lambda_{\max}(P) = \max_{x \in \mathbb{R}^n, x \neq 0} \left\{ \frac{x^\top Px}{x^\top x} \right\}, \quad \lambda_{\min}(P) = \min_{x \in \mathbb{R}^n, x \neq 0} \left\{ \frac{x^\top Px}{x^\top x} \right\}.
\]

For two matrices $A, B$ with $A \preceq B$, we have
\[
\frac{x^\top Ax}{x^\top x} \leq \frac{x^\top Bx}{x^\top x},
\]
for any vector $x \in \mathbb{R}^n (x \neq 0)$. Therefore, we have
\[
\max_{x \in \mathbb{R}^n, x \neq 0} \left\{ \frac{x^\top Ax}{x^\top x} \right\} \leq \max_{x \in \mathbb{R}^n, x \neq 0} \left\{ \frac{x^\top Bx}{x^\top x} \right\}, \quad \min_{x \in \mathbb{R}^n, x \neq 0} \left\{ \frac{x^\top Ax}{x^\top x} \right\} \leq \min_{x \in \mathbb{R}^n, x \neq 0} \left\{ \frac{x^\top Bx}{x^\top x} \right\}.
\]

So we have $\lambda_{\max}(A) \leq \lambda_{\max}(B)$ and $\lambda_{\min}(A) \leq \lambda_{\min}(B)$.
B Proof of Theorem 4.3

The Hessian of \( f \) is \( H \triangleq D^\frac{1}{2} HD^\frac{1}{2} \). Thus, we have \( E[H] = D^\frac{1}{2} E[H] D^\frac{1}{2} \). By equations 6 and 8, we have \( M_1 \preceq E[H] \preceq M_2 \). Rearranging the terms yields \( E[H] = M_1 \succeq 0 \) and \( M_2 - E[H] \succeq 0 \). Therefore, for any vector \( x \in \mathbb{R}^M \), we have

\[
x^\top (E[H] - D^\frac{1}{2} M_1 D^\frac{1}{2}) x = x^\top D^\frac{1}{2} (E[H] - M_1) D^\frac{1}{2} x = \left( D^\frac{1}{2} x \right)^\top (E[H] - M_1) (D^\frac{1}{2} x) \geq 0,
\]

\[
x^\top (D^\frac{1}{2} M_2 D^\frac{1}{2} - E[H]) x = x^\top D^\frac{1}{2} (M_2 - E[H]) D^\frac{1}{2} x = \left( D^\frac{1}{2} x \right)^\top (M_2 - E[H]) (D^\frac{1}{2} x) \geq 0.
\]

(15)

So we have

\[
D^\frac{1}{2} M_1 D^\frac{1}{2} \preceq E[H] \preceq D^\frac{1}{2} M_2 D^\frac{1}{2}.
\]

(16)

By Lemma A.2, we have

\[
\lambda_{\min}(E[H]) \geq \lambda_{\min}(D^\frac{1}{2} M_1 D^\frac{1}{2}) = (\lambda_1 - \alpha(\lambda_1 + \lambda_2)),
\]

\[
\lambda_{\max}(E[H]) \leq \lambda_{\max}(D^\frac{1}{2} M_2 D^\frac{1}{2}) = \lambda_2.
\]

(17)

Thus, the condition number of \( E[H] \) satisfies \( \kappa(E[H]) \leq \lambda_2/(\lambda_1 - \alpha(\lambda_1 + \lambda_2)) = \Theta(1) \).

C Proof of Theorem 4.4

We consider the minimum eigenvalue of \( H \). Let \( x_0 \in \mathbb{R}^M (x_0 \neq 0) \) such that \( \lambda_{\min}(H) = x_0^\top H x_0 / x_0^\top x_0 \). Let \( x_1 = D^{-\frac{1}{2}} x_0 \). Then, we have

\[
\lambda_{\min}(H) \leq \frac{x_1^\top H x_1}{x_1^\top x_1} = \frac{x_1^\top D^\frac{1}{2} HD^\frac{1}{2} x_1}{x_1^\top x_1} = \frac{x_0^\top H x_0}{x_0^\top D^{-1} x_0} \leq \frac{x_0^\top H x_0}{n_{\max} x_0^\top x_0} = \frac{N}{n_{\max}} \lambda_{\min}(H),
\]

(18)

where (a) follows from \( x_0^\top H x_0 < 0 \) and \( D^{-1} = \text{diag}\{n_1/N, n_2/N, \ldots, n_M/N\} \).

