Relative Fair Federated Learning

Alexandru-Andrei Iacob
aai30@cam.ac.uk

Abstract

The canonical loss function of Federated Learning is a weighted average of the local client objective functions without regard to performance distribution amongst clients. A recently proposed class of algorithms, q-FFL, promotes fairness by raising the loss function to an exponent. While promising, this mechanism is sensitive to loss magnitude and requires extensive tuning to the dataset. This work proposes p-FFL, a relative form of fair aggregation where client loss functions are weighted solely on the basis of their position in the sorted order. Multiple algorithm versions based on p-FFL are implemented to emphasise different fairness functions which weigh a client's position linearly, geometrically or logarithmically. They are tested both in terms of fairness and convergence speed on three different ML applications. The results show that while p-FFL can improve the fairness of client performance distribution it is not yet competitive in its current form to q-FFL.

1 Introduction

The data and system heterogeneity inherent to Federated Learning McMahan et al. [2017], Kairouz et al. [2019] make difficult the construction of a global model capable of providing satisfactory results for all potential devices that it may be trained or deployed. Such heterogeneity may be intrinsic to the explored setting, e.g languages being heavily localised, meaning that decreased performance of the global model on a subset of clients could be unavoidable. A potential solution to this issue is to construct a "fairer" global model. In this context, fairness refers to the uniformity of model performance across devices with different data and system characteristics. Constructing a fairer model requires modifying the objective function used in FL.

The canonical Federated Learning (FL) objective was was formalised by Li et al. [2018] as follows:

$$\min_{w} f(w) = \sum_{k=1}^{m} p_k F_k(w) \tag{1}$$

where f is the global loss, w is the model, m is the total number of client, F_k are the local client loss functions and p_k are the weights associated with each client—generally chosen as its proportion of the training examples. The first version of Fair FL (FFL) was proposed by Li et al. [2019] who define a family of FFL algorithms, q-FFL, based around the following modification to the loss function

$$\min_{w} f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w) \tag{2}$$

where q is a parameter controlling the fairness of the weighing based on the value of the loss. A larger q implies a higher degree of fairness. When q tends towards infinity, the objective becomes identical to improving performance on only the worst-performing client. While Li et al. [2019] show a great deal of improvement over classical FL algorithms, their approach has two primary shortcomings. First, if one of the clients is an extreme outlier, they can skew the results significantly for high q values. Thus the model can either be made to diverge, or potentially manipulated to over-emphasise the needs of a specific client. Second, the optimal q-values vary significantly across domains. Li et al.

[2019] report q's ranging from 0.001 to 5 according to task. The primary hypothesis explored in this report is that developing a less loss-sensitive FFL algorithm could improve the robustness of results and require reduced parameter tuning between tasks.

This work brings three main contributions to the field of FFL. First, it provides a new class of relatively fair FL algorithms deemed p-FFL. This class uses a fairness mechanism based on generalised Gini social-evaluation functions [Weymark, 1981] rather than the α-fairness [Lan et al., 2010] of q-FFL. Second, it constructs three implementations—one of which has two variations—of p-FFL by modifying FedAvg to weigh client models using a linear, geometric or logarithmic function of their loss-sorted position. Third, it provides an experimental evaluations of these algorithms on three of the ML tasks from Li et al. [2019], showing that they can achieve a fairer distribution of performance than standard FedAvg but not q-FedAvg.

2 Background

2.1 q-FedAvg

The most important implementation detail of the q-FedAvg algorithm, the main exponent of the q-FFL class, is that it accounts for the change in objective function by scaling the model-training step-size according to an estimation of the objective function's Lipschitz constant. There are also two particular properties of q-FedAvg of relevance to this report. First, it uses the proportion of training examples held by a client as the probability of selecting that client rather than weighing the model updates by it. The authors justify this decision as being more efficient for q-FedAvg and show that it outperforms uniform sampling with weighted aggregation. Second, a q value of 0 makes q-FedAvg behaviour correspond to classical FedAvg with a tunable aggregate learning rate.

