# FASTER FEDERATED LEARNING WITH DECAYING NUMBER OF LOCAL SGD STEPS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In Federated Learning (FL), client devices collaboratively train a model without sharing the private data present on the devices. Federated Stochastic Gradient Descent (FedSGD) is a recent generalisation of the popular Federated Averaging algorithm. Recent works show that when client data is distributed heterogeneously, the loss function minimised by FedSGD differs from the *true* loss that would be minimised by centralised training. Previous works propose decaying the client learning rate, $\gamma$, to allow FedSGD to minimise the true loss. We propose instead decaying the number of local SGD steps, $K$, that clients perform during training rounds to allow minimisation of the true loss. Decaying $K$ has the added benefit of reducing the total computation that clients perform during FedSGD. Real-world applications of FL use large numbers of low-powered smartphone or Internet-of-Things clients, so reduction of computation would provide significant savings in terms of energy and time. In this work, we prove for quadratic objectives that annealing $K$ allows FedSGD to approach the true minimiser. We then perform thorough experimentation on three benchmark FL datasets to show that decaying $K$ can achieve the same generalisation performance as decaying $\gamma$, but with up to $3.8\times$ less total steps of SGD performed by clients.

## 1 INTRODUCTION

Federated Learning (FL) is a recent distributed machine learning paradigm that aims to collaboratively train a model using data owned by numerous clients, without these clients sharing their potentially sensitive data. Practical applications of FL range from 'cross-device' scenarios, where a huge number of typically unreliable clients with small quantities of data per client participate, to 'cross-silo' scenarios with smaller numbers of reliable clients, each possessing larger quantities of data (Kairouz et al., 2019). Cross-device tasks include mobile-keyboard next-word prediction (Hard et al., 2018), (Ramaswamy et al., 2019), (Wang et al., 2019), digital-assistant keyword-spotting (Leroy et al., 2019), or content-recommendation (Ammad-ud-din et al., 2019). Cross-silo tasks can involve medical institutions processing healthcare data (Sheller et al., 2020), (Huang et al., 2019).

The challenges faced in developing practical and useful FL systems (in particular for cross-device FL) include: the heterogeneous distribution of data across client devices; the communication costs associated with the training process; and unreliable client devices that can arbitrarily join and leave the FL process. McMahan et al. (2017) published an early FL algorithm, Federated Averaging (FedAvg), which has been the subject of much study. FedAvg works in an iterative fashion similar to distributed SGD, but performs multiple steps of SGD on clients between aggregations. The communication overhead of uploading and downloading the FL model is therefore significantly reduced over distributed SGD, at the cost of performing more total computation on clients during the training process. Myriad extensions to FedAvg and iterative FL have been proposed including compression of the communicated model gradients (Sun et al., 2020), multi-task FL (Smith et al., 2017), FL with decision trees (Li et al., 2020a), iterative FL where client neurons are aligned between steps (Wang et al., 2020), and more. Comprehensive surveys of FL are provided by Kairouz et al. (2019), Li et al. (2020b), Lo et al. (2020).

Fewer works have been published on analysing the convergence properties of FL and FedAvg, especially in the heterogeneous setting. Woodworth et al. (2020) proved that, for quadratic objectives, Local-SGD methods (of which FedAvg is a special case) converge at least as fast as dis-

tributed SGD, but that Local-SGD does not dominate distributed SGD for convex problems. Li et al. (2020c) proved a convergence rate of $\mathcal{O}(\frac{1}{T})$ for FedAvg on strongly convex problems, with non-Independently, Identically Distributed (non-iid) client data, and without all clients participating in each round. They also highlighted the need for decaying the client learning rate for FedAvg to converge. With the aim of improving the convergence speed of FedAvg, Reddi et al. (2020) generalised FedAvg to update the global model by treating client model updates as a *psuedo-gradient*. The global model is then updated with an SGD-like step incorporating a server learning rate, $\eta$. This generalisation is named FedSGD, of which FedAvg is a special case (when $\eta = 1$). The authors then extended FedSGD use adaptive optimization schemes such as Adam (Kingma & Ba, 2014) when updating the global model (FedAdam). The authors provide convergence rates for FedAdam in the convex setting with full device participation. Broadly, they showed that a larger number of local iterations, $K$, leads to fewer rounds of communication for iid clients. Charles & Konecný (2020) generalised several distributed learning algorithms (termed *Local-Update methods*) and analysed their convergence in the quadratic setting. They showed that, by performing multiple steps of SGD between aggregations, Local-Update methods (of which FedAvg is a special case) can provide faster convergence, but converge to a *surrogate* minimiser that may differ from the *true* minimizer that would be attained by training on clients' pooled data. The authors prove that reducing the client learning rate during training is necessary to reach the true minimiser.

Previous works show that decaying the client learning rate $\gamma$ allows FedSGD to optimise the true objective. In this paper, we propose instead decaying the number of local steps $K$ that clients perform between aggregation rounds of FedSGD. Decaying $K$ also allows FedSGD to reach the true minimiser, but with the benefit of decreased total computation on clients. Our contributions are: (1) convergence analysis of FedSGD with decaying $K$ for quadratic objectives; (2) demonstration of this convergence on a simple quadratic problem; (3) thorough experimentation that investigate the impact of decaying $\gamma$ or $K$ on the generalisation performance of FedSGD using 3 benchmark FL datasets. The experiments show that decaying $K$ can reach the same generalisation ability as FedSGD with decaying $\gamma$, whilst performing up to $3.8\times$ less total SGD steps.

## 2 FedSGD with Decaying Number of Local SGD Steps

### 2.1 The FedSGD Algorithm

Reddi et al. (2020) first generalised FedAvg (McMahan et al., 2017) to FedSGD by treating the updates sent from workers as a *psuedo-gradient*. This psuedo-gradient then is used to update the aggregate model in an SGD-like step. FedSGD allows tuning of the server learning rate $\eta$, which (Reddi et al., 2020) show can improve convergence over standard FedAvg.

**Algorithm 1** FedSGD: ServerUpdate

1: Initialise global model, $x_1$
2: **while** termination criteria not met **do**
3: Select round clients, $\mathcal{I}_t$
4: **for each** client $i \in \mathcal{I}_t$ in parallel **do**
5: $q_t^i \leftarrow \text{ClientUpdate}^i(x, \gamma, K)$
6: **end for**
7: $q_t \leftarrow \frac{1}{\sum_{i \in \mathcal{I}_t} |\mathcal{D}_i|)} \sum_{i \in \mathcal{I}_t} |\mathcal{D}_i| q_t^i$
8: $x_{t+1} \leftarrow x_t - \eta_t q_t$
9: **end while**

**Algorithm 2** FedSGD: ClientUpdate

1: function $\mathbf{ClientUpdate}^i(x, \gamma, K)$
2: $x_1 \leftarrow x$
3: **for** $k = 1..K$ **do**
4: sample set $S_k$ from $\mathcal{D}_i$ of size $B$
5: $g_k \leftarrow (1/B) \sum_{z \in S_k} \nabla f(x_k; z)$
6: $x_{k+1} \leftarrow x_k - \gamma g_k$
7: **end for**
8: **return** $x - x_k$

Algorithms 1 and 2 outline the operation of FedSGD. The server initialises a global model, $x_1$ (Algorithm 1, line 1), and begins the iterative training phase. Each round, a subset of clients $\mathcal{I}_t$ are selected for participation (Algorithm 1, line 3). Each client downloads the global model and updates it locally by performing SGD with client learning-rate $\gamma$ on their local dataset (Algorithm 2, lines 4-6). The difference between the clients' local models and the round's global model are sent to the server (Algorithm 2, line 8). The sever averages these updates, weighted by the number of samples on each client (Algorithm 1, line 7), and uses this average update as the psuedo-gradient in an SGD-

like step to update the global model with server learning rate $\eta$ (Algorithm 1, line 8). Another round then starts with the new global model.

