Fed+: A Family of Fusion Algorithms for Federated Learning

Pengqian Yu*[1], Laura Wynter†[1], and Shiau Hong Lim‡[1]

[1] IBM Research, Singapore

Abstract

We present a class of methods for federated learning, which we call Fed+, pronounced FedPlus. The class of methods encompasses and unifies a number of recent algorithms proposed for federated learning and permits easily defining many new algorithms. The principal advantage of this class of methods is to better accommodate the real-world characteristics found in federated learning training, such as the lack of IID data across the parties in the federation. We demonstrate the use and benefits of this class of algorithms on standard benchmark datasets and a challenging real-world problem where catastrophic failure has a serious impact, namely in financial portfolio management.

1 Introduction

Federated learning is a technique for training machine learning models without sharing data, introduced by McMahan *et al.* [2017]; Konečnỳ *et al.* [2015, 2016], and gaining momentum over the past few years. Enabling parties to train large machine learning models without sharing data allows not only to satisfy privacy concerns, but also, in many cases, to increase accuracy and reduce training times. For many real-world applications it is not feasible to share data outside of an organization or a country. This directive has been enshrined into law in various jurisdictions, notably GDPR in the European Union Hoofnagle *et al.* [2019], and FERPA Sieber [2007] (education), HIPAA Cohen and Mello [2018] (medical data), in the United States. The issue of data privacy is also of importance to public opinion, as evidenced by reactions to recent data breaches Cheng *et al.* [2017]; Kammoun *et al.* [2019]. Federated learning, by allowing model training while respecting these regulations, has appealed to practitioners in numerous domains from medical research to finance to the Internet of Things.

Federated learning involves a possibly varying set of parties participating in a parallel training process through a centralized aggregator. The aggregator has access only to the parties' model parameters, or gradients, but not to the data itself. Parallel, distributed gradient descent involves a similar setup, but in federated learning communication is minimized by the parties each performing a number of iterations locally before sending their parameters or gradients to the aggregator. These local iterations complicate the theoretical convergence of the federated learning process, and for this reason most algorithms require all parties' data to be drawn as IID observations from a common data distribution. It has been observed that not only the theoretical convergence but also the practical convergence of the training is hindered in non-IID federated settings. Significant accuracy benefits can accrue from increasing the diversity of datasets used in model training; however, as participating parties bring more diverse data sets, federated learning training can suffer irreparably, leading in some cases to catastrophic failure Corinzia and Buhmann [2019]. Clearly, this has created problems in settings where federated learning can bring the most value.

*peng.qian.yu@ibm.com
†lwynter@sg.ibm.com
‡shonglim@sg.ibm.comt

We thus present a class of methods for federated learning which we call Fed+, pronounced FedPlus. The class of methods encompasses and unifies a number of recent algorithms proposed for federated learning, including several that allow for non-IID data. Importantly, Fed+ permits easily defining many new algorithms. The Fed+ paradigm allows for data heterogeneity by relaxing the requirement that the parties must reach a full consensus. The main idea behind Fed+ is to abandon the requirement that all parties converge to a single central point. Indeed, forcing diverse parties to solve for an average solution across all parties can lead to a failure of the training process. Instead, Fed+ advocates for a soft approach to federated learning: each party's training can be improved through cooperation with other parties. Fed+ applies equally well to traditional algorithms for federated learning, such as the original FedAvg McMahan *et al.* [2017], regularized algorithms such as FedProx Li *et al.* [2018], robust methods, such as Robust Federated Aggregation (RFA), which leverages the geometric median Pillutla *et al.* [2019] and the coordinate-wise median Yin *et al.* [2018], and other variants such as that of Smith *et al.* [2017]; Corinzia and Buhmann [2019].

To evaluate the performance of our proposed approach, it is important to assess how it and the alternatives fare when catastrophic failure happens such as that which occurs in real-world problems. To this end, we examine a reinforcement learning-based financial portfolio management problem that consists of sequentially allocating wealth to a collection of assets in consecutive trading periods. Due to the sequential decision-making nature of portfolio management, it is straightforward to apply reinforcement learning (RL) to model asset reallocation; see for example Almahdi and Yang [2017], model-free off-policy RL Jiang *et al.* [2017], an optimal hedging framework Buehler *et al.* [2019], and state-augmented RL Ye *et al.* [2020]. Federated learning is of significant potential value in this application for two reasons: (i) Historical financial data is limited. Consider the S&P500: the size of a training set for any S&P500 asset over the past 10 years is only 2530 observations, as there are 253 trading days per year. Traders thus augment public data using their own data models which they do not wish to share with other traders. Federated learning offers an avenue for them to jointly train policies on far more data, without revealing their private data. (ii) Second, financial markets are highly non-stationary and as such the policies learned from the training data may not generalize well due to distribution shift. Federated learning on heterogeneous data offers the benefits of multi-task learning in a privacy-protected manner. We illustrate the catastrophic collapse in the training of a federated RL policy using several different base fusion algorithms. All of the failures can be eliminated with the use of Fed+ in addition to the base algorithm. We further illustrate the Fed+ approach on the synthetic dataset created for FedProx by Li *et al.* [2018] as well as on a number of other common datasets that have been adapted for federated learning in the image and natural language domains.

2 Related Work

The original and most intuitive federated learning algorithm is FederatedAveraging (or FedAvg) McMahan *et al.* [2017]. However, its element-wise mean update is susceptible to corrupted values by faulty parties Tyler [2008]. Furthermore, it has been observed in practice that FedAvg can lead to failure of the training process. Recently, Li *et al.* [2020] showed that FedAvg as initially defined can converge to a point which is not a solution to the original problem; the authors proposed to add a learning rate that decreases at each federated round and with that modification they provide a theoretical convergence guarantee for the algorithm, even when the data is not IID. On the other hand, the resulting algorithm is very slow to converge and hence not efficient in practice.

A recent line of work involves enhancing the robustness of federated learning to corrupted updates from the parties. Pillutla *et al.* [2019] propose Robust Federated Aggregation (RFA), and argue that federated learning can be made robust to corrupted updates by replacing the weighted arithmetic mean aggregation by an approximate geometric median. In Yin *et al.* [2018], the authors propose a Byzantine-robust distributed statistical learning algorithm based on the coordinate-wise median of model weights, with a focus on achiev-

ing optimal statistical performance. Both robust federated learning algorithms, RFA Pillutla *et al.* [2019] and coordinate-wise median Yin *et al.* [2018], involve training a single global model on distributed data with assumptions on the maximum number of adversarial parties, usually that the number is less than one half. As we will see, such methods are not robust to non-IID data and can, as FedAvg, lead to a collapse of the learning process. However, both of these methods can be integrated into, and improved by, our Fed+ framework.

In an effort to better handle non-IID data, Li *et al.* [2018] introduce a regularization term in the form of a proximal distance from the FedAvg solution in their FedProx algorithm. Hanzely and Richtárik [2020] propose a local-global mixture method that overlaps with Fed+ in some of the settings considered and as such is subsumed by the Fed+ family of methods. Others like Li *et al.* [2020]; Karimireddy *et al.* [2019] seek to explain the non-convergence of FedAvg while proposing new algorithms: Pathak and Wainwright [2020]; Charles and Konecný [2020]; Malinovsky *et al.* [2020] offer explanation and new algorithms, called FedSplit and LocalUpdate, and Local Fixed Point, respectively, and obtain tight bounds on the number of communication rounds required to achieve an ϵ accuracy. These algorithms look similar to Fed+ but have an important difference in that they require the convergence of all parties to a common model.

Federated learning with heterogeneous data is similar to transfer learning Tirinzoni *et al.* [2018], which aims to transfer the experience gained in learning to perform one task to help improve learning performance in a related but different task. In this case, agents essentially perform different tasks since they train on non-IID data. Similarly, federated learning is related to multi-task learning and multi-agent learning, where each party may have limited historical data but as a group they have enough to train a deep neural network model. In transfer learning, however, it is assumed that observations are shared across tasks, while the federated setting does not allow for data to be shared across parties. Our proposed Fed+ can also be used in conjunction with multi-task federated learning Smith *et al.* [2017]; Corinzia and Buhmann [2019].

3 The Fed+ Framework

The Fed+ framework is designed to better handle federated learning when the data across the parties is not IID. The key observation is that Fed+ does not require all parties to converge to a single central point. Forcing diverse parties to solve for an average solution across all parties can lead to a failure of the training process. Instead, Fed+ advocates for a soft approach to federated learning: each party's training can be improved through cooperation with other parties while maintaining a unique local solution.