D Proof of Theorem 4.5

Under Assumption 4.1 and equation 8, we have \( H \preceq M_2 \). By Lemma A.2, we further have

\[
\lambda_{\max}(H) \leq \lambda_{\max}(M_2) = \frac{n_{\max} \lambda_2}{N}.
\]

(19)

Similar to equation 15, we have

\[
\lambda_{\max}(H) = \lambda_{\max}(D^\frac{1}{2} H D^\frac{1}{2}) \leq \lambda_{\max}(D^\frac{1}{2} M_2 D^\frac{1}{2}) = \lambda_2.
\]

(20)

Thus, we have the upper bound of the eigenvalues of \( H \) and \( \bar{H} \).

E Proof of Theorem 4.9

E.1 Additional Notations

Let \( X_i \) denote \( X_{S(i)} \) and \( U \) denote \( \frac{1}{N} \cdot D = \text{diag}\{1/n_1, 1/n_2, \ldots, 1/n_M\} \). We have

\[
X' = U \cdot \sum_{i=1}^N x_i
\]

(21)

We then assume that FedSubAvg always activates all the clients at the beginning of each communication round and then uses the parameters maintained by a few selected clients to generate the next-round parameter. It is clear that this update scheme is equivalent to the original. Then, the update of FedSubAvg can be summarized as: for all \( i \in [N] \),

\[
y_i^{t+1} = x_i^t - \gamma g_i^t,
\]

(22)
We next bound $A_{t+1} = \begin{cases} y_i^{t+1} - x_i^t + y_i^t \\ \frac{N}{K} \sum_{j \in C_{t+1}} (y_j^{t+1} - x_j^{t+1}) \end{cases}$ if $t$ is not a multiple of $I$, \( y_i^{t+1} - x_i^t + y_i^t \) if $t$ is a multiple of $I$, \( y_i^{t+1} - x_i^t + y_i^t \) if $t$ is a multiple of $I$.

where $g_i^t = \nabla F(x_i^t, \xi_i^t)$ is the local gradient of client $i$ at iteration $t$, and $U_i$ denotes the local part of $U$ for client $i$. Clearly, in this update scheme, when $t$ is a communication iteration, we have \[ E_{C_{t+1}} [X^{t+1}] = E_{C_{t+1}} \left[ X^{t+1-1} + \frac{N}{K} \sum_{j \in C_{t+1}} (y_j^{t+1} - x_j^{t+1}) \right] = U \cdot \sum_{i=1}^{N} y_i^{t+1} \triangleq Y^{t+1} \]

Additionally, $X^{t+1} = Y^{t+1}$ also holds when $t$ is not a communication iteration. Therefore, $Y^{t+1} = E[X^{t+1}]$.

### E.2 Key Lemmas

**Lemma E.1.**

\[ E \left[ \frac{1}{N} \sum_{i=1}^{N} || x_i^t - X_i^t ||^2 \right] \leq 4 \gamma^2 I^2 G^2. \]

**Proof.** FedSubAvg requires communication every $I$ iterations. Therefore, for any $t \geq 0$, there exists a $t_0 \leq t$, such that $t - t_0 \leq I - 1$ and $x_i^{t_0} = X_i^{t_0}$ for all $i \in N$. Then, we have

\[
\begin{align*}
E \left[ \frac{1}{N} \sum_{i=1}^{N} || x_i^t - X_i^t ||^2 \right] \\
= & E \left[ \frac{1}{N} \sum_{i=1}^{N} \left( || x_i^t - X_i^{t_0} || - || X_i^{t_0} - X_i^t || \right) \right] \\
\leq & E \left[ \frac{1}{N} \sum_{i=1}^{N} \left[ \sum_{t=t_0}^{t-1} \gamma g_i^t \right] - \sum_{t=t_0}^{t-1} \sum_{i=1}^{N} \gamma U_i \sum_{i=1}^{N} g_i^t \right] \\\n\leq & 2E \left[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=t_0}^{t-1} || g_i^t ||^2 \right] + 2E \left[ \frac{1}{N} \sum_{i=1}^{N} \sum_{t=t_0}^{t-1} \gamma U_i \sum_{i=1}^{N} g_i^t \right].
\end{align*}
\]