### 2.2 CONVERGENCE ANALYSIS OF FEDSGD WITH DECAYING K

Charles & Konecný (2020) generalised FedSGD further into *LocalUpdate*, which also encompasses similar distributed-training algorithms such as MAML (Finn et al., 2017) and distributed SGD. They showed for quadratic objectives that when $K > 1$ and $\gamma > 0$, the loss function minimised by LocalUpdate differs from the loss function that would have been minimised by centralised training or by distributed SGD. The authors showed that by decaying $\gamma$ during training, the original loss function can be recovered. We conduct an analysis to show that decaying $K$ also allows the original loss to be recovered, which gives the extra benefit of less total computation needed during training.

We aim to train a model $x \in \mathbb{R}^d$. We consider a set of clients $\mathcal{I}$, with probability distribution $\mathcal{P}$. Each client $i \in \mathcal{I}$ possesses a distribution of data $\mathcal{D}_i$ on an example space $\mathcal{Z}$. We assume each $z \in \mathcal{Z}$ can be characterised by symmetric matrix $A_z \in \mathbb{R}^{d\times d}$ and vector $c_z \in \mathbb{R}^d$. The loss for a given model $x$ and sample $z$, $f(x; z)$, the expected loss on client $i$, $f_i(x)$, and expected loss over clients $f(x)$ are given by:

$$f(x; z) = \frac{1}{2}\|A_z^{1/2}(x - c_z)\|^2 \tag{1}$$

$$f_i(x) = \mathop{\mathbb{E}}_{z\sim\mathcal{D}_i}[f(x; z)] \tag{2}$$

$$f(x) = \mathop{\mathbb{E}}_{i\sim\mathcal{P}}[f_i(x)]. \tag{3}$$

In previous works such as McMahan et al. (2017), the authors aim to minimise the objective $f(x)$. However, heterogeneous client datasets lead the actual objective minimised by FedSGD to differ from $f(x)$. Charles & Konecný (2020) show that when $K > 1$ and $\gamma > 0$, the objective minimised by FedSGD, $\tilde{f}(x, \gamma, K)$, is actually the expectation over client *surrogate* loss functions $\tilde{f}_i(x, \gamma, K)$:

$$\tilde{f}_i(x, \gamma, K) = \frac{1}{2}\|(Q_i(\gamma, K)A_i)^{1/2}(x - c_i)\| \tag{4}$$

$$\tilde{f}(x, \gamma, K) = \mathop{\mathbb{E}}_{i\sim\mathcal{P}}[\tilde{f}_i(x, \gamma, K)]. \tag{5}$$

The distortion matrix $Q_i$ amplifies the heterogeneity of client data. Charles & Konecný (2020) go on to demonstrate that when the client learning rate $0 < \gamma < L_i^{-1}$ (where $L_i^{-1}$ is the Lipschitz constant of $A_i$), the eigenvalues of matrix $Q_i(\gamma, K)A_i$ are given by:

$$\phi_\lambda(\gamma, K) = \gamma^{-1}(1 - (1 - \gamma\lambda)^K), \tag{6}$$

where $\lambda$ is an eigenvalue of $A_i$. This allows Charles & Konecný (2020) to bound the distance between the minimiser of the surrogate loss function $x^*(\gamma, K)$ and the true minimiser $x^*$ as:

$$\|x^*(\gamma, K) - x^*\| \leq \sigma_c \left(1 + \frac{\sigma_A}{\mu}\right) \frac{LK - \phi_L(\gamma, K)}{\phi_\mu(\gamma, K)}, \tag{7}$$

where $\mathop{\mathrm{Var}}_{i\sim\mathcal{P}}[c_i] \leq \sigma_c^2$, $\mathop{\mathrm{Var}}_{i\sim\mathcal{P}}[A_i] \leq \sigma_A^2$, and with the assumption that $\mu I \preceq A_i \preceq LI, \forall i \in \mathcal{I}$. By inspecting Equation 7, it can be seen that as $\gamma \to 0$, then $\|x^*(\gamma, K) - x^*\| \to 0$. Charles & Konecný (2020) later use this to show that by decaying $\gamma$ during training, the global loss function minimised by FedSGD approaches the true loss function. However, inspection of Equation 7 also shows that as $K \to 1$, this distance also goes to 0. We now use this fact to show that decaying $K$ during training also allows FedSGD to minimise the true loss, with the added benefit of deceasing client computation per round during training.

Our goal is to derive a bound on the distance between iterations of FedSGD (where $K$ is decreasing between iterations), and the true minimiser $x^*$. We first need to bound the distance between surrogate minimisers with different values of $K$. For this analysis, we assume a continuous $K$. In reality $K$ is an integer, but continuous $K$ can be approximated by rounding $K_t$. Derivations of the following Theorems and Corollaries are given in Appendix A.

**Theorem 1** *Let* $0 < \gamma < L^{-1}$ *and* $1 \leq K_1 \leq K_2$*, then*

$$\|x^*(\gamma, K_1) - x^*(\gamma, K_2)\| \leq \sigma_c \left(1 + \frac{\phi_L(\gamma, K_2)}{\phi_\mu(\gamma, K_2)}\right) \left(\frac{\phi_L(\gamma, K_1) - \frac{K_1}{K_2}\phi_L(\gamma, K_2)}{\phi_\mu(\gamma, K_1)}\right). \tag{8}$$

The above theorem bounds the distance between the surrogate minimisers for two values of $K$, however, it can be simplified to make the dependence on $K_1$ and $K_2$ more explicit.

**Corollary 1** *Let* $0 < \gamma < L^{-1}$ *and* $1 \leq K_1 \leq K_2$*, then*

$$\|x^*(\gamma, K_1) - x^*(\gamma, K_2)\| \leq 2\sigma_c \frac{L^2}{\mu^2} \left( \frac{K_2 - K_1}{K_2} \right) \tag{9}$$

$$\leq 2\sigma_c \frac{L^2}{\mu^2} (K_2 - K_1). \tag{10}$$

Although Corollary 1 provides a looser bound than Theorem 1, the dependence on the distance $(K_2 - K_1)$ is clearer. Equation 10 comes from the fact that $K_2 \geq 1$. Both Equation 9 and the looser bound in Equation 10 are used within Theorem 2 to show that with appropriately chosen rate of $K$-decay, successive surrogate minimisers will be close enough such that the distance does not increase faster than FedSGD can converge given the choice of $\eta$.

To bound the distance between iterations of FedSGD, $x_t$, and the true minimiser, $x^*$, appropriate schedules of $\eta_t$, $\gamma$, and $K_t$ must be chosen. Here we decay the $\eta_t$ with $(1/t)$ as per Charles & Konecný (2020), but with fixed $\gamma$. $K_t$ is decayed with $(1/t)$. The motivation behind the inclusion of $b$ in the definition of $K_t$ is not immediately obvious, but is required to ensure the surrogate minimiser does not change too fast for FedSGD to converge. If the condition number $\kappa = L/\mu$ is large enough to impact the choice of $K_0$, the user could simply increase $K_0$ to sufficiently counteract $\kappa$ (albeit at the cost of a lower $\gamma$).

**Theorem 2** *Let*

$$\eta_t = \frac{a_t}{b+t}, a_t = \frac{3}{\phi_\mu(\gamma, K_t)}, b = 3\kappa^2, \gamma \leq \frac{\ln(2)}{K_0 \mu}, K_t = \max\left\{ \frac{K_0}{b+t}, 1 \right\}, \tag{11}$$

*then*

$$\mathbb{E}[\|x_t - x^*\|^2] \leq \frac{2\nu + 16\sigma_c^2 \kappa^2 K_0^2}{b+t}, \tag{12}$$

*where*

$$\nu = \max\left\{ \frac{36G^2}{\mu^2 K_0 M B}, (b+1)\|x_1 - x_1^*\|^2 \right\}. \tag{13}$$

$G$ is a bound on client-update variance given in Charles & Konecný (2020), $M$ is the number of clients participating in each round and $B$ is the batch size used. This theorem shows that with FedSGD, the distance between the model at each step and the true minimiser can be bounded when $\eta$ and $K$ are decayed. As per Equation 11, when using constant $\gamma$, $\gamma$ must be set sufficiently small to ensure convergence with the largest value of $K$, namely $K_0$. The $16\sigma_c^2\kappa^2 K_0^2$ term shows that choosing a larger value of $K_0$ increases the maximum distance to the true minimiser, and this will most likely dominate the $1/K_0$ term present in the definition of $\nu$ in most cases. However, as demonstrated later for real-world datasets, choice of larger $K_0$ leads generally to faster convergence (at least in the initial stages of training).