Fed+ thus requires redefining the overall objective of the federated learning training process. Consider the original FedAvg objective of minimizing the average loss over N parties.

$$\min_x F(x) := \frac{1}{N}\sum_{n=1}^{N} f_n(x) \tag{1}$$

where $x \in \Re^D$ is the model parameter to be learned, and $f_n : \Re^D \to \Re$ is the local loss function, for each n. Equivalently, it can be formulated as follows:

$$\min_{X=(x_1,x_2,\ldots,x_N)\in\Re^{ND}} F(X) := \frac{1}{N}\sum_{n=1}^{N} f_n(x_n)$$
$$\text{subject to} \quad x_1 = x_2 = \cdots = x_N.$$

Instead, we take a softer approach and propose the following objective for the overall federated training process:

$$\min_X F(X,\alpha) := \frac{1}{N}\sum_{n=1}^{N} f_n(x_n) + \alpha_n B(x_n, C(X)) \tag{2}$$

where $B(\cdot,\cdot)$ is a distance function, and $C : \Re^{ND} \to \Re^D$ is an aggregate function that computes a central point of $x_1, \ldots, x_N$. When $\alpha = 0$, problem (2) reduces to the non-federated setting where every party solves independently a local objective function. On the other hand, for $\alpha_n > 0$ for all n and $C(X) = \frac{1}{N}\sum_n x_n$, if one sets B such that $B(x, \hat{x}) = \infty$ if $x \neq \hat{x}$ and $B(x, \hat{x}) = 0$ otherwise, then (2) is equivalent to problem (1). More generally, the distance function B may be the usual Euclidean L2 norm, an Lp norm, or other Bregman divergence measures, such as a scaled proximal mapping

$$B(x_n, \hat{x}) := \frac{1}{2}\|x_n - \hat{x}\|_Q^2, \tag{3}$$

for a symmetric, positive definite matrix Q, where $\|x\|_Q = \sqrt{\langle x, x\rangle_Q}$ and $\langle x, y\rangle_Q = x'Qy$. The scaled proximal mapping of (3) can serve to weight each component's contribution differently depending on the application and information available during model training.

While problem (2) can be solved centrally, we are interested in the federated setting where each party n solves its own version of the problem, with learning rate γ^k and the following update:

$$\begin{aligned} x_n^{k+1} &= x_n^k - \gamma_n^k \nabla F_n(x_n^k), \\ &= x_n^k - \gamma^k [\nabla f_n(x_n^k) + \alpha_n^k \nabla B(x_n^k, C(X^k))]. \end{aligned} \tag{4}$$

In practice, one typically implements a stochastic gradient version of (4) where $\nabla f_n(x_n^k)$ is approximated by a random sample.

The Fed+ method is a family of algorithms for solving problem (2) by each party without requiring all parties to agree on a single, common model, defined in Algorithm 1. We argue that this softer approach to federated learning offers the benefits of federation without the pitfall of model failure that can occur in real-world implementations. The two main steps of the method are, after initializing values, (a) for the local parties, n, to update at round k their models, x_n^k, as a function of the central model value, $\hat{x}^k := C(X^k)$, as in (4), performing some number of such update steps within the round k and maintaining the last updated value as their current model, (b) for the parties to send this last updated value to the aggregator, who computes a new central value of the models over all participating parties, increments the round number, sends the new central value to the parties, and the process repeats from the last updated models at the parties. A main difference between Fed+ and most other algorithms is that the parties do not set the common model as their starting point at each successive round.

So as to encompass a number of important special cases, Algorithm 1 makes use of a number of parameters, α_n^k, γ_n^k and p_1, p_2, where p_1 controls the set of parties actively participating in each federated round and p_2 controls the number of local iterations at each party between federated updates. The Fed+ method is described for the case where parties transmit their models (e.g. in the form of neural network weights) to the aggregator; however, party gradients can be accommodated in a similar manner.

Possibly the simplest efficient form of Fed+ can be obtained by setting B to the L2 norm and α_n^k and p_2 to constants. We call this algorithm FedAvg+.

Definition 1 (New method, FedAvg+)**.** *Let $B(x_n, \hat{x}) := \frac{1}{2}\|x_n - \hat{x}\|^2$, $C(X) := \frac{1}{N}\sum_n x_n$, and set for all n, k, $\alpha_n^k = \alpha$ and p_2 to constants. We call this form of Algorithm 1, FedAvg+.*

The Fed+ method includes several knobs, the setting of which allows one to recover a number of existing algorithms, as well as to readily define new algorithms with varying properties, such as greater robustness.

Proposition 1 (FedProx method of Li *et al.* [2018])**.** *Let $B(x_n, \hat{x}) := \frac{1}{2}\|x_n - \hat{x}\|^2$, $C(X) := \frac{1}{N}\sum_n x_n$, $\alpha_n^k = \alpha$ for all $n = 1, \ldots, N$, $k = 1, \ldots, K$ and p_2 constants, and set current model at line 11 to $x_n^k \leftarrow \hat{x}$. Then Algorithm 1 reduces to FedProx Li* et al. *[2018].*

Algorithm 1 : Fed+

Initialization:

1: Define maximum number of federated rounds K for N parties, and parameters α_n^k, γ_n^k, $p_1 \in (0,1)$, $p_2 \in \mathcal{Z}$. Each party initializes and sends x_n^0 to the Aggregator, which computes the central value $\hat{x}^0 \leftarrow C(X^0)$.

Aggregator:

2: **for** round $k = 1, \ldots, K$ **do**
3: Sample parties n with p_1 to obtain set $\mathcal{P}^k \subseteq \{1, \ldots, N\}$.
4: **for** each party $n \in \mathcal{P}^k$ **in parallel do**
5: Aggregator sends $\hat{x}^{k-1}$ to party.
6: $x_n^k \leftarrow$ Local-Solve$(n, k, \hat{x}^{k-1})$.
7: Party sends x_n^k to Aggregator.
8: **end for**
9: Compute $\hat{x}^k \leftarrow C(X^k)$.
10: **end for**

Local-Solve (n,k,$\hat{x}$): //run on each active party n

11: Run $T = p_2$ local iterations, starting with current model $x_n^k \leftarrow x_n^{k-1}$.
12: **for** $t = 1, \ldots, T$ **do**
13: $x_n^k \leftarrow x_n^k - \gamma_n^k(\nabla f_n(x_n^k) + \alpha_n^k \nabla B(x_n^k, \hat{x}))$.
14: **end for**

Pathak and Wainwright [2020] revisit FedProx and show that it converges to a different fixed point than that of (1). Our aim is, as that of Hanzely and Richtárik [2020] discussed below, precisely this: for party n to converge to a fixed point that optimizes (2), a solution that solves party n's problem while benefitting in a softer fashion from the models of the other parties.

Remark 1 (Global-local mixture)**.** *Fed+ can be shown to resemble the local-global mixture algorithm of Hanzely and Richtárik [2020] when $B(x_n, \hat{x}) := \frac{1}{2}\|x_n - \hat{x}\|^2$, $\alpha_n^k = \alpha$ for all $n = 1, \ldots, N$, $k = 1, \ldots, K$ and the update step in line 13 of Algorithm 1 is modified. In the local-global mixture method, in each round, a random choice is made by the aggregator whether to perform a purely local update, or a gradient step with respect to B, which corresponds to moving all x_n^k towards $\hat{x}^k$. For consecutive local updates, this is equivalent to running Local-Solve in Algorithm 1 with $\alpha = 0$ with a random p_2.*

Note that when $B(x_n, \hat{x}) := \frac{1}{2}\|x_n - \hat{x}\|^2$, the update in line 13 of the Fed+ Algorithm has a particularly elegant and intuitive form. For a given party n, let $\gamma_n^k = \gamma_n$, $\alpha_n^k = \alpha_n$ for all $k = 1, \ldots, K$

$$\begin{aligned} x_n^k &\leftarrow x_n^k - \gamma_n[\nabla f_n(x_n^k) + \alpha_n \nabla B(x_n^k, \hat{x})], \\ &= x_n^k - \gamma_n[\nabla f_n(x_n^k) + \alpha_n(x_n^k - \hat{x})], \\ &= (1 - \hat{\alpha_n})x_n^k + \hat{\alpha_n}\hat{x} - \gamma_n \nabla f_n(x_n^k), \end{aligned}$$

where $\hat{\alpha}_n = \gamma_n \alpha_n$. The third term vanishes at the solution to each local problem x_n^* and so the update can be seen as moving along the line given by the $\hat{\alpha}_n$-convex combination of the local model x_n^k and the average model $\hat{x}$. This observation was also made by Hanzely and Richtárik [2020].