We first focus on bounding $A_1$:

\[ A_1 \leq \frac{1}{N} \sum_{i=1}^{N} \sum_{t=t_0}^{t-1} \gamma^2 g_i^t \leq \frac{\gamma^2 (I - 1)^2}{N} \sum_{i=1}^{N} g_i^t \leq \gamma^2 G^2 (I - 1)^2. \]

We next bound $A_2$:

\[ A_2 \leq \frac{\gamma^2 (I - 1)}{N} \sum_{i=1}^{N} \sum_{t=t_0}^{t-1} \left( \sum_{i=1}^{N} g_i^t \right)^2 \]

where $A_3$ can be bounded as follows:

\[ A_3 \leq \sum_{m=1}^{M} \sum_{m \in S(i)} \left( \sum_{m \in S(i)} \frac{g_i^t}{n_{m}} \right)^2 \leq \sum_{m=1}^{M} \frac{1}{n_{m}} \left( \sum_{m \in S(i)} g_i^t \right)^2 \leq \sum_{m=1}^{M} \left( g_i^t \right)^2 \leq NG^2. \]

Substituting equations \(26\), \(27\) and \(28\) into \(25\) yields

\[ E \left[ \frac{1}{N} \sum_{i=1}^{N} || x_i^t - X_i^t ||^2 \right] \leq 4 \gamma^2 G^2 (I - 1)^2. \]
E.3 Completing the Proof of Theorem 4.9

Proof. For each client $i$, by the $L$-smoothness of $f_i(\cdot)$, we have

$$
\mathbb{E} [ f_i (X_i^{t+1}) ] \leq \mathbb{E} [ f_i (X_i^t) ] + \mathbb{E} \left[ \langle X_i^{t+1} - X_i^t, \nabla f_i (X_i^t) \rangle \right] + \frac{L}{2} \mathbb{E} \left[ \| X_i^{t+1} - X_i^t \|^2 \right].
$$

(30)

We first focus on bounding $C_i^1$. 

$$
C_i^1 = \mathbb{E} \left[ \langle Y_i^{t+1} - X_i^t, \nabla f_i (X_i^t) \rangle \right] + \mathbb{E} \left[ \langle X_i^{t+1} - Y_i^{t+1}, \nabla f_i (X_i^t) \rangle \right]
= \mathbb{E} \left[ -\gamma U_i \sum_{i=1}^N g_i^t, \nabla f_i (X_i^t) \right] = \mathbb{E} \left[ -\gamma U \sum_{i=1}^N g_i^t, \nabla f_i (X_i^t) \right]
$$

(31)

Substituting $C_i^1$ over $i \in [N]$ yields

$$
\mathbb{E} \left[ \frac{1}{N} \sum_{i=1}^N C_i^1 \right] = -\gamma \mathbb{E} \left[ \left\langle U \sum_{i=1}^N g_i^t, \frac{1}{N} \sum_{i=1}^N \nabla f_i (X_i^t) \right\rangle \right]
= -\frac{\gamma}{N} \mathbb{E} \left[ \left\langle U \sum_{i=1}^N \nabla f_i (x_i^t), \sum_{i=1}^N \nabla f_i (X_i^t) \right\rangle \right]
\overset{(a)}{=} -\frac{\gamma}{N} \mathbb{E} \left[ \left\langle V \sum_{i=1}^N \nabla f_i (x_i^t), V \sum_{i=1}^N \nabla f_i (X_i^t) \right\rangle \right]
= -\frac{\gamma}{2N} \mathbb{E} \left[ \left( \sum_{i=1}^N \nabla f_i (x_i^t) \right)^\top U \left( \sum_{i=1}^N \nabla f_i (X_i^t) \right) \right]
- \frac{\gamma}{2N} \mathbb{E} \left[ \left( \sum_{i=1}^N \nabla f_i (x_i^t) \right)^\top U \left( \sum_{i=1}^N \nabla f_i (X_i^t) \right) \right]
+ \frac{\gamma}{2N} \mathbb{E} \left[ \left( \sum_{i=1}^N \nabla f_i (x_i^t) \right)^\top U \left( \sum_{i=1}^N \nabla f_i (X_i^t) \right) \right]
$$