## 3 Illustrative Example

As discussed above, the objective function minimised by FedSGD is not necessarily the same as the function that would be minimised by centralised training on pooled data. In order to intuitively demonstrate the difference, we perform FedSGD on a simple one-dimensional quadratic problem using varying $K$, the decaying $\gamma$ proposed by Charles & Konecný (2020), and the decaying $K$ scheme proposed here.

We simulate clients with a quadratic loss function given by $f(x; z) = \frac{1}{2}zx^2 - x$. Each client is given an identity $i \in [1, 3]$, and possesses one data point $z_i = i$. Therefore, the minimiser for client $i$, $x_i^*$ is given by $1/z = 1/i$. The (un-normalised) probability of each client being selected each round is given by: $p(i) = 1/\sqrt{i}$. The true loss function (as per Equation 3) is therefore the expectation over client loss functions, $f(x) = \mathbb{E}[f_i(x)] = \int_1^3 (\frac{1}{2}zx^2 - x) z^{\frac{-1}{2}} dz$. $f(x)$ is minimised by $x^* \approx 0.523$. Figure 1 shows the models produced by FedSGD with different $\gamma$ and $K$ schedules. For all simulations we use server learning rate $\eta = 0.001$, $M = 10$ clients per round, batch size

$B = 1$, and initial client learning rate $\gamma = 0.1$. We start with an arbitrarily-chosen initial model value of $x_1 = 0.4$ and average each curve over 10 runs. We also vary the $\gamma$-decay rate, $d_\gamma$, where $\gamma_t = \gamma(d_\gamma)^t$, and $K$-decay rate, $d_K$, where $K_t = \lceil K_0(d_K)^t \rceil$.

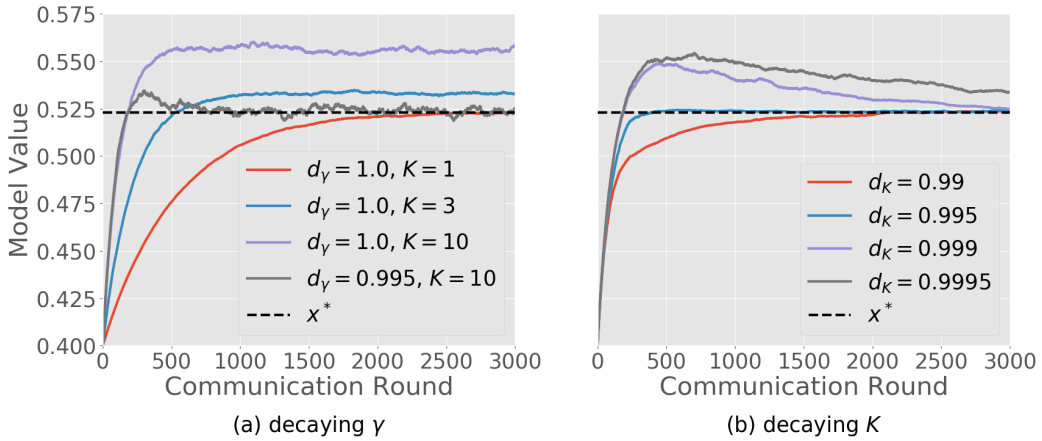


Figure 1: Convergence of FedSGD to true minimiser $x^*$ on a simulated quadratic dataset. (**a**) Varying $K$ with constant $\gamma$ ($d_\gamma = 1.0$), and decaying $\gamma$ ($d_\gamma = 0.995$). (**b**) Varying $d_K$ with constant $\gamma$.

Figure 1 (a) shows that with constant $\gamma$ ($d_\gamma = 1$), $\|x^*(\gamma, K) - x^*\|$ increases with $K$ (red, blue, purple curves), as per Equation 7. Larger values of $K$ converge faster because clients perform more steps of SGD between rounds. Decaying $\gamma$ during training (grey curve) when $K > 1$, however, gives both fast initial convergence and convergence to $x^*$. As can be seen at round 500, the grey curve overshoots $x^*$ before converging. This illustrates the need for correctly tuning $d_\gamma$: overly-aggressive decay will converge slowly due to low learning rates, but weak decay will cause FedSGD to begin to converge to a surrogate minimiser before $x^*$, also slowing down training.

A similar effect is demonstrated in Figure 1 (b). Here all schemes use constant $\gamma$ ($d_\gamma = 1$), and $K_0 = 10$. We vary $d_K$ to illustrate similar training dynamics as occur when varying $d_\gamma$, as discussed above. With aggressive decay ($d_K = 0.99$, red curve), $K$ quickly decays to 1, so FedSGD does not converge much faster than distributed SGD. This is in contrast to the very slow decay case ($d_K = 0.9995$, grey curve), which shows very rapid initial convergence, but ultimately takes longer to reach $x^*$ because $\|x^*(\gamma, K) - x^*\|$ was not reduced rapidly enough. A decay rate of $d_K = 0.995$ is a good choice here due to fastest convergence to $x^*$, and perform $84\%$ less total SGD steps compared to clients with constant $K = 10$ over 3000 rounds.

This simple 1-dimensional quadratic example serves to demonstrate the trade-off between faster convergence with larger $K$, and convergence to $x^*$. The benefit of using large $K_0$ and decaying $K$ during training is made apparent. We later perform FedSGD with fixed $\gamma$, annealing $\gamma$ and annealing $K$ on real-world datasets for non-quadratic models, to examine the impact of this theory in realistic scenarios and the implications for generalisation, which has not been studied before from this perspective.

## 4 Experimental Evaluation

### 4.1 Setup

We experimentally test FedSGD with a constant $\gamma$, decaying $\gamma$ as proposed by Charles & Konecný (2020), and with decaying $K$ as proposed in this paper, on three benchmark FL datasets: StackOverflow (Tensorflow-Federated, 2020), FEMNIST (Caldas et al., 2018), and Shakespeare (Caldas et al., 2018). The above analysis is for minimisation of quadratic objectives by FedSGD, so does not necessarily apply to non-quadratic objectives and generalisation. We therefore empirically study these characteristics for the first time. Details of the datasets and models are given below, with further details given in Appendix B.

**StackOverlow -** contains posts and their associated tags from stackoverflow.com, grouped by the writer of the post. We limit this dataset to 1000 training clients and 10,000 testing samples, with a maximum of 500 samples per client. As in Reddi et al. (2020) and Charles & Konecný (2020), we construct a normalised bag-of-words vector for presence of the 10,000 most-used tokens for each post, and a binary vector for presence of the 500 most-used tags. We train a logistic regression model with sigmoid outputs and Binary Cross-Entropy loss. Reddi et al. (2020) showed that for this dataset, replacing the SGD-like step of FedSGD (line 8 of Algorithm 1) with an adaptive optimiser such as Adam (FedAdam) significantly increases the rate of convergence. We therefore used FedAdam instead of FedSGD for this dataset, to allow the model to converge within 4000 rounds.

**FEMNIST -** is an adaptation of the EMNIST dataset. Each sample is a $32 \times 32$ greyscale image of one of 62 classes of lowercase letters, uppercase letters, and digits. Samples are grouped by the writer of the characters. We train a Convolutional Neural Network (CNN) on this dataset consisting of convolutional, max-pooling, and fully-connected layers, using Categorical Cross-Entropy loss. 1000 training clients were chosen at random, with $20\%$ of each user's samples held out for testing.

**Shakespeare -** the complete plays of William Shakespeare. The plays are parsed such that each role in each play becomes a client, and we take the 1000 clients with most samples, reserving the last $20\%$ of each client's samples for testing. This text data is then used for a next-character prediction task. A Gated Recurrent Unit (GRU) network was trained on the dataset using Categorical Cross-Entropy loss.