Remark 2 (Relation to model-agnostic meta-learning)**.** *Hanzely and Richtárik [2020], see also Charles and Konecný [2020], note that party n's update obtained by subtracting the gradient of the loss function from the average local model resembles MAML Finn* et al. *[2017], which updates the party's model by subtracting the gradient of the global model.*

Federated algorithms which are robust to adversarial attacks are of interest and have motivated the development of several robust versions of federated aggregation such as the use of the geometric median Pillutla *et al.* [2019] and that of the coordinate-wise median Yin *et al.* [2018]. The Fed+ framework can be shown to reduce to those algorithms as follows.

Proposition 2 (Robust 1: Geometric median)**.** *Let $\alpha_n^k = 0$ for all $k = 1, \ldots, K$ and $n = 1, \ldots, N$, and C be the geometric median function*

$$C(X) := \arg\min_z \sum_{n=1}^{N} \|z - x_n\|.$$

Set the current model at line 11 to $x_n^k \leftarrow \hat{x}$. Then, the algorithm is equivalent to the RFA method of Pillutla et al. *[2019].*

Proposition 3 (Robust 2: Coordinate-wise median)**.** *Let $\alpha_n^k = 0$ for all $k = 1, \ldots, K$ and $n = 1, \ldots, N$, and C be the coordinate-wise median such that for each $d = 1, \ldots, D$,*

$$C(X)_d := \texttt{med}\{x_{1,d}, \ldots, x_{N,d}\},$$

where the median of each component of x is computed independently. Set the current model at line 11 to $x_n^k \leftarrow \hat{x}$. Then, Fed+ is equivalent to the algorithm of Yin et al. *[2018], that uses coordinate-wise median aggregation and is faster to compute than the RFA method above.*

While these forms of median are more robust to outliers than the mean used in (1), median-based methods are not immune to training failure as a result of non-IID party-level data as we shall illustrate in the experimental section of this paper. As such, it is of interest to use the Fed+ framework in robust aggregation. One can readily obtain a new family of robust algorithms by setting $\alpha_n^k > 0$, using a robust measure of centrality C such as the geometric or coordinate-wise median, and using Line 11 as defined in Algorithm 1.

Remark 3. *(Convergence of Fed+) A convergence proof of Fed+ can be derived under appropriate conditions when C is given by the mean, and B is the standard Euclidean distance. However, when using other forms of C, the overall function $F(X) = 1/N \sum_n f_n(x_n) + \alpha B(x_n, C(X))$ need not exhibit the required properties due to the second argument of $B(x_n, C(X))$. Indeed, the overall mapping of each party's function $F_n(x_n, C(X))$ need not be well-defined, even when $f_n(x_n)$ is strongly convex and $B(x_n, C(X))$ is strictly convex in its first argument. Consider for example the geometric median, in which case $C(X) = \arg\min_z \sum_{n=1}^{N} \|z - x_n\|$, and coordinate-wise median, which is non-smooth. As such, we leave a thorough analysis of Fed+ convergence to future work.*

In the next section, we show numerically that Fed+ works remarkably well in practice not only when $B(x_n, C(X))$ is continuously differentiable using C as given by the mean, such in our FedAvg+ instantiation, but also the robust yet non-smooth variants of Fed+: namely RFA+ and coordinate-wise median+, which offer far superior performance to the original baseline algorithms, FedAvg, RFA, and coordinate-wise median.

4 Experiments

We begin by illustrating the type of catastrophic failure that can occur in real-world federated learning training on a reinforcement learning-based financial portfolio management problem that sequentially allocates wealth to assets over time. The problem is explained further in the next section. The key observation, shown in Figure 1, is that replacing the local party models with the common, central (e.g. average) model can lead

to dramatic spikes in model changes. These dramatic spikes can trigger training failure for the federation as a whole. Specifically, the figure shows the mean and standard deviation of the change in neural network parameter values before and after a federated learning aggregation step. Three federated aggregation methods illustrated are: FedAvg, RFA using the geometric median, and the coordinate-wise median as well as the no fusion case where each party trains independently on its own data , and the FedAvg+ version of our proposed Fed+ approach.

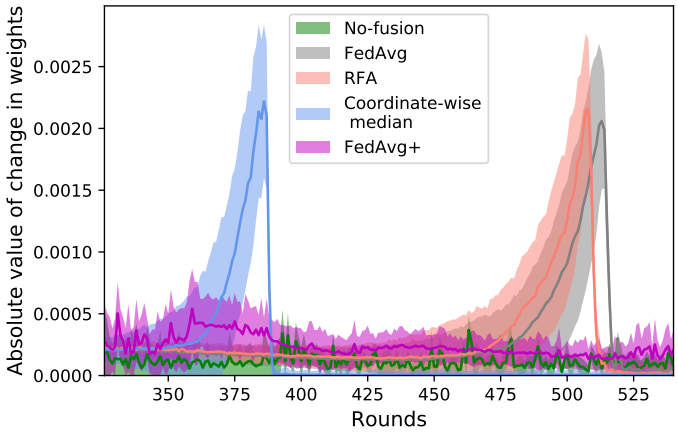

Figure 1: Impact of federated learning aggregation on consecutive model changes. The figure shows the absolute value of the model change (here the model is the neural network parameters) before and after federated model aggregation on the financial portfolio optimization problem. FedAvg, RFA and coordinate-wise median cause large spikes in the parameter change that do not occur without federated learning or when using Fed+. The large spikes coincide precisely with training collapse, shown in Figure 2 (bottom three figures).

Such dramatic model change can lead to a collapse of the training process. Figure 2 (three bottom curves) shows the collapse of training, averaged over parties, using FedAvg, RFA and coordinate-wise median. Note that this example *does not involve adversarial parties or party failure in any way*, as evidenced from the fact that the single-party training on the same dataset (top curve) does not suffer any failure. Rather, this is an example of federated learning on a real-world application where parties' data are not IID from a single dataset. As such, it is conceivable that federated model failure would be a relatively common occurrence in real-world applications using the vast majority of algorithms.

Next, Figure 3 motivates the Fed+ approach by illustrating the behavior of the algorithms as a function of how close the local party moves from a purely local model towards a common, central model. A local party update occurs in each subplot on the left side, at $\alpha = 0$. Observe that the local updates improve the performance from the previous aggregation indicated by the dashed lines. However, performance degrades after the subsequent aggregation, corresponding to the right-hand side of each subplot, where $\alpha = 1$. In fact, for FedAvg and RFA, performance of the subsequent aggregation is worse than the previous value (dashed line). Intermediate values of α correspond to moving towards, but not reaching, the common, central model.

4.1 Standard Federated Datasets

We evaluate Fed+ on standard federated learning benchmarks including the non-identical synthetic dataset of Li *et al.* [2018], a convex classification problem with MNIST LeCun *et al.* [1998] and FEMNIST Cohen *et al.* [2017]; Caldas *et al.* [2018]; Li *et al.* [2018] and a non-convex text sentiment analysis task called

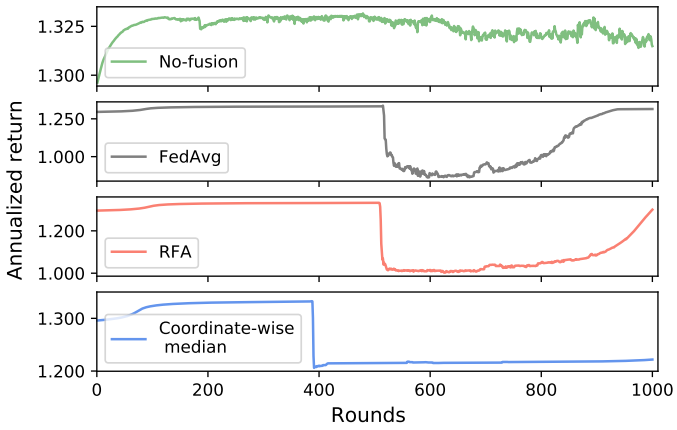

Figure 2: Illustration of training collapse. Average training performance over 50 parties of the financial portfolio optimization problem. The top curve is the average of 50 single-party training processes, each party on its own data. The training processes collapse using all 3 federated learning algorithms. Note that there are no adversarial parties in the federation, nor are there any party-level failures across the 50 parties. This can be verified from the top curve that does not experience training collapse. Rather, the training collapse is due to the federated learning process itself, and occurs when party-level training is forced to concur with a single, central model. A deeper understanding of this collapse can be gleaned from Figure 3, which shows what happens to the training before and after an aggregation step.