(32)

where $V = \text{diag}(1/\sqrt{n_1}, 1/\sqrt{n_2}, \ldots, 1/\sqrt{n_M})$ and $U = V^2$ in (a), while $D$ can be bounded as follows:

$$
D \leq \frac{1}{n_{\min}} \left\| \sum_{i=1}^N (\nabla f_i (x_i^t) - \nabla f_i (X_i^t)) \right\|^2 \leq \frac{NL^2}{n_{\min}} \sum_{i=1}^N \| x_i^t - X_i^t \|^2 \leq \frac{4N^2\gamma^2G^2L^2(1 - I)^2}{n_{\min}},
$$

(33)

We next consider bounding $C_i^2$:

$$
C_i^2 \leq 2\mathbb{E} \left[ \| Y_i^{t+1} - X_i^t \|^2 \right] + 2\mathbb{E} \left[ \| X_i^{t+1} - Y_i^{t+1} \|^2 \right].
$$

(34)
Federated Submodel Averaging

Since \( Y_{t+1}^i = X_i - \gamma U_i \sum_{t=1}^N g_i^t \), we have

\[
\mathbb{E} \left[ \frac{1}{N} \sum_{i=1}^N E_i^t \right] = \mathbb{E} \left[ \frac{1}{N} \sum_{i=1}^N \gamma^2 \|U_i \sum_{i=1}^N g_i^t\|^2 \right]
\]

\[
= \frac{\gamma^2}{N} \mathbb{E} \left[ \left( \sum_{i=1}^N g_i^t \right)^T U \left( \sum_{i=1}^N g_i^t \right) \right]
\]

\[
= \frac{\gamma^2}{N} \mathbb{E} \left[ \left( \sum_{i=1}^N (g_i^t - \nabla f_i(x_i^t)) \right)^T U \left( \sum_{i=1}^N (g_i^t - \nabla f_i(x_i^t)) \right) \right]
\]

\[+ \frac{2\gamma}{N} \mathbb{E} \left[ \left( \sum_{i=1}^N (g_i^t - \nabla f_i(x_i^t)) \right)^T U \left( \sum_{i=1}^N (g_i^t - \nabla f_i(x_i^t)) \right) \right]
\]

\[
\leq \frac{\gamma^2}{n_{\min} N} \mathbb{E} \left[ \| \sum_{i=1}^N (g_i^t - \nabla f_i(x_i^t)) \|^2 \right]
\]

\[+ \frac{\gamma^2}{N} \mathbb{E} \left[ \left( \sum_{i=1}^N (g_i^t - \nabla f_i(x_i^t)) \right)^T U \left( \sum_{i=1}^N (g_i^t - \nabla f_i(x_i^t)) \right) \right]
\]

\[
\leq \frac{\gamma^2 \sigma^2}{n_{\min}} + \frac{\gamma^2}{N} \mathbb{E} \left[ \left( \sum_{i=1}^N \nabla f_i(x_i^t) \right)^T U \left( \sum_{i=1}^N \nabla f_i(x_i^t) \right) \right].
\]

If \( t \) is not a communication iteration, we have \( E_2^t = 0 \); otherwise, we have

\[
\mathbb{E}_{C_{t+1}}[E_2^t] = \mathbb{E}_{C_{t+1}} \left[ \left\| X_i^{t+1} + \frac{N}{K} U_i \sum_{j \in C_{t+1}} (y_j^{t+1} - x_j^t) - U_i \sum_{j=1}^N y_j^{t+1} \right\|^2 \right]
\]

\[
= \mathbb{E}_{C_{t+1}} \left[ \left\| \frac{N}{K} U_i \sum_{j \in C_{t+1}} (y_j^{t+1} - x_j^t) - \left( U_i \sum_{j=1}^N y_j^{t+1} - X_i^t \right) \right\|^2 \right]
\]

\[
\leq \frac{N^2}{K^2} \mathbb{E}_{C_{t+1}} \left[ \sum_{j \in C_{t+1}} (y_j^{t+1} - x_j^t) \circ (y_j^{t+1} - x_j^t) \right]
\]

\[
= \frac{N}{K} \mathbb{E}_{C_{t+1}} \left[ \sum_{j=1}^N (y_j^{t+1} - x_j^t) \circ (y_j^{t+1} - x_j^t) \right]
\]

\[
\leq \frac{N I}{K} \mathbb{E}_{C_{t+1}} \left[ \sum_{j=1}^N \sum_{t=0}^{t+1} \gamma^2 g_j^t \circ g_j^t \right]
\]

\[
\leq \frac{N I \gamma^2}{K} \mathbb{E}_{C_{t+1}} \left[ \sum_{j=1}^N \sum_{t=-l}^{t+1} g_j^t \circ g_j^t \right],
\]

where \( u_i \) denotes the local part of \((1/n_i^2, 1/n_i^2, \ldots, 1/n_i^2)\) for client \( i \), \( \circ \) denotes the element-wise multiplication, and (a) follows from that \( \mathbb{E}[\| z - \mathbb{E}[z] \|^2] \leq \mathbb{E}[\| z \|^2] \) holds for any random vector \( z \).
Federated Submodel Averaging