For each dataset, three values of $K$ were selected such that the model's test-set accuracy (Recall for StackOverflow) could converge within 4000 rounds. For each value of $K$, $\eta_0$, $d_\eta$ and $\gamma$ were tested in a grid search, and the combination of settings that gave the highest test accuracy over the last 500 rounds are presented (with accuracy measured every 40 rounds).

For the three value of $K$ found as above, experiments with decaying-$\gamma$ and decaying-$K$ were then performed. For the decaying-$\gamma$ experiments, $\eta_0$, $d_\eta$ and $\gamma_0$, $d_\gamma$ were varied in a grid search, using constant $K$. For the decaying-$K$ experiments, $\eta_0$, $d_\eta$, $\gamma$, and $d_K$ were searched. As above, the best combination of settings to give the highest test-set performance during the 4000 rounds were plotted for each $K$.

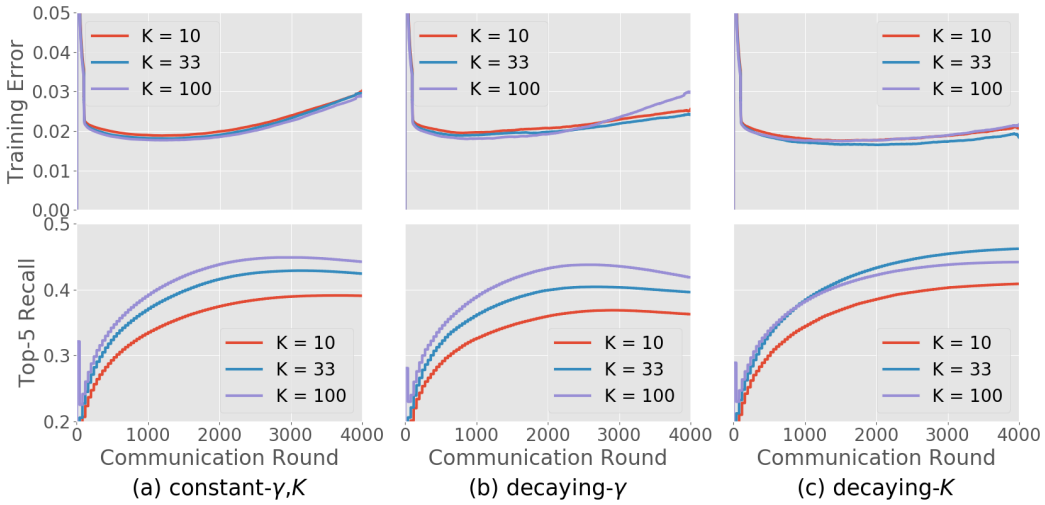


Figure 2: Training error and Top-5 Recall of FedAdam on the StackOverflow dataset. (a) Constant-$\gamma$/$K$ scheme, (b) decaying-$\gamma$ scheme, (c) decaying-$K$ scheme.

### 4.2 Results

Figures 2-4 show the training error and test-set performance for the constant-$\gamma$/$K$, decaying-$\gamma$ and decaying-$K$ schemes, for three datasets. Each curve is an average over 5 trials with different random seeds. Error curves are smoothed over a window of size 200

Figure 2 (a) shows that for the StackOverflow dataset, FedAdam with constant-$\gamma$/$K$ can converge both faster and to a higher test-set Recall with higher $K$. However, Figures 3 (a) and 4(a) have

different dynamics: higher $K$ converges faster, but final test accuracy is harmed. For all three datasets, the scenarios with higher $K$ were able to reach lower training error within 4000 rounds, despite using mostly the same $\gamma$ (hyperparameters for all settings are given in Table 5 of Appendix B). These results appear to be in contradiction to Equation 7, which states that with increasing $K$, the distance to the true minimiser increases for quadratic objectives. Equation 7 would imply that by increasing $K$, the minimum achievable training error would increase. However, when a decaying the server learning rate, $\eta$, is used, higher minimum error does not appear to be a problem (as opposed to the experiments in Charles & Konecný (2020), where fixed $\eta$ and $\gamma$ were used).

Figures 3 (b) and 4 (b) all show that by decaying $\gamma$ during training, the test-set performance for FEMNIST and Shakespeare is either the same or is improved compared to constant-$\gamma/K$. Table 1 gives the best test-set performance for each scheme and each setting. The proof earlier in the paper for quadaratic objective concerns minimisation of the loss function, and does not consider generalisation ability. It appears, however, that by decaying $\gamma$ during training, this generalisation performance can be improved. This is likely due to the fact that clients can perform more fine-tuned adjustments on the model between aggregations steps, and the 'noise' included with the global model by performing multiple steps of SGD between aggregations (represented by distortion-matrix $Q_i$ in the above analysis) is reduced. For StackOverflow, however, decaying $\gamma$ suring training could not improve over constant $\gamma$. Here, it appears that improvement can only be made by decaying $K$.

The decaying-$K$ scheme shown in Figures 3 (c) and 4 (c) give very similar improvements to test-set performance as the decaying-$\gamma$ scheme. By decaying $K$ during training, the maximum test-set performance that the global models achieved during training was either the same or improved compared to constant-$\gamma/K$. However, with this scheme, clients performed significantly less total SGD steps and thus computation over the 4000 training rounds compared to the other two schemes. Table 1 presents the total amount of SGD steps saved by the decaying-$K$ schemes over the 4000 training rounds for the different scenarios. For the StackOverflow dataset in Figure 2 (c), significant improvement in terms of test-set Recall were made over constant-$\gamma/K$ or decaying-$\gamma$. The unusual training dynamics of this dataset compared to FEMNIST and Shakespeare may be due to the use of FedAdam and/or the convex model. Future work may benefit from comparing the training dynamics of different typos of model and Federated optimiser.

These results imply that the convergence-speed benefit gained by performing multiple steps of SGD between aggregations comes at the cost of lower test-set performance. Intuitively, by performing multiple steps of SGD between aggregations in FL, the performance of the global model is slightly degraded as the heterogeneity in client data distributions is amplified, as demonstrated in the above analysis. However, towards the end of training, more 'fine-tuned' adjustments need to be made to the model to improve performance. Instead of decaying $\gamma$ to try to mitigate this performance degradation, these results show that decaying-$K$ to turn FedSGD (and its variants) into simple distributed-SGD can achieve the same objective.

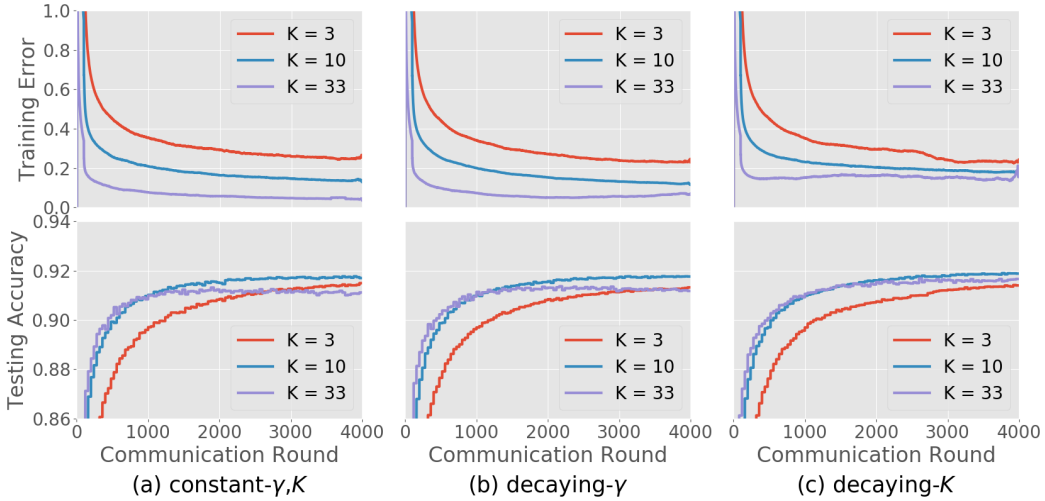


Figure 3: Training error and Top-1 Accuracy of FedSGD on the FEMNIST dataset. (a) Constant-$\gamma/K$ scheme, (b) decaying-$\gamma$ scheme, (c) decaying-$K$ scheme.