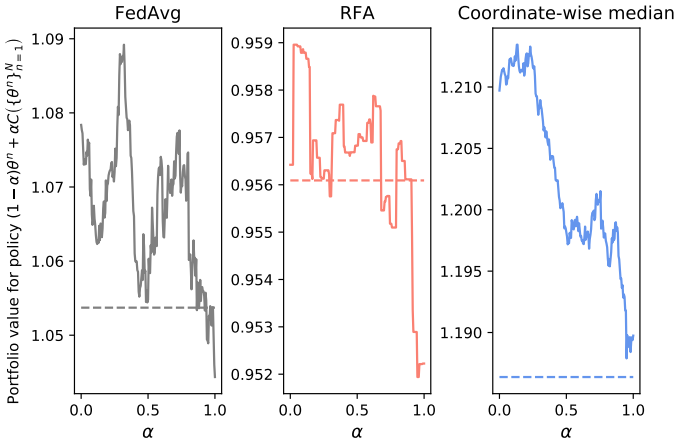

Figure 3: Before and after aggregation, along the line given by varying $\alpha \in [0, 1]$ using a convex combination of local update and the common model. Performance on the financial portfolio optimization problem is shown as a function of locally shifting α. Dashed lines represent the common model at the previous round. The right-hand side lower than the left-hand means that a full step towards averaging (or median) all parties, i.e., $\alpha = 1$, degrades local performance. This is the case with the standard FedAvg as well as with the robust methods.

Sentiment140 (Sent140) Go *et al.* [2009]. The descriptions of these benchmarks and the way in which heterogeneity was imposed can be found in the supplementary material. The models of benchmarks are the

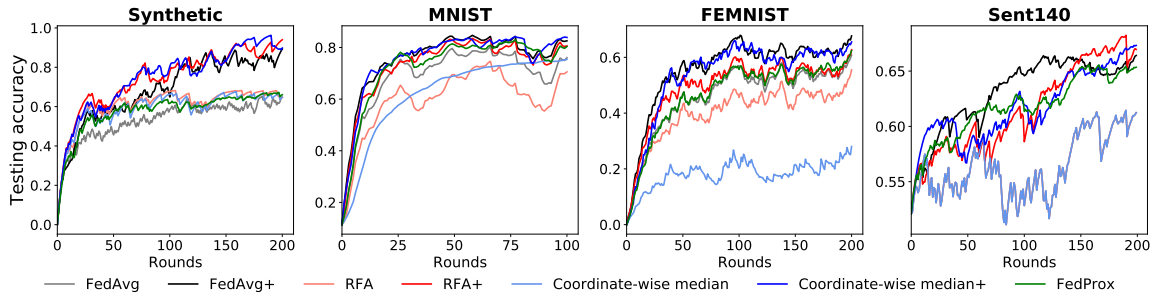

Figure 4: Performance of Fed+, i.e., FedAvg+, RFA+ and coordinate-wise median+, is far superior to that of the baselines.

same as those of Li *et al.* [2018] and the best μ reported in FedProx is used. In Figure 4, we report the testing performance for baselines algorithms FedAvg, FedProx, RFA and coordinate-wise median together with our proposed Fed+ algorithms. In all cases shown here, $\alpha_n^k = \alpha$ for all $n = 1, \ldots, N$ and $k = 1, \ldots, K$, where the constant value was tuned as a hyperparameter. In the supplementary material, we provide results for Fed+ with decaying α^k, where that version is of interest when the tuning of α would be problematic.

Overall, on the benchmark problems, the robust federated learning algorithms perform the worst on these non-IID data sets. FedProx uses a proximal term which, as we know from Li *et al.* [2018], is beneficial in heterogeneous settings as compared with FedAvg. However, all of the baselines produce a single global model not specific to the parties. Not only does Fed+ improve performance, it often also speeds up the learning convergence, as shown in Figure 4. The best Fed+ algorithm can improve the FedProx's performance on these four data sets by 9.82% on average. The robust variants of Fed+, namely RFA+ and coordinate-wise median+, outperform FedAvg+ on the synthetic, MNIST and Sent140 datasets, respectively. This shows the benefits of incorporating robust statistics such as the geometric median and coordinate-wise median rather than using the average, as proposed in the global-local mixture method of Hanzely and Richtárik [2020].

4.2 Financial Portfolio Management

Financial portfolio management is a process of sequentially allocating wealth to a collection of assets in consecutive trading periods Markowitz [1959]; Haugen and Haugen [2001]. Let t denote the index of asset trading days, $\boldsymbol{y}_t$ denote price relative vector, $\boldsymbol{Y}_t$ denote price matrix and $\boldsymbol{w}_t$ be portfolio weights. Details of the portfolio management problem as well as necessary definitions are provided in the supplementary material. The reinforcement learning model of financial portfolio management is, at time step t, to observe the *state* $s_t \triangleq (\boldsymbol{Y}_t, \boldsymbol{w}_{t-1})$, take an *action* (choose portfolio weights) $a_t = \boldsymbol{w}_t$ and receive an immediate *reward* $r(s_t, a_t) \triangleq \ln(\beta_t \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1})$ where β_t is a transaction factor. The RL policy μ_θ parameterized by deep neural networks with weights θ involves maximizing the return of the portfolio: $\max_{\mu_\theta} J(\mu_\theta) \triangleq \max_{\mu_\theta} \sum_{t=1}^{T} \ln(\beta_t \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1})/T$. By the deterministic policy gradient theorem Silver *et al.* [2014], the optimal μ_θ can be found from $\theta \leftarrow \theta + \lambda \nabla_\theta J(\mu_\theta)$ where λ is the learning rate. Historical financial data is limited. The size of a training set for any S&P500 asset over the past 10 years is only 2530 observations, as there are 253 trading days per year. Traders thus each build private models to generate more training examples; we use Geometric Brownian Motion (GBM), Variable-order Markov (VOM) model and Generative Adversarial Network (GAN). Further details of the RL model itself, the architecture and the data generative models can be found in the supplementary material.

Given N parties, each with its own trading asset universe and data $\mathcal{I}_n$ for $n \in \{1, 2, \ldots, N\}$. The number of assets in each universe $\mathcal{I}_n$ is the same. The experiments are performed on 30 assets from the S&P500

technology sector. Each party n constructs its own asset universe $\mathcal{I}_n$ by randomly choosing 9 assets and pre-trains a private data augmentation method, i.e., GBM, VOM, or GAN, on their assets in $\mathcal{I}_n$. S&P price data from 2006–2018 is used for training data augmentation and the RL policies; 2019 price data is used for testing.

All parties participate in the training, i.e., $\mathcal{P}^k = \{1, \ldots, N\}$. The number of global rounds $K = 1000$ and local RL iterations $T = 50$. Results are based on training over a sufficiently wide grid of fixed α (typically 10-13 values on a multiplicative grid of resolution 10^{-1} or 10^{-3}). Results are for the best fixed α selected individually for each experiment. Two metrics measure performance: the most intuitive is the geometric average return earned by an investment each year over a given period, i.e., annualized return. To take into account risk and volatility, we also report the Sharpe ratio Sharpe [1966], which in its simplest form is $\mathbb{E}[X]/\sqrt{\text{var}[X]}$ where X is the (random) return of a portfolio. The Sharpe ratio is thus the additional return an investor receives per unit of increase in risk.

Results are presented for a 10 party federation, each party with a different asset universe. Results show that FedAvg degrades the performance of all parties due largely to training collapse. The robust methods, RFA and coordinate-wise median, achieve stable learning and yield better performance in the 10-party federation, provided in the supplementary material. Average performance with 50 parties is shown in Table 1 with average learning curves in Figure 2 and Figure 6. In this larger federation, both robust federated learning methods, RFA and coordinate-wise median, experience training collapse. This suggests larger distribution mismatch across the 50 asset universes.

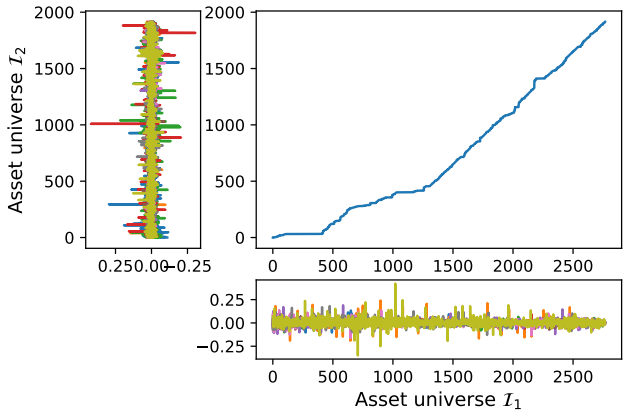

Figure 5: Dynamic time warping/alignment curve for the data of the financial portfolio management problem, shown for returns of asset universes $\mathcal{I}_1$ and $\mathcal{I}_2$. Note that two multi-variate time-series have different length. The graphic shows that the distance between data distributions given by the asset universes increases with the length of the time series, up to 2500 in this figure, leading to heterogeneity across parties.