Summing over \( i = [N] \), we have

\[
\mathbb{E}\left[ \frac{1}{N} \sum_{i=1}^{N} E_2^i \right] \leq \frac{1}{N} \sum_{i=1}^{N} \frac{NI\gamma^2}{K} u_i \sum_{\tau=t-I}^{t} g_{j^*}^\tau \circ g_{j^*}^\tau
\]

\[
= \frac{I\gamma^2}{K} \sum_{i=1}^{N} u_i \sum_{\tau=t-I}^{t} g_{j^*}^\tau \circ g_{j^*}^\tau (37)
\]

Taking an average of equation (30) over \( i \in [N] \), substituting equations (31), (33), (34), (35), (37) into (30), and rearranging the terms yields

\[
\mathbb{E}\left[ \frac{1}{N} \left( \sum_{i=1}^{N} \nabla f_i(X^t_i) \right)^\top U \left( \sum_{i=1}^{N} \nabla f_i(X^t_i) \right) \right] \leq \frac{2}{\gamma} \left[ \mathbb{E}[f(X^t)] - \mathbb{E}[f(X^{t+1})] \right] + \frac{2\gamma L\sigma^2}{\nu_{min}} + \frac{4N\gamma^2G^2L^2(I-1)^2}{\nu_{min}^2} + \frac{2\gamma NI^2G^2L}{\nu_{min}K}. (38)
\]

Summing over \( t \in \{1, 2, \cdots, T\} \) and dividing both sides by \( T \) yields

\[
\mathbb{E}\left[ \frac{1}{T} \sum_{i=1}^{T} \frac{1}{N} \left( \sum_{i=1}^{N} \nabla f_i(X^t_i) \right)^\top U \left( \sum_{i=1}^{N} \nabla f_i(X^t_i) \right) \right] \leq \frac{2}{\gamma T} \left[ f(X^t) - f(X^{t+1}) \right] + \frac{2\gamma L\sigma^2}{\nu_{min}} + \frac{4N\gamma^2G^2L^2}{\nu_{min}^2} + \frac{2\gamma NI^2G^2L}{\nu_{min}K}. (39)
\]

Therefore, we have

\[
\mathbb{E}\left[ \frac{1}{T} \sum_{i=1}^{T} \nabla f(X^t_i)^\top D \nabla f(X^t_i) \right] = \mathbb{E}\left[ \frac{1}{T} \sum_{i=1}^{T} \frac{1}{N} \left( \sum_{i=1}^{N} \nabla f_i(X^t_i) \right)^\top U \left( \sum_{i=1}^{N} \nabla f_i(X^t_i) \right) \right] \leq \frac{2}{\gamma T} \left[ f(X^t) - f(X^{t+1}) \right] + \frac{2\gamma L\sigma^2}{\nu_{min}} + \frac{4N\gamma^2G^2L^2}{\nu_{min}^2} + \frac{2\gamma NI^2G^2L}{\nu_{min}K}. (40)
\]

\[\square\]

F Experimental Details

F.1 Feature Heat Distributions on Datasets

Figure 5 shows the feature heat distributions of the head features on four datasets in NLP or RS. We can observe that the feature heat (i.e., the number of feature-involved clients) varies widely among different features.