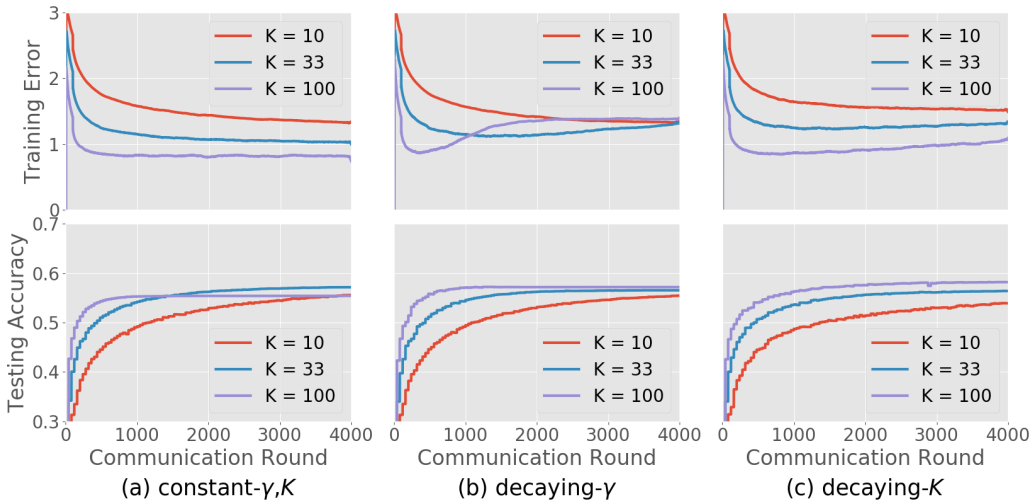


Figure 4: Training error and Top-1 Accuracy of FedSGD on the Shakespeare dataset. (a) Constant-$\gamma$/$K$ scheme, (b) decaying-$\gamma$ scheme, (c) decaying-$K$ scheme.

Table 1: Lowest training error and highest test-set accuracy/recall achieved during training for the decaying-$\gamma$ and decaying-$K$ schemes for different training scenarios, and the total number of SGD steps saved by decaying-$K$ over 4000 rounds. Best overall test-set performance(s) for each dataset given in bold.

| | | StackOverflow | | | FEMNIST | | | Shakespeare | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | K | 10 | 33 | 100 | 3 | 10 | 33 | 10 | 33 | 100 |
| Decaying $\gamma$ | Min Train Error | 0.02 | 0.02 | 0.02 | 0.22 | 0.11 | 0.05 | 1.33 | 1.12 | 0.87 |
| | Max Test Acc | 0.37 | 0.40 | 0.44 | 0.91 | **0.92** | 0.91 | 0.55 | 0.57 | 0.57 |
| Decaying $K$ | Min Test Error | 0.02 | 0.02 | 0.02 | 0.23 | 0.16 | 0.14 | 1.44 | 1.23 | 0.84 |
| | Max Test Acc | 0.41 | **0.46** | 0.42 | 0.91 | **0.92** | **0.92** | 0.54 | 0.56 | **0.58** |
| Fraction SGD Steps Used | | 0.55 | 0.51 | 0.50 | 0.65 | 0.55 | 0.26 | 0.55 | 0.51 | 0.50 |

The results in Table 1 show that the test-set performance achieved by decaying-$\gamma$ can be matched or even surpassed by decaying-$K$, whilst also performing far less computation during training. This saved computation would represent a significant energy and time saving for a real-word FL deployment.

## 5 Conclusion

In this work, we proposed decaying the number of local SGD steps, $K$, performed in the ClientUpdate phase of the popular FedSGD algorithm during Federated Learning (FL). This leads the FedSGD algorithm to become distributed SGD as $K \to 1$. We proved that for quadratic objectives, decaying $K$ allows FedSGD to minimise the loss function that would have been minimised during centralised training on pooled client data, i.e. the *true* loss. We demonstrated on a simple quadratic example the importance of tuning the decay rate of $K$, in order to achieve both the fast initial convergence provided by FedSGD, but also minimisation of the true loss. Previous works suggest decaying the client learning rate, $\gamma$, during training in order to minimise the true loss. However, thorough experimentation of three benchmark FL datasets indicates that decaying $K$ provides very similar benefit in terms of generalisation performance as decaying $\gamma$, with the significant added benefit of decreasing the total number of SGD steps that clients must perform during FedSGD. In our experiments, out decaying-$K$ scheme was able match or surpass the test-set performance of the decaying-$\gamma$ scheme in all scenarios, with up to $3.8\times$ less steps of SGD performed by clients. In real-world FL deployments, this would represent significant energy and time savings for the FL process.

## REFERENCES

Muhammad Ammad-ud-din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888*, 2019.

Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Zachary Charles and Jakub Konecný. On the outsized importance of learning rates in local update methods. *arXiv e-prints arXiv:2007.00878*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1126–1135, 2017.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *arXiv e-prints arXiv:1811.03604*, 2018.

Li Huang, Andrew L. Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of Biomedical Informatics*, 99:103291, 2019. ISSN 1532-0464.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, et al. Advances and open problems in federated learning. *arXiv e-prints arXiv:1912.04977*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv e-prints arXiv:1412.6980*, 2014.

David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6341–6345, 2019.

Qinbin Li, Zeyi Wen, and Bingsheng He. Pratical federated gradient boosting decision trees. In *34th AAAI Conference on Artificial Intelligence*, volume 34, pp. 4642–4649, 2020a.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, may 2020b. doi: 10.1109/MSP.2020.2975749.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2020c.

Sin Kit Lo, Qinghua Lu, Chen Wang, Hye-Young Paik, and Liming Zhu. A systematic literature review of federated machine learning: From a software engineering perspective. *Journal of the Association for Computing Machinery*, 37(4), Aug 2020. doi: 10.1145/1122445.1122456.

Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. *20th International Conference on Artifical Intelligence and Statistics*, 2017.

Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard. *arXiv e-prints arXiv:1906.04329*, 2019.

S. Reddi, Zachary Charles, M. Zaheer, Zachary A. Garrett, Keith Rush, Jakub Konecný, S. Kumar, and H. McMahan. Adaptive federated optimization. *arXiv e-prints arXiv:2003.00295*, 2020.

Micah Sheller, Brandon Edwards, G. Reina, Jason Martin, Sarthak Pati, Aikaterini Kotrotsou, Mikhail Milchenko, Weilin Xu, Daniel Marcus, Rivka Colen, and Spyridon Bakas. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10, 12 2020. doi: 10.1038/s41598-020-69250-1.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4427–4437, 2017. ISBN 9781510860964.

Haifeng Sun, Shiqi Li, F. Richard Yu, Qi Qi, Jingyu Wang, and Jianxin Liao. Towards communication-efficient federated learning in the internet of things with edge computing. *IEEE Internet of Things Journal*, 2020.

Tensorflow-Federated. Tensorflow federated stackoverflow dataset. 2020. URL `https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow/load_data`.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020.

Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv e-prints arXiv:1910.10252*, 2019.

Blake Woodworth, Kumar Kshitij Patel, Sebastian U. Stich, Zhen Dai, Brian Bullins, H. Brendan McMahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? *arXiv e-prints arXiv:2002.07839*, 2020.

# APPENDIX A PROOF OF THEOREMS

## A.1 THEOREM 1

Theorem 1 bounds the distance between surrogate minimisers that use different values of $K$. Our proof extends the proof of (Charles & Konecný, 2020): the Theorems, Corollaries and Lemmas up to Theorem 21 are used in this proof, with modification to/reparametrisations of some formulas. We encourage readers to follow the previous proofs to more thoroughly understand the proof presented here. The major reparamterisation is to Lemma 10, which gives the eigenvalues of matrix $Q_i(\gamma, K)A_i$ as a function of $\lambda$ (any eigenvalue of $A_i$), $\gamma$ and $K$:

$$\phi_{\lambda,\gamma}(K) = \gamma^{-1}(1 - (1 - \gamma\lambda)^K). \tag{14}$$

This is because we use constant $\gamma$ and varying $K$ during training, rather than constant $K$ with varying $\gamma$.