Data distribution similarity between each pair of universes (each asset universe is a multi-variate time-series) was measured using Dynamic Time Warping (DTW) Berndt and Clifford [1994]. Unlike the Euclidean distance, DTW compares time series of variable size and is robust to shifts or dilatations across time. The averaged pairwise distances for the 50 asset universes is 154.51, which is 21.63% larger than that of the 10 asset universes. The DTW warping/alignment curve for asset universes $\mathcal{I}_1$ and $\mathcal{I}_2$ are illustrated in Figure 5.

Performance across baseline methods is shown in Figure 2 where the failure of the standard methods, FedAvg, RFA and coordinate-wise median, is clear. The parameter change before and after federated model aggregation is shown in Figure 1. Observe that there are large spikes at the point of collapse of FedAvg,

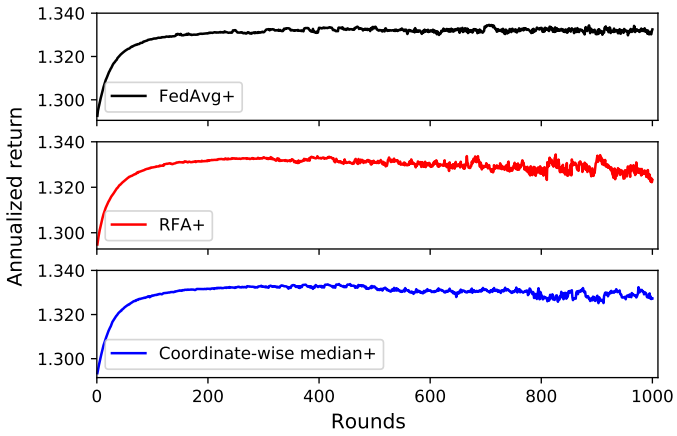

Figure 6: Average learning performance of Fed+ algorithms on the financial portfolio optimization problem with 50 parties. Compare with Figure 2 taken from the same problem using the baseline algorithms. No training collapse occurs with any of the Fed+ algorithm variants.

Table 1: Average performance of different methods over 50 parties.

Method	Annualized return	Sharpe ratio
FedAvg	29.37%	1.53
RFA	29.62%	1.62
Coordinate-wise median	22.12%	1.20
FedAvg+	32.86%	1.66
RFA+	31.95%	1.62
Coordinate-wise median+	32.35%	1.63

RFA and coordinate-wise median, implying that single-model methods may not work well in heterogeneous environments. Fed+, on the other hand, stabilizes each party's learning process and is robust, as shown in Figure 6. Table 1 further shows that Fed+ and our variant algorithms outperform FedAvg, RFA and coordinate-wise median, improving the annualized return by 5.35% and the Sharpe ratio by 0.19 on average. This improvement implies that each party benefits from sharing model parameters using Fed+, whereas the benefits were not seen with FedAvg, RFA and coordinate-wise median.

For the best performing variant of Fed+ on this application, FedAvg+, we further investigate how the risk-return performance improves as more parties share their local tasks. In particular, compared with single-party, no-federated-learning (no-fusion) training, the maximum improvements in annualized return are 5.56%, 13.53% and 34.54%, and in Sharpe ratio are 0.27, 0.66 and 1.63 when $N = 5, 10$ and 50. This demonstrates that the benefit of using Fed+ increases as the number of parties increases.

5 Conclusion

We introduce a novel federated learning framework called Fed+ designed to better handle the statistical heterogeneity inherent in federated settings. The class of methods encompasses and unifies a number of recent

algorithms proposed for federated learning, including several that allow for non-IID data and permits easily defining many new algorithms. The Fed+ paradigm relaxes the requirement that the parties must reach a full consensus at a single central point. Indeed, as we have shown in a real-world setting, forcing diverse parties to solve for an average solution across all parties can lead to a failure of the training process. Instead, Fed+ advocates for a soft approach to federated learning: each party's training can be improved through cooperation with other parties. Fed+ avoids the training collapse and learning failure that can occur with standard methods, including robust algorithms. Empirically, we demonstrate that Fed+ and its derivative algorithms, that we call FedAvg+, RFA+, and coordinate-wise median+, achieve significantly better performance on benchmark datasets and on a challenging, real-world financial portfolio management problem.

References

Saud Almahdi and Steve Y Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, 2017.

Ron Begleiter, Ran El-Yaniv, and Golan Yona. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

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

Zachary Charles and Jakub Konecnỳ. On the outsized importance of learning rates in local update methods. *ArXiv*, abs/2007.00878, 2020.

Long Cheng, Fang Liu, and Danfeng Yao. Enterprise data breach: causes, challenges, prevention, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(5):e1211, 2017.

I Glenn Cohen and Michelle M Mello. HIPAA and protecting health information in the 21st century. *Jama*, 320(3):231–232, 2018.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *ArXiv*, abs/2002.05516, 2020.

Robert A Haugen and Robert A Haugen. *Modern investment theory*, volume 5. Prentice Hall Upper Saddle River, NJ, 2001.

Chris Jay Hoofnagle, Bart van der Sloot, and Frederik Zuiderveen Borgesius. The European Union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98, 2019.

Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

Niaz Kammoun, Ahmed Bounfour, Altay Özaygen, and Rokhaya Dieye. Financial market reaction to cyber-attacks. *Cogent Economics & Finance*, 7(1):1645584, 2019.

Sai Praneeth Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. SCAFFOLD: Stochastic controlled averaging for on-device federated learning. *ArXiv*, abs/1910.06378, 2019.

Takeaki Kariya and Regina Y Liu. Options, futures and other derivatives. In *Asset Pricing*, pages 9–26. Springer, 2003.

Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127, AMTL Workshop 2019*, 2018.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of FedAvg on non-iid data. volume Arxiv, abs/1907.02189 of *ICLR*, 2020.

G. Malinovsky, D. Kovalev, E. Gasanov, Laurent Condat, and Peter Richtárik. From local SGD to local fixed point methods for federated learning. *ICML*, Arxiv, abs/2004.01442, 2020.

Harry Markowitz. *Portfolio selection: Efficient diversification of investments*, volume 16. John Wiley New York, 1959.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

Reese Pathak and M. Wainwright. FedSplit: An algorithmic framework for fast federated optimization. *ArXiv*, abs/2005.05238, 2020.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

William F Sharpe. Mutual fund performance. *The Journal of business*, 39(1):119–138, 1966.

Joan Sieber. Family Educational Rights and Privacy Act (FERPA). *Journal of Empirical Research on Human Research Ethics*, 2(1):101–101, 2007.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML14, page I387I395. JMLR.org, 2014.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Andrea Tirinzoni, Rafael Rodriguez Sanchez, and Marcello Restelli. Transfer of value functions via variational methods. In *Advances in Neural Information Processing Systems*, pages 6179–6189, 2018.

David E Tyler. Robust statistics: Theory and methods, 2008.

Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1112–1119, 2020.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. volume 80 of *Proceedings of Machine Learning Research*, pages 5650–5659, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

6 Appendix

7 The Fed+ Framework Proofs

Proposition 1 (FedProx method of Li *et al.* [2018])**.** *Let* $B(x_n, \hat{x}) := \frac{1}{2}\|x_n - \hat{x}\|^2$, $C(X) := \frac{1}{N}\sum_n x_n$, $\alpha_n^k = \alpha$ *for all* $n = 1, \ldots, N$, $k = 1, \ldots, K$ *and* p_2 *constants, and set current model at line 11 to* $x_n^k \leftarrow \hat{x}$. *Then Algorithm 1 reduces to FedProx Li* et al. *[2018].*

Proof. The results immediately follow since $B(x_n, \hat{x})$ corresponds to the proximal term, and $C(X)$ is the same as the server aggregation in Li *et al.* [2018]. □

Proposition 2 (Robust 1: Geometric median)**.** *Let* $\alpha_n^k = 0$ *for all* $k = 1, \ldots, K$ *and* $n = 1, \ldots, N$, *and* C *be the geometric median function*

$$C(X) := \arg\min_z \sum_{n=1}^{N} \|z - x_n\|.$$

Set the current model at line 11 to $x_n^k \leftarrow \hat{x}$. *Then, the algorithm is equivalent to the RFA method of Pillutla* et al. *[2019].*