F.2 The Approximation of Scaffold in CTR Prediction

In CTR prediction, we make an approximation to Scaffold since a resource-constrained client cannot keep the control variate. Therefore, we perform the controlled update on the cloud server in each round. Let \( c \) denote the globally controlled variate, and let \( c_i \) denote client \( i \)’s local control variate. At the end of each round in Scaffold, we have:

\[
c_{new} \leftarrow c_{old} + \frac{1}{N} \sum_{i \in S_c} (c_{i,new} - c_{i,old}),
\]

\[
c_{odd} \approx \frac{1}{N} \sum_{i=1}^{N} c_{i,odd},
\]

\[
c^\text{new} \leftarrow c^\text{old} + \frac{1}{N} \sum_{i \in S_c} (c_{i,new} - c_{i,old}),
\]

\[
(41)
\]
Federated Submodel Averaging

Figure 5: Feature heat distributions on four datasets (only the head features are shown). The x-axis represents feature heat, i.e., the number of clients involving a movie/word/item, and the y-axis represents the number of movies/words/items under a certain heat.

where $S_c$ is the selected clients at the round. Taking expectation with respect to the selected clients, we have

$$E[c_{new}^{new}] = c_{old} + \frac{|S_c|}{N} \sum_{i=1}^{N} (c_{new}^i - c_{old}^i) \approx \frac{N - |S_c|}{N} c_{old} + \frac{|S_c|}{N} \frac{1}{N} \sum_{i=1}^{N} c_{new}^i \approx \frac{N - |S_c|}{N} c_{old} + \frac{|S_c|}{N} \cdot E \left[ \frac{1}{|S_c|} \sum_{i \in S_c} c_{new}^i \right].$$ (42)

In addition, the global update $\Delta X \approx -\eta Ic$ and client $i$'s local update $\Delta x_i \approx -\eta Ic_i$. By equation 42 we can approximate the global update by

$$\Delta X^{new} \approx \frac{N - |S_c|}{N} \Delta X^{old} + \frac{|S_c|}{N} \left( \frac{1}{|S_c|} \sum_{i \in S_c} \Delta x_i \right).$$ (43)

Therefore, we run Scaffold approximately by weighted averaging the original global update and the aggregated local updates to get the new global update every communication round.

F.3 Hyperparameters

For the tasks of rating classification and sentiment analysis, we set the local batch size to 5 and set the local iteration number to 10 in all FL algorithms. We set the batch size to 250 and set the iteration number in each round to 10 in CentralSGD. For the CTR prediction on Amazon, we set the batch size to 4 and set the local iteration number to 10 in all FL algorithms. We set the batch size to 400 and set the iteration number in each round to 10 in CentralSGD. For the CTR prediction on the Alibaba dataset, we set the batch size to 32 and set the local iteration number to 10 in all FL
algorithms. We set the batch size to 3,200 and set the iteration number in each round to 10 in CentralSGD. We tune the learning rate for each algorithm, and the learning rates are recorded in Table 4.

Table 4: Learning rate for each experiment.

|                   | CentralSGD | FedAvg | FedProx | Scaffold | FedSubAvg |
|-------------------|------------|--------|---------|----------|-----------|
| MovieLens         | 0.1        | 0.1    | 0.1     | 0.1      | 0.1       |
| Sent140           | 0.1        | 0.1    | 0.1     | 0.1      | 0.1       |
| Amazon            | 0.05       | 0.1    | 0.1     | 0.1      | 0.05      |
| Alibaba           | 1          | 1      | 1       | 1        | 0.3       |

F.4 Additional Results

We show the test accuracies (ACCs) or AUCs for each experiment. Figure 6 compares the test ACCs or AUCs of FedSubAvg and baselines under default settings. Figure 7 compares the test ACCs or AUCs of FedSubAvg with different numbers of participating client per round $K$. All the results from test ACC or AUC are consistent with the results from the train loss.
Figure 7: Test ACCs or test AUCs of FedSubAvg on different datasets with the varying number of selected clients per round $K$.

F.5 Supplementary Notes for the Experiments

In our experiments, all FL algorithms are extended to the weighted case. In particular, the correction coefficient $N/n_m$ in FedSubAvg is extended to $\sum_{i=1}^{N} w_i / \sum_{j|m \in S(j)} w_j$, where $w_i$ is the size of client $i$'s local training data. For the rating classification, the MovieLens dataset is available from https://grouplens.org/datasets/movielens/1m/. We randomly select 20% of the samples as the test dataset and leave the remaining 80% as the training set, and further randomly choose 10,000 samples from the train set to evaluate train losses. For the sentiment analysis, the Sentiment140 dataset is available from http://help.sentiment140.com/for-students, and we randomly select 20% of the samples as the test dataset and leave the remaining 80% as the training set. For the CTR prediction, the Amazon dataset is available from http://jmcauley.ucsd.edu/data/amazon/ and we partition the dataset based on the timestamp. In addition, experiments are conducted on machines with one NVIDIA GeForce RTX 2080Ti GPU.