### A.1.1 LEMMA 1

If $\gamma \leq L^{-1}$ (assuming $\mu I \preceq A_i \preceq LI, \forall i \in \mathcal{I}$), $1 \leq K_1 \leq K_2$, then:

$$C = K_1^{-1}Q_i(\gamma, K_1)A_i - K_2^{-1}Q_i(\gamma, K_2)A_i \tag{15}$$

is positive semidefinite and satisfies

$$\lambda_{\max}(C) \leq K_1^{-1}\phi_{L,\gamma}(K_1) - K_2^{-1}\phi_{L,\gamma}(K_2). \tag{16}$$

**Proof** Since $Q_i(\gamma, K_1)A_i$ and $Q_i(\gamma, K_2)A_i$ share the same eigenvectors as $A_i$, and the eigenvalues of a matrix $M$ multiplied by scalar $s$ are equal to the eigenvalues of $sM$, using Equation 14, the eigenvalues of $C$ are of the form

$$h(\lambda) = K_1^{-1}\gamma^{-1}(1 - (1 - \gamma\lambda)^{K_1}) - K_2^{-1}\gamma^{-1}(1 - (1 - \gamma\lambda)^{K_2}). \tag{17}$$

Since $\gamma \leq L_i^{-1}$, inspection of $h(\lambda)$ shows that $h(\lambda) \geq 0$ for $\lambda \in [\mu_i, L_i]$ and $1 \leq K_1 \leq K_2$. This implies $C$ is positive semidefinite. Computing $h'(\lambda)$ gives:

$$h'(\lambda) = (1 - \lambda\gamma)^{K_1 - 1} - (1 - \gamma\lambda)^{K_2 - 1}. \tag{18}$$

As $0 < (1 - \gamma\lambda) < 1$ and $K_1 \leq K_2$ then $h'(\lambda) \geq 0$. The maximum eigenvalue of $C$ therefore satisfies

$$\lambda_{\max}(C) \leq h(\lambda) = K_1^{-1}\phi_{L,\gamma}(K_1) - K_2^{-1}\phi_{L,\gamma}(K_2). \tag{19}$$

### A.1.2 PROOF OF THEOREM 1

Define the following terms for notational convenience:

$$\begin{aligned} B_{i,j} &= K_j^{-1}Q_i(\gamma, K_j)A_i \\ B_j &= \mathop{\mathbb{E}}_i[B_{i,j}] \\ v_j &= \mathop{\mathbb{E}}_i[B_{i,1}c_i] - \mathop{\mathbb{E}}_i[B_{i,2}]\mathop{\mathbb{E}}_i[c_i] \\ \phi_{L,K} &= \phi_{L,\gamma}(K) \\ \phi_{\mu,K} &= \phi_{\mu,\gamma}(K) \end{aligned}$$

As per Lemma 30 of (Charles & Konecný, 2020):

$$\|x^*(\gamma, K_1) - x^*(\gamma, K_2)\| \leq \underbrace{\|B_1^{-1}\|}_{T_1}\underbrace{\|v_1 - v_2\|}_{T_2} + \underbrace{\|B_1^{-1} - B_2^{-1}\|}_{T_3}\underbrace{\|v_2\|}_{T_4}. \tag{20}$$

To bound this term, we derive bounds on $T_1, T_2, T_3, T_4$. Bounding $T_1$ using Theorem 16, Lemma 31, and Lemma 11 of (Charles & Konecný, 2020):

$$\begin{aligned} T_1 &= \|B_1^{-1}\| && (21)\\ &\leq \mathbb{E}[\|B_{1,i}^{-1}\|] && (22)\\ &= \mathbb{E}[\|(K_1^{-1}Q_i(\gamma, K_1)A_i)^{-1}\|] && (23)\\ &\leq K_1\phi_{\mu,K_1}^{-1}. && (24) \end{aligned}$$

Bounding $T_2$ using the definition of $B_{i,j}$ and Lemma 1:

$$T_2 = \|v_1 - v_2\| \tag{25}$$
$$= \|\mathbb{E}[(B_{i,1} - B_{i,2})(c_i - c)]\| \tag{26}$$
$$\leq \sqrt{\mathbb{E}[\|B_{i,1} - B_{i,2}\|^2]\mathbb{E}[\|c_i - c\|^2]} \tag{27}$$
$$\leq \sigma_c \sqrt{\mathbb{E}[\|B_{i,1} - B_{i,2}\|^2]} \tag{28}$$
$$= \sigma_c \sqrt{\mathbb{E}[\|K_1^{-1} Q_i(\gamma, K_1) A_i - K_2^{-1} Q_i(\gamma, K_2) A_i\|]} \tag{29}$$
$$\leq \sigma_c \left(K_1^{-1} \phi_{L,K_1} - K_2^{-1} \phi_{L,K_2}\right). \tag{30}$$

Using the fact that for invertible matrices $X, Y \in \mathbb{R}^{n \times n}$:

$$\|X^{-1} - Y^{-1}\| = \|X^{-1}(X - Y)Y^{-1}\| \leq \|X^{-1}\| \|Y^{-1}\| \|X - Y\|. \tag{31}$$

Applying this to $T_3$, bounding $\|B_1^{-1}\|$ and $\|B_2^{-1}\|$ as per $T_1$, and bounding $\|B_1 - B_2\|$ using Lemma 1:

$$T_3 \leq \|B_1^{-1}\| \|B_2^{-1}\| \|B_1 - B_2\| \tag{32}$$
$$\leq \left(K_1 \phi_{\mu,K_1}^{-1}\right) \left(K_2 \phi_{\mu,K_2}^{-1}\right) \left(K_1^{-1} \phi_{L,K_1} - K_2^{-1} \phi_{L,K_2}\right). \tag{33}$$

$T_4$ can be bounded using the definition of $\sigma_c$ and $\phi_{L,\gamma}(K)$:

$$T_4 = \|\mathop{\mathbb{E}}_i[B_{i,2} c_i] - \mathop{\mathbb{E}}_i[B_{i,2}] \mathop{\mathbb{E}}_i[c_i]\| \tag{34}$$
$$\leq \sigma_c \sqrt{\mathop{\mathbb{E}}_i[\|B_{i,2}\|]^2} \tag{35}$$
$$= \sigma_c \sqrt{\mathop{\mathbb{E}}_i[\|K_2^{-1} Q_i(\gamma, \Theta_{1:K_2}) A_i\|^2]} \tag{36}$$
$$\leq \sigma_c K_2^{-1} \phi_{L,K_2}. \tag{37}$$

Combining the terms in Equation 20:

$$\|x^*(\gamma, K_1) - x^*(\gamma, K_2)\| \leq T_1 T_2 + T_3 T_4 \tag{38}$$
$$\leq \left(\frac{K_1}{\phi_{\mu,K_1}}\right) \left(\sigma_c \left(\frac{\phi_{L,K_1}}{K_1} - \frac{\phi_{L,K_2}}{K_2}\right)\right) \tag{39}$$
$$+ \left(\left(\frac{K_1}{\phi_{\mu,K_1}}\right) \left(\frac{K_2}{\phi_{\mu,K_2}}\right) \left(\frac{\phi_{L,K_1}}{K_1} - \frac{\phi_{L,K_2}}{K_2}\right)\right) \left(\sigma_c \frac{\phi_{L,K_2}}{K_2}\right) \tag{40}$$
$$= \sigma_c \left(\frac{\phi_{L,K_1} - \frac{K_1}{K_2}\phi_{L,K_2}}{\phi_{\mu,K_1}} + \frac{\phi_{L,K_1}\phi_{L,K_2} - \frac{K_1}{K_2}\phi_{L,K_2}^2}{\phi_{\mu,K_1}\phi_{\mu,K_2}}\right) \tag{41}$$
$$= \sigma_c \left(1 + \frac{\phi_{L,\gamma}(K_2)}{\phi_{\mu,\gamma}(K_2)}\right) \left(\frac{\phi_{L,\gamma}(K_1) - \frac{K_1}{K_2}\phi_{L,\gamma}(K_2)}{\phi_{\mu,\gamma}(K_1)}\right). \tag{42}$$