Proof. Note that the overall federated training process reduces to the non-federated setting where every party solves independently a local objective function when $\alpha_n^k = 0$. The results then follow by the definition of RFA method in Pillutla *et al.* [2019]. □

Proposition 3 (Robust 2: Coordinate-wise median)**.** *Let* $\alpha_n^k = 0$ *for all* $k = 1, \ldots, K$ *and* $n = 1, \ldots, N$, *and* C *be the coordinate-wise median such that for each* $d = 1, \ldots, D$,

$$C(X)_d := \texttt{med}\{x_{1,d}, \ldots, x_{N,d}\},$$

where the median of each component of x *is computed independently. Set the current model at line 11 to* $x_n^k \leftarrow \hat{x}$. *Then, Fed+ is equivalent to the algorithm of Yin* et al. *[2018], that uses coordinate-wise median aggregation and is faster to compute than the RFA method above.*

Proof. Note that the overall federated training process reduces to the non-federated setting where every party solves independently a local objective function when $\alpha_n^k = 0$. The results then follow by the definition of coordinate-wise median aggregation method in Yin *et al.* [2018]. □

8 Standard Federated Datasets

8.1 Additional Details on the Test Problems

We evaluate Fed+ on standard federated learning benchmarks including the non-identical synthetic dataset of Li *et al.* [2018], a convex classification problem with MNIST LeCun *et al.* [1998] and FEMNIST Cohen *et al.* [2017]; Caldas *et al.* [2018]; Li *et al.* [2018] and a non-convex text sentiment analysis task called Sentiment140 (Sent140) Go *et al.* [2009]. Hyperparameters are the same as those of Li *et al.* [2018] and use the best μ reported in FedProx. Data is randomly split for each local party into an 80% training set and a 20% testing set. The number of selected parties per round is 10 for all experiments on all datasets. Learning rates are 0.01, 0.03, 0.003 and 0.3 for synthetic, MNIST and FEMNIST and Sent140 datasets, respectively. The experiments used a fixed $\alpha = 0.001$. In some cases it may be of interest to use a decaying α. Results in that

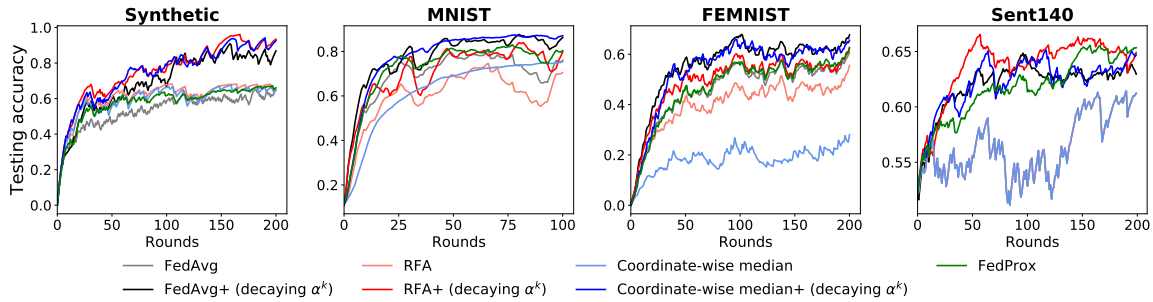

Figure 7: Results on standard test problems, illustrating Fed+ with constant as well as with decaying α. While constant α, when properly tuned, is always preferable, decaying α provides good results (e.g. when tuning of α is impractical) above and beyond that which the baseline algorithms can provide. The only exception is in the Sent140 example, where FedProx is competitive.

setting are produced by using an initial $\alpha^1 = 0.001$ and setting $\alpha^{k+1} = 0.99\alpha^k$. We simulate the federated learning setup (1 aggregator N parties) on a commodity machine with 16 Intel® Xeon® E5-2690 v4 CPUs and 2 NVIDIA® Tesla P100 PCIe GPUs.

To generate non-identical synthetic data, we follow a similar setup to that in Li *et al.* [2018], additionally imposing heterogeneity among parties. In particular, for each party n, we generate samples (X_n, Y_n) according to the model $y = \arg\max(\text{softmax}(Wx + b))$, $x \in \mathbb{R}^{60}, W \in \mathbb{R}^{10\times 60}, b \in \mathbb{R}^{10}$. We model $W_n \sim \mathcal{N}(u_n, 1)$, $b_n \sim \mathcal{N}(u_n, 1)$, $u_n \sim \mathcal{N}(0, \zeta)$; $x_n \sim \mathcal{N}(v_n, \Sigma)$, where the covariance matrix Σ is diagonal with $\Sigma_{j,j} = j^{-1.2}$. Each element in the mean vector v_n is drawn from $\mathcal{N}(B_n, 1), B_n \sim N(0, \beta)$. Therefore, ζ controls how much local models differ from each other and β controls how much the local data at each party differs from that of other parties. In order to better characterize statistical heterogeneity and study its effect on convergence, we choose $\zeta = 1000$ and $\beta = 10$. Our goal is to learn a global W and b. There are $N = 30$ parties in total and the number of samples on each party follows a power law.

Three real datasets curated from prior work in federated learning are tested McMahan *et al.* [2017]; Caldas *et al.* [2018]. First, we consider a convex classification problem using MNIST LeCun *et al.* [1998] with multinomial logistic regression. To impose statistical heterogeneity, we distribute the data among N =1,000 parties such that each party has samples of only one digit and the number of samples per party follows a power law. The input of the model is a flattened 784-dimensional (28×28) image, and the output is a class label between 0 and 9.

We then study a more complex 62-class Federated Extended MNIST Cohen *et al.* [2017]; Caldas *et al.* [2018] (FEMNIST) dataset proposed in Li *et al.* [2018] using the same model. The heterogeneous data partitions in FEMNIST are generated by subsampling 10 lower case characters ('a'-'j') from EMNIST dataset Cohen *et al.* [2017] and distributing only 5 classes to each party. There are N =200 parties in total. The input of the model is a flattened 784-dimensional (28×28) image, and the output is a class label between 0 and 9.

To address non-convex settings, we consider a text sentiment analysis task on tweets from Sentiment140 Go *et al.* [2009] (Sent140) with a two layer LSTM binary classifier containing 256 hidden units with pretrained 300D GloVe embedding [Pennington *et al.*, 2014]. There are N =772 parties in total. Each twitter account corresponds to a party. The model takes as input a sequence of 25 characters, embeds each of the characters into a 300-dimensional space by looking up Glove and outputs one character per training sample after 2 LSTM layers and a densely-connected layer. We consider the highly heterogeneous setting where there are 90% stragglers (see Li *et al.* [2018] for more details).

8.2 Additional Results

In general, one would set α_n^k to a constant, which may be party n-dependent or may be identical across parties. Indeed, the best results are obtained in this setting, though it is to be noted that the choice of α in this setting was tuned as a hyperparameter.

In some cases, it is of interest to allow the parameters α_n^k to decay. In this case, the parties eventually solve near-pure local problems. Results are presented in Figure 7 for the standard test problems. Observe from the figure that decaying α is never preferable to a constant α in these experiments. However, decaying α may be seen as a way to hedge against imprecise tuning of the parameter. Nonetheless, the performance of Fed+ with a decaying α is for the most part superior to that of the baseline algorithms which do not use Fed+, with the sole exception of the Sent140 dataset.

9 Financial Portfolio Management Problem

We start with an overview of the necessary background on reinforcement learning, our notation and definitions. Then we present the model itself, the data augmentation methods, and the experimental setup in detail.

9.1 Reinforcement Learning

In reinforcement learning (RL), the goal is to learn a policy to control a system modeled by a Markov decision process which is defined as a 6-tuple $\langle \mathcal{S}, \mathcal{A}, P, r, T, \gamma \rangle$. Here, $\mathcal{S} = \bigcup_t \mathcal{S}_t$ is the state space and $\mathcal{A} = \bigcup_t \mathcal{A}_t$ is the action space, both assumed to be finite dimensional and continuous; $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition kernel and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; T is the (possibly infinite) decision horizon; $\gamma \in (0, 1]$ is the discount factor. Deterministic policy gradient learns a deterministic target policy using deep neural networks with weights θ Silver *et al.* [2014]. A policy parameterized by θ is a mapping $\mu_\theta : \mathcal{S} \to \mathcal{A}$, specifying the action to choose in a particular state. At each time step $t \in \{1, \ldots, T\}$, the agent in state $s_t \in \mathcal{S}_t$ takes a deterministic action $a_t = \mu_\theta(s_t) \in \mathcal{A}_t$, receives the reward $r(s_t, a_t)$ and transits to the next state s_{t+1} according to P.