Li et al. [2019] defines a "fair" FL algorithm as one with a more uniform client testing accuracy distribution when running the global model on their individual partitions. Of course, every model having an accuracy of 0 would be perfectly fair, as such, the focus is on maintaining a similar average accuracy to a classical FL algorithm.

2.2 Relative Fairness

The proposed solution to the shortcomings of q-FFL relies on fairness metrics based on generalised Gini social-evaluation functions (G2SFs). It was inspired by the work of Sim et al. [2021] on Fair Collaborative Bayesian Optimisation.

2.2.1 Gini Social-evaluation Function

The original Gini index is a relative index for measuring inequality, initially formulated for specifically for income inequality. For the purposes of this work, the particular unit of measurement is not relevant and metrics can be calculated for any vector $\mathbf{u} \in R^n$—although it can be seen as representing some form of utility vector. According to Weymark [1981] , the Gini index $I(\mathbf{u})$ it can be defined as

$$I(\mathbf{u}) = 1 - \frac{1}{n^2}\left(\frac{\sum_1^n (2i-1)\phi(\mathbf{u})_i}{\mu(\mathbf{u})}\right) \tag{3}$$

where ϕ is a sorting operator inducing a descending permutation of values in $\mathbf{u}$ and $\mu(\mathbf{u})$ is the mean. The innermost sum assigns higher weights to lower values using the first n odd numbers as coefficients. This is known as the Gini Social-evaluation Function(GSF):

$$G_{SF}(\mathbf{u}) = \frac{1}{n^2}\left(\sum_1^n (2i-1)\phi(\mathbf{u})_i\right) \tag{4}$$

where n^2 is the sum of the first n odd numbers and is used as a normalisation factor. The Gini index creates a measure of inequality by subtracting mean-normalised fairness from 1. All elements being equal to the mean would make the GSF also equal to the mean and thus $I(\mathbf{u}) = 1 - \mu/\mu = 0$. Consequently, minimising the inequality of a distribution based on the Gini index is equivalent to maximising fairness based on the GSF by bringing all elements as close to the mean as possible.

2.3 Generalised Gini Social-evaluation Functions

The specific coefficients used in the GSF are arbitrary, according to Weymark [1981]. Allowing for the usage of different sets of weights creates an entire family of tunable fairness measures. Weymark [1981] defines the generalised Gini social-evaluation functions (G2SFs) as weighted sums of vector elements with non-increasing or non-decreasing weights $\mathbf{p}$ where the sorted order induced by ϕ decides the element-weight pairing

$$G_{2SF}(\mathbf{u}) = \mathbf{p}^T \cdot \phi(\mathbf{u}) \tag{5}$$

For example, an ascending order with non-increasing weights would put the highest emphasis on the smallest values in $\mathbf{u}$. The particular choice of sort order is arbitrary as long as $\mathbf{p}$ follows the opposite direction. For the purposes of designing an FFL algorithm, tuning the weight vector is equivalent to tuning both how fair the distribution should be based on the relative order of elements and the specific importance placed on an element being at a given index. Similarly to q-based fairness, the most extreme case of this would be to define $\mathbf{p} = (1, 0, ..., 0)$ which would optimise only for the lowest value in the list.

3 Methods

3.1 Relative-FFL Formulation

Given a particular weight vector $\mathbf{p} \in R^m$, we can re-state the FL objective function from eq.1

$$\begin{aligned} &\mathbf{F}(w) = (F_1(w), ..., F_m(w)) \\ &\min_w f_p(w) = \mathbf{p}^T \cdot \phi(\mathbf{F}(w)) \end{aligned} \tag{6}$$

This allows for the creation of a family of p-FedAvg algorithms where the client models are weighed by the corresponding element of $\mathbf{p}$. However, the original FedAvg [McMahan et al., 2017] assigned the weights p_k as the proportion of total examples seen by a clients. To maintain parity with q-FedAvg, p-FedAvg will also use the proportion of total examples of a client as the probability of picking that client rather than as a weight. However, if a scenario does not allow for such selection, multiplying by the proportion of total examples before applying the fairness-focused weighing is also a viable option.

One of the primary advantages of q-FedAvg is that the degree of fairness is controlled through a single parameter. As such, for an algorithm in the p-FedAvg family to prove useful it should also require as little manual tuning as possible in order to arrive at a fair performance distribution. This severely restricts the potential means of generating $\mathbf{p}$. Additionally, given that Federated Learning is not guaranteed to receive a constant number of client models and losses each round, any means of generating a weight vector must be resilient to changes in the number of clients—unlike more traditional applications [Sim et al., 2021].

3.2 Algorithm Formulations

Every version of the p-FedAvg uses the same underlying structure while only changing $\mathbf{p}$. The common structure is shown in algorithm 1, the programmer must only provide standard FedAvg parameters alongside the sorting operator—ascending or descending—and a method for dynamically generating a weight vector as long as the number of received models at a given round. All of the following methods divide the coefficients by their sum at the end as to avoid being heavily impacted by a variable number of clients.

3.2.1 Arithmetic Series

An arithmetic series-based version of p-FedAvg requires providing the starting value p_1 and the increment d while using an ascending sorting operator ϕ.

$$p_k = p_1 + (k-1)d \tag{7}$$

The original GSF (sec.2.2.1) is equivalent to such a series with $p_1 = 1, d = 2$. It is important to note that since the weights are linearly spaced and normalised by the total sum, the specific choice of a and d are not relevant outside of the magnitude of their difference. For very large values of d and small values of a, the first element will be largely ignored from the aggregation. If this gap between the first element and the rest is removed then the weight of an element depends entirely on its position and the overall number of values $p_k = 2(k+1) \times {}^{1}\!/\!{}_{n(n+1)}$. Given the similar behaviour of weights generated through such an arithmetic series, the rest of this report will specifically use the gini coefficients as the most time-tested of them.

3.2.2 Geometric Series

Using a geometric series for p-FedAvg is significantly simpler as normalising its terms by the total sum of the series makes the weights invariant with respect to the starting value. As such, it can be written entirely on the basis of the geometric term z and number of elements n. Unlike the arithmetic series, such a spacing is very sensitive to the geometric term with a strong trend for optimising for the worst-performing client as z values rise. This makes it the most similar of the p-FedAvg versions to q-FedAvg as choosing a good value of z may require quite a significant amount of trial-and-error. However, unlike q-FedAvg it is still resistant to the absolute loss values of the clients meaning that once found a good z value could potentially better generalise across domains.

$$p_k = \frac{q^k(q-1)}{q^n - 1} \tag{8}$$

Bounding z to values between 0 and 1.0 and using a descending sorting operator allow for a more intuitive understanding of the relation between z and the degree of fairness. Since $\lim_{z\to 1.0} p_k = {}^{1}\!/\!{}_{n}$ values of z close to 1.0 tend to result in a simple weighted average similar to the original FedAvg. On the other hand, as the values approach 0.0 the behaviour of p-FedAvg approaches only optimising for the worst-performing client.

3.2.3 Harmonic Series

The prior two versions of p-FedAvg may both emphasise fairness to an extent that significantly harms final model performance. A promising alternative for applications where global model performance is the foremost priority but still desire some fairness mechanism is to use the harmonic numbers for weight generation. For the n-th harmonic number $H_n \in O(log(n))$, making the spacing between weights increase only logarithmically.

$$p_k = H_k = \sum_{i=1}^{k} {}^{1}\!/\!{}_{i} \tag{9}$$

3.3 Experimental Setup

The experimental design is meant to replicate that of Li et al. [2019] as closely as possible. Great care has been taken to use the exact same datasets, data partitions, models and TensorFlow version [Abadi et al., 2015] by first forking the open-source repo and following the parameters and instructions of the authors verbatim. However, computational and space constraints limit the ability to compare p-FedAvg against q-FedAvg on every single ML application present in the original paper. An AWS EC2 instance equipped with an Nvidia T4 GPU with 16GB of VRAM has was used for training and testing all the models. This hardware setup is limited compared to the 8 NVidia 1080Ti GPUs used by Li et al. [2019].

3.3.1 Datasets and Models

The datasets and models used by Li et al. [2019] are curated from previous work in FL and meant to allow for easy evaluation of fair FL algorithms