### A.2 COROLLARY 1

The $(\phi_{L,\gamma}(K_1) - \frac{K_1}{K_2}\phi_{L,\gamma}(K_2))$ term from Equation 42 first needs to be bounded. Using the definition of $\phi_{L,\gamma}(K)$:

$$\phi_{L,\gamma}(K_1) - \frac{K_1}{K_2}\phi_{L,\gamma}(K_2) = \sum_{k=1}^{K_1} (1 - \gamma L)^{k-1} L - \frac{K_1}{K_2} \sum_{k=1}^{K_2} (1 - \gamma L)^{k-1} L \tag{43}$$
$$\leq \sum_{k=1}^{K_1} (1 - \gamma L)^{k-1} L - \frac{K_1}{K_2} \sum_{k=1}^{K_1} (1 - \gamma L)^{k-1} L \tag{44}$$
$$= \left(1 - \frac{K_1}{K_2}\right) \phi_{L,\gamma}(K_1). \tag{45}$$

Substituting this and Lemma 35 from (Charles & Konecný, 2020) $\left(\frac{\phi_{L,\gamma}(K)}{\phi_{\mu,\gamma}(K)} \leq \frac{L}{\mu}\right)$ into Theorem 1, and using the fact that $L/\mu \geq 1$ and $K_2 \geq 1$:

$$\|x^*(\gamma, \Theta_{1:K_1}) - x^*(\gamma, \Theta_{1:K_2})\| \leq \sigma_c \left(1 + \frac{\phi_{L,\gamma}(K_2)}{\phi_{\mu,\gamma}(K_2)}\right) \left(\frac{\phi_{L,\gamma}(K_1) - \frac{K_1}{K_2}\phi_{L,\gamma}(K_2)}{\phi_{\mu,\gamma}(K_1)}\right) \tag{46}$$

$$\leq \sigma_c \left(1 + \frac{L}{\mu}\right) \left(\left(1 - \frac{K_1}{K_2}\right) \frac{\phi_{L,\gamma}(K_1)}{\phi_{\mu,\gamma}(K_1)}\right) \tag{47}$$

$$\leq 2\sigma_c \frac{L^2}{\mu^2} \left(\frac{K_2 - K_1}{K_2}\right) \tag{48}$$

$$\leq 2\sigma_c \frac{L^2}{\mu^2} (K_2 - K_1). \tag{49}$$

For the later Theorem 2, both Equation 48 and Equation 49 are used to bound the distance between minimisers with different values of $K$.

### A.3 THEOREM 2

Defining the following for notational convenience:

$$\begin{aligned} \mu_t &= \phi_{\mu,\gamma}(K_t) \\ L_t &= \phi_{L,\gamma}(K_t) \\ \tilde{f}_t(x) &= \tilde{f}(x, \gamma, K_t) \\ x_t^* &= \arg\min \tilde{f}_t(x) \\ \kappa &= L/\mu \end{aligned}$$

The distance from any $x_t$ and $x^*$ can be bounded as:

$$\mathbb{E}[\|x_t - x^*\|^2] \leq 2\mathbb{E}[\|x_t - x_t^*\|^2] + 2\|x_t^* - x^*\|^2. \tag{50}$$

Lemma 25 of (Charles & Konecný, 2020) bounds the distance between $x_{t+1}$ and $x_t^*$. The authors then use this to form a recursion relation for consecutive steps of FedSGD, for any $\omega_t > 0$:

$$\mathbb{E}[\|x_{t+1} - x_t^*\|^2] \leq (1 - \eta_t\mu_t)\, \mathbb{E}[\|x_t - x_t^*\|] + \eta_t^2 \frac{K_t G^2}{MB}, \tag{51}$$

$$\alpha_{t+1} \leq (1 + \omega_t) \left((1 - \eta_t\mu_t)\, \alpha_t + \eta_t^2 \frac{K_t G^2}{MB}\right) + \left(1 + \omega_t^{-1}\right) \|x_t^* - x_{t+1}^*\|^2. \tag{52}$$

Let $C = 4\sigma_c^2\kappa^2$ and let $\hat{t} = b + t$. We create the following inductive hypothesis:

$$\alpha_t \leq \frac{\nu + C}{\hat{t}}. \tag{53}$$

The term for $t = 1$ is given as:

$$\alpha_1 = \|x_1 - x_1^*\|^2 = \frac{(b+1)\|x_1 - x_1^*\|^2}{b+1} \leq \frac{\nu_1}{b+1}. \tag{54}$$

Which therefore means we can come to the same conclusions, except using a $K_t$ term instead of $K$:

$$(1 - \eta_t\mu_t)\, \alpha_t + \eta_t^2 \frac{K_t G^2}{MB} \leq \left(\frac{\hat{t} - 2}{\hat{t}}\right)(\nu + C) - \frac{C}{\hat{t}^2} + \frac{1}{\hat{t}^2} \underbrace{\left(\frac{9K_t G^2}{\mu_t^2 MB} - \nu\right)}_{\xi_t}. \tag{55}$$

Lemma 18 of (Charles & Konecný, 2020) allows $\xi_t$ to be simply bounded. The choice of a constant $\gamma$ and decaying $K$ means the tightest constraint on $\gamma$ must be used for any value of $K$ to ensure convergence, i.e. $\gamma \leq \ln(2)/K_0\mu$. Substituting this into the definition of $\xi_t$, using the fact that $K_t/K_0 \leq 1$, and using the definition of $\nu$:

$$\xi_t \leq \frac{9K_t G^2}{\left(\frac{K_0\mu}{2}\right)^2 MB} - \frac{36G^2}{\mu^2 K_0 MB} \leq 0, \tag{56}$$

therefore:

$$(1+\omega_t)\left((1-\eta_t\mu_t)\,\alpha_t+\eta_t^2\frac{K_tG^2}{MB}\right)\leq\frac{\nu+C}{\hat{t}+1}-\frac{C}{(\hat{t}-2)(\hat{t}+1)}. \tag{57}$$

This bounds the first part of Equation 52. Using Corollary 1 (Equation 49), and the fact that $\kappa^2 < b < \hat{t}+1$:

$$\|x_t^*-x_{t+1}^*\|^2\leq 4\sigma_c^2\frac{L^4}{\mu^4}\left(\frac{K_0}{\hat{t}}-\frac{K_0}{\hat{t}+1}\right)^2 \tag{58}$$

$$=\frac{C\kappa^2}{\hat{t}^2\left(\hat{t}+1\right)^2} \tag{59}$$

$$\leq\frac{C}{\hat{t}^2\left(\hat{t}+1\right)}. \tag{60}$$

Then:

$$\left(1+\omega_t^{-1}\right)\|x_t^*-x_{t+1}^*\|^2\leq\frac{C}{\left(\hat{t}+2\right)\left(\hat{t}+1\right)}. \tag{61}$$

Combining Equation 52, Equation 57, Equation 61 proves the inductive hypothesis:

$$\mathbb{E}[\|x_{t+1}-x_t^*\|^2]=\alpha_{t+1}\leq\frac{\nu+C}{\hat{t}+1}-\frac{C}{\left(\hat{t}-2\right)\left(\hat{t}+1\right)}+\frac{C}{\left(\hat{t}+2\right)\left(\hat{t}+1\right)}\leq\frac{\nu+C}{\hat{t}+1}. \tag{62}$$

The second term of Equation 50 can be bounded using Corollary 1 (Equation 48), using $K_1 = 1$ and $K_2 = K_t$, and the same $\kappa^2 < b < \hat{t}+1$ trick:

$$\|x_t^*-x_{t+1}^*\|^2\leq 4\sigma_c^2\frac{L^4}{\mu^4}\left(\frac{\frac{K_0}{\hat{t}}-\frac{K_0}{\hat{t}+1}}{\frac{K_0}{\hat{t}}}\right)^2 \tag{63}$$

$$=\frac{4\sigma_c^2\kappa^4}{\left(\hat{t}+1\right)^2} \tag{64}$$

$$\leq\frac{4\sigma_c^2\kappa^2}{\hat{t}+1}. \tag{65}$$

Combining equation 50, equation 62 and Equation 65 gives the final Theorem 2:

$$\mathbb{E}[\|x_t-x^*\|^2]\leq\frac{2\left(\nu+C\right)}{\hat{t}+1}+2\left(\frac{4\sigma_c^2\kappa^2}{\hat{t}+1}\right) \tag{66}$$

$$\leq\frac{2\nu+8\sigma_c^2\kappa^2\left(K_0^2+1\right)}{b+t} \tag{67}$$

$$\leq\frac{2\nu+16\sigma_c^2\kappa^2K_0^2}{b+t}. \tag{68}$$

## APPENDIX B EXPERIMENTAL DETAILS

Details of the datasets, model architectures and hyperparameters used for Figures 2-4 and Table I are given here. All models were implemented in Tensorflow.