An RL agent's objective is to maximize its expected return given the start distribution

$$J(\mu_\theta) \triangleq \mathbb{E}_{s_t \sim P}[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, \mu_\theta(s_t))].$$

The optimal policy to achieve the objective can be found by applying the deterministic policy gradient theorem (see Theorem 1 in Silver *et al.* [2014]), where the idea is to adjust the parameters θ of the policy in the direction of the performance gradients $\nabla_\theta J(\mu_\theta)$. The deterministic policy gradients can be estimated more efficiently than their stochastic counterparts Sutton *et al.* [2000], avoiding a problematic integral over the action space.

9.2 Portfolio Optimization Model

Assume the financial market is sufficiently liquid such that any transactions can be executed immediately with minimal market impact. Following Jiang *et al.* [2017], let t denote the index of asset trading days and $v_{i,t}$, $i = \{1, \ldots, \eta\}$ the closing price of the i^{th} asset at time t, where η is the number of assets in a given asset universe. The price vector $\boldsymbol{v}_t$ consists of the closing prices of all n assets. An additional dimension (the first dimension indexed by 0) in $\boldsymbol{v}_t$, $v_{0,t}$, denotes the cash price at time t. We normalize all temporal

variations in $\boldsymbol{v}_t$ with respect to cash so $v_{0,t}$ is constant for all t. Define the price relative vector at time t as $\boldsymbol{y}_t \triangleq \boldsymbol{v}_{t+1} \oslash \boldsymbol{v}_t = (1, v_{1,t+1}/v_{1,t}, \ldots, v_{\eta,t+1}/v_{\eta,t})^\top$ where $\oslash$ denotes element-wise division.

Define $\boldsymbol{w}_{t-1}$ as the portfolio weight vector at the beginning of time t where its i^{th} element $w_{i,t-1}$ represents the proportion of asset i in the portfolio after capital reallocation and $\sum_{i=0}^{\eta} w_{i,t} = 1$ for all t. The portfolio is initialized with $\boldsymbol{w}_0 = (1, 0, \ldots, 0)^\top$. At the end of time t, the weights evolve according to $\boldsymbol{w}'_t = (\boldsymbol{y}_t \odot \boldsymbol{w}_{t-1})/(\boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1})$, where $\odot$ is element-wise. The reallocation from $\boldsymbol{w}'_t$ to $\boldsymbol{w}_t$ is gotten by selling and buying relevant assets. Paying all fees, this reallocation shrinks the portfolio value by $\beta_t \triangleq c \sum_{i=1}^{\eta} |w'_{i,t} - w_{i,t}|$ where $c = 0.2\%$ is the buy/sell fee; let ρ_{t-1} denote the portfolio value at the beginning of t and ρ'_t at the end, so $\rho_t = \beta_t \rho'_t$.

The normalized close price matrix at t is $\boldsymbol{Y}_t \triangleq [\boldsymbol{v}_{t-l+1} \oslash \boldsymbol{v}_t | \boldsymbol{v}_{t-l+2} \oslash \boldsymbol{v}_t | \cdots | \boldsymbol{v}_{t-1} \oslash \boldsymbol{v}_t | \mathbf{1}]$ where $\mathbf{1} \triangleq (1, 1, \ldots, 1)^\top$ and l is the time embedding. The financial portfolio management can then be formulated as a RL problem where, at time step t, the agent observes the *state* $s_t \triangleq (\boldsymbol{Y}_t, \boldsymbol{w}_{t-1})$, takes an *action* (portfolio weights) $a_t = \boldsymbol{w}_t$ and receives an immediate *reward* $r(s_t, a_t) \triangleq \ln(\rho_t/\rho_{t-1}) = \ln(\beta_t \rho'_t/\rho_{t-1}) = \ln(\beta_t \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1})$. Considering the policy μ_θ, the objective of RL agent is to maximize an objective function parameterized by θ: $\max_{\mu_\theta} J(\mu_\theta) = \max_{\mu_\theta} \sum_{t=1}^{T} \ln(\beta_t \boldsymbol{y}_t \cdot \boldsymbol{w}_{t-1})/T$. By deterministic policy gradient theorem Silver *et al.* [2014], the optimal μ_θ can be found via the following update rule for parameters θ:

$$\theta \leftarrow \theta + \lambda \nabla_\theta J(\mu_\theta) \tag{5}$$

where λ is the learning rate.

Suppose now that there are N participants (parties), each with an RL agent performing portfolio management. Each party has its trading asset universe (data) $\mathcal{I}_n$ for $n \in \{1, 2, \ldots, N\}$, and the cardinality (number of assets) in every asset universe $\mathcal{I}_n$ is the same. The goal is to aggregate each party's model such that it can benefit from others. Since each party is trading on various asset universes, this multi-agent portfolio management problem is essentially a multi-task problem. In particular, each party's task is a Markov decision processes $\langle \mathcal{S}_n, \mathcal{A}, P_n, r, T, \gamma \rangle$ where the action space $\mathcal{A}$ and the reward function r are common to all parties. The state space $\mathcal{S}_n$ depends on each party's trading asset universe $\mathcal{I}_n$ and the transition kernel P_n should be inferred from the underlying stock price dynamics of $\mathcal{I}_n$. Each party is maximizing its objective and updating its RL agent using the update rule (5). In addition, each party shares its RL model updates through federated learning and receives an updated model that can improve its performance.

9.3 Data Augmentation Methods

Three types of generative models are used to enrich and distinct each party's data set for training the RL agents. The parameters of each model and the data generated are private and local to each party.

Geometric Brownian Motion (GBM) is a continuous-time stochastic process in which the logarithm of the randomly varying quantity follows a Brownian motion with drift Kariya and Liu [2003]. GBM is often used in mathematical finance to model stock prices in the BlackScholes model Black and Scholes [1973] mainly because the expected returns of GBM are independent of the values of the stock price, which agrees with what we expect in reality Kariya and Liu [2003]. In addition, a GBM process shows the same kind of "roughness" in its paths as we often observe in real stock prices. The close price of asset i follows a GBM if it satisfies the following stochastic differential equation: $dv_{i,t} = \mu_i v_{i,t} dt + \sigma_i v_{i,t} dW_t$. Here W_t is a Brownian motion, μ_i is the mean return of the stock prices given a historical date range and σ_i is the standard deviation of returns of the stock prices in the same date range. The differential equation can be solved by an analytic solution: $v_{i,t} = v_{i,0} \exp((\mu_i - \sigma_i^2/2)t + \sigma_i W_t)$. We choose the initial price value $v_{i,0}$ to be the last day's close price of the asset in the training set, and we use the asset returns in the date range of RL training set to estimate μ_i and σ_i.

Variable-order Markov (VOM) models are an important class of models that extend the well known Markov chain models Begleiter *et al.* [2004], where each random variable in a sequence with a Markov property

depends on a fixed number of random variables. In contrast, in VOM models this number of conditioning random variables may vary based on the specific observed realization. Given a sequence of returns $\{\boldsymbol{p}_t\}_{t=0}^{T}$ where $p_{i,t} = (v_{i,t+1} - v_{i,t})/v_{i,t}$ for asset i in asset universe $\mathcal{I}$ at time t, the VOM model learns a model $\mathbb{P}$ that provides a probability assignment for each return in the sequence given its past observations. Specifically, the learner generates a conditional probability distribution $\mathbb{P}(\boldsymbol{p}_t|\boldsymbol{h}_{t'})$ where the $h_{i,t'} = \{p_{i,t'}\}_{t'=t-k-1}^{t-1}$ represents a sequence of historical returns of length k up to time t. VOM models attempt to estimate conditional distributions of the form $\mathbb{P}(\boldsymbol{p}_t|\boldsymbol{h}_{t'})$ where the context length $|h_{i,t'}| = k$ varies depending on the available statistics. The changes in the logarithm of exchange rates, price indices, and stock market indices are usually assumed normal in the BlackScholes model Black and Scholes [1973]. In this paper, we make a similar assumption and let $\mathbb{P}$ be a multivariate log-normal distribution. That is, $\ln(\boldsymbol{p}_t|\boldsymbol{h}_{t'}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and covariance matrix of the assets' returns.

Generative Adversarial Network (GAN) is a machine learning framework that uses two neural networks, pitting one against the other, in order to generate new, synthetic instances of data that can pass for real data Goodfellow *et al.* [2014]. One neural network, called the generator, is responsible for the generation of stock price paths, and the second one, the discriminator, has to judge whether the generated paths are synthetic or from the same underlying distribution as the data (i.e., the asset returns). We denote $\boldsymbol{x}$ as a collection of all assets' daily returns $p_{i,t}$ in asset universe $\mathcal{I}$ for $t = 0, 1, \ldots, T$. To learn the generator's distribution $\mathbb{P}_g$ over the asset returns $\boldsymbol{x}$, we define a prior on input noise variables $\mathbb{P}_{\boldsymbol{z}}(\boldsymbol{z})$, then represent a mapping to data space as $G(\boldsymbol{z}; \theta_g)$, where G is a differentiable function represented by dense layers with parameters θ_g. We use convolution layers Krizhevsky *et al.* [2012] followed by dense layers for $D(\boldsymbol{x}; \theta_d)$ that outputs a single scalar. $D(\boldsymbol{x})$ represents the probability that $\boldsymbol{x}$ came from the historical price returns rather than $\mathbb{P}_g$. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $\log(1 - D(G(\boldsymbol{z})))$. In other words, D and G play the following two-player minimax game: $\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$.

Each trader generates one year of additional, synthetic time-series data for each of their assets. The synthetic data is generated every 1000 RL local training iterations and appended to the last day's real closing prices of the assets. This combined synthetic–real data, which is strictly confidential to each party, is used by each party to train its RL agent. The agent models follow the Ensemble of Identical Independent Evaluators (EIIE) topology with the online stochastic batch learning of Jiang *et al.* [2017], the latter of which samples mini-batches consecutively in time to train the EIIE networks. We use the same hyperparameters in Jiang *et al.* [2017] for agent models. We conduct our experiments on 10 virtual machines where each machine has 32 Intel® Xeon® Gold 6130 CPUs and 128 GM RAM and can support up to 5 party processes.

9.4 Additional Results

Table 2: Average norms of covariance matrix for assets' returns in different asset universes.

Asset universes	$\mathcal{I}_1, \mathcal{I}_3, \mathcal{I}_4, \mathcal{I}_5$	$\mathcal{I}_2, \mathcal{I}_6, \mathcal{I}_7, \mathcal{I}_8, \mathcal{I}_9, \mathcal{I}_{10}$
Averaged ℓ_2-norm	1.35×10^{-3}	1.73×10^{-3}
Averaged Frobenius norm	1.51×10^{-3}	1.93×10^{-3}

We consider the case where there are $N = 10$ parties, each with different asset universes. The learning curves obtained by training each party independently (no fusion) can be found in Figure 8. Note that parties $2, 6, 7, 8, 9$ and 10 have unstable, diverging learning patterns as compared to parties $1, 3, 4$ and 5. As shown in Table 2, the averaged ℓ_2 and Frobenius norms of the covariance matrix for assets' returns in asset universes $\mathcal{I}_2, \mathcal{I}_6, \mathcal{I}_7, \mathcal{I}_8, \mathcal{I}_9$ and $\mathcal{I}_{10}$ are larger. In the federated setting, the average portfolio value in the testing period

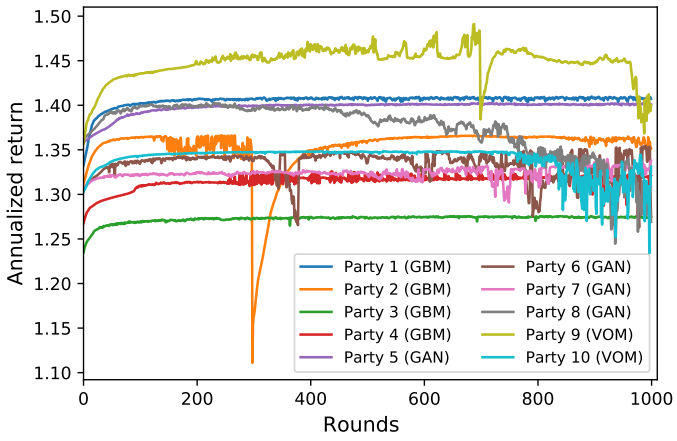

Figure 8: Learning curve for each party in the financial portfolio management problem, without fusion. Notice that several parties experience some training issues, which combined, are compounded and lead to training collapse in the federated training (see for example Figure 10).

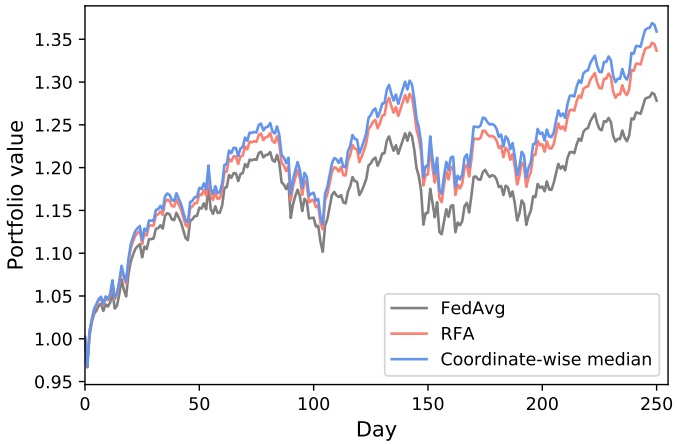

Figure 9: Average portfolio value in the 10-party financial portfolio management problem in the test period, for baseline methods.

Table 3: Average performances of different methods over 10 parties.

Method	Annualized return	Sharpe ratio
FedAvg	27.96%	1.52
RFA	35.09%	1.77
Coordinate-wise median	36.04%	1.79

for 10 parties is shown in Figure 9. For baseline methods FedAvg McMahan *et al.* [2017], RFA Pillutla *et al.* [2019] and coordinate-wise median Yin *et al.* [2018], we average across the 10 different asset universes; the

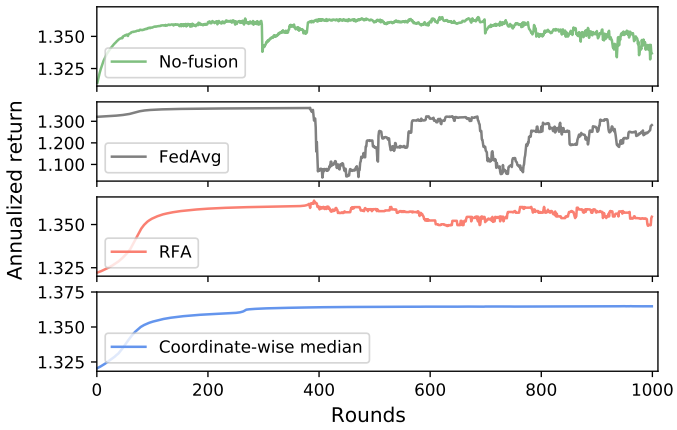

Figure 10: Average learning performance over 10 parties in the financial portfolio management problem, across methods. As in the main paper, note the training failure of the federated learning algorithms, which is not caused by adversarial parties or party-level failure, as evidenced by the single-party training curve at the top of the figure.

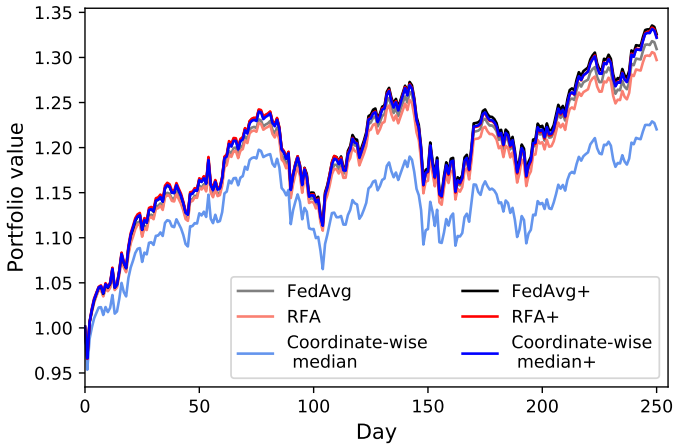

Figure 11: Average portfolio value of the 50-party financial portfolio management problem during the test period, across methods. The Fed+ methods are superior to the baselines by a significant margin as regards the financial portfolio management application.

average performance is summarized in Table 3.

To investigate the learning behavior, we first plot the average learning performance across 10 parties in Figure 10. It is clear that the average performance for FedAvg training is largely affected by participants with diverging learning such as parties $2, 6, 7, 8, 9$ and 10 as illustrated in Figure 8. FedAvg degrades the performance of all parties due in a large part to the training collapse (at training round 380). The robust federated learning methods, RFA and coordinate-wise median, achieve stable learning and yield better performance in 10-party federation. Finally, in the 50-party federation setting, the average portfolio value in the testing period is shown in Figure 11. Results are again as expected with Fed+ outperforming the other algorithms.