**StackOverflow -** posts from the website stackoverflow.com. Each post has an associated list of tags, which is used as the target for each sample. A post about web development may have the tags: {'html', 'css', 'javascript'}, for example. The data from (Tensorflow-Federated, 2020) is preprocessed including lowercasing text, separating every token with spaces, removing non-ascii symbols etc. Similar to (Reddi et al., 2020) and (Charles & Konecný, 2020), we take the 10,000 most common tokens and the 500 most common tags, and construct a (normalised to 1) bag-of-words vector and a binary tag-vector for each sample. Due to very large variance in number of samples per user and total number of users available in the dataset, we limit to a maximum of 250 samples per client, 1000 training clients (for a total of 331027 training samples) and 10000 testing samples. The model trained is a 1-vs-rest logistic regression model with 10,000 inputs and 500 outputs, trained with Binary Cross-Entropy. As previously shown in (Reddi et al., 2020), convergence for this dataset using this model is slow, and did not converge within 4000 rounds. (Reddi et al., 2020) showed that replacing the update rule of ServerUpdate (Algorithm 1, line 8) with an adaptive optimiser such as Adam (FedAdam) gave much faster convergence than FedSGD for this dataset. The psuedo-code for FedAdam as we used in the StackOverflow experiments is given below. The performance metric used for this dataset is Recall using the Top-5 predictions of the model as implemented in tensorflow.keras.metrics.Recall. For all experiments we fix $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ as per the original Adam paper (Kingma & Ba, 2014).

**Algorithm 3** FedAdam: ServerUpdate

Initialise global model, $x_1$
Initialise 1st and 2nd moments, $m_1 \leftarrow 0$, $v_1 \leftarrow 0$
**while** termination criteria not met **do**
  Select round clients, $\mathcal{I}_t$
  **for each** client $i \in \mathcal{I}_t$ in parallel **do**
    $q_t^i \leftarrow \mathrm{ClientUpdate}^i(x, \gamma, K)$
  **end for**
  $q_t \leftarrow (1/M) \sum_{i \in I_t} q_t^i$
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) q_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)(q_t)^2$
  $\eta_t \leftarrow \alpha \frac{\sqrt{1-(\beta_2)^t}}{1-(\beta_1)^t}$
  $x_{t+1} \leftarrow x_t - \eta_t \frac{m_t}{\sqrt{v_t}+\epsilon}$
**end while**

**FEMNIST -** the EMNIST dataset where the samples are grouped by the person who wrote the characters. Samples are a $32 \times 32$ greyscale image of the characters a-zA-Z0-9. 2000 users from the dataset were used for training, with $20\%$ of each user's samples held out for testing. Preprocessing and grouping was performed using the utilities provided by (Caldas et al., 2018). The Convolutional (Conv) Neural Network model trained on the dataset had the following architecture. Model was trained using Categorical Cross-Entropy loss.

Table 2: Architecture of FEMNIST CNN Model

| Layer | Details | Activation |
|---|---|---|
| Conv | $3 \times 3$ kernel, stride 1, 32 outputs | ReLU |
| Max Pooling | $2 \times 2$ pooling, stride 2 | - |
| Conv | $3 \times 3$ kernel, stride 1, 64 outputs | ReLU |
| Max Pooling | $2 \times 2$ pooling, stride 2 | - |
| Fully Connected | 512 outputs | ReLU |
| Fully Connected | 64 outputs | Softmax |

**Shakespeare -** the complete plays of William Shakespeare. Each character in each play becomes one client. The lines of each client are used as samples for a next-character-prediction task. 1000

clients were used for training, with the last $20\%$ of lines for each client used as the testing samples. Preprocessing and grouping was performed using the utilities provided by (Caldas et al., 2018). The Gated Recurrent Unit (GRU) model trained on the dataset had the following architecture. Model was trained using Categorical Cross-Entropy loss.

Table 3: Architecture of Shakespeare GRU Model

| Layer | Details | Activation |
|---|---|---|
| Character-embedding | $77 \rightarrow 8$ embedding | - |
| GRU | Tanh activation, sigmoid recurrent activation, $256$ outputs | - |
| GRU | Tanh activation, sigmoid recurrent activation, $256$ outputs | - |
| Fully-connected | 77 outputs | Softmax |

**Hyperparameters -** were tested using a grid-search of the relevant hyperparameters for each of the constant-$\gamma$/$K$, decaying-$\gamma$, decaying-$K$ schemes. Values tested are given below. Other relevant hyperparemeters include $C = 0.01$ fraction of clients sampled per round, batch size $B = 32$. The curves presented used the hyperparmeters that gave the highest average test accuracy over the last 500 rounds.

Table 4: Hyperparameter Grid Search Values

| Parameters | $\eta_0$, $\gamma_0$ | $K_0$ | $d_\eta$, $d_\gamma$, $d_K$ |
|---|---|---|---|
| Values | $\{3.0, 1.0, 0.3, 0.1, 0.03, 0.01\}$ | $\{100, 33, 10, 3\}$ | $\{0.9996, 0.999, 0.996, 0.99\}$ |

The following table gives the actual values chosen for each scenario.

Table 5: Hyperparameters Used for FL Scenarios

| | | constant-$\gamma$/$K$ | | | decaying-$\gamma$ | | | | decaying-$K$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | $K$ | $\eta_0$ | $d_\eta$ | $\gamma$ | $\eta_0$ | $d_\eta$ | $\gamma_0$ | $d_\gamma$ | $\eta_0$ | $d_\eta$ | $\gamma$ | $d_K$ |
| StackOverflow | 10 | 0.3 | Adam | 1.0 | 0.3 | Adam | 1.0 | 0.999 | 0.3 | Adam | 1.0 | 0.9996 |
| | 33 | 0.3 | Adam | 1.0 | 0.3 | Adam | 1.0 | 0.999 | 0.3 | Adam | 1.0 | 0.9996 |
| | 100 | 0.3 | Adam | 1.0 | 0.3 | Adam | 1.0 | 0.9996 | 0.3 | Adam | 1.0 | 0.9996 |
| FEMNIST | 3 | 1 | 0.9996 | 0.3 | 1 | 0.9996 | 0.3 | 0.9996 | 1 | 0.9996 | 0.3 | 0.9996 |
| | 10 | 1 | 0.9996 | 0.3 | 1 | 0.9996 | 0.3 | 0.9996 | 1 | 0.9996 | 0.3 | 0.9996 |
| | 100 | 1 | 0.9996 | 0.3 | 1 | 0.9996 | 0.3 | 0.9996 | 1 | 0.9996 | 0.3 | 0.999 |
| Shakespeare | 10 | 1.0 | 0.9996 | 1.0 | 1.0 | 0.9996 | 1.0 | 0.9996 | 1.0 | 0.9996 | 1.0 | 0.9996 |
| | 33 | 1.0 | 0.999 | 1.0 | 1.0 | 0.999 | 1.0 | 0.999 | 1.0 | 0.999 | 1.0 | 0.9996 |
| | 100 | 1.0 | 0.996 | 1.0 | 1.0 | 0.996 | 1.0 | 0.996 | 1.0 | 0.996 | 1.0 | 0.9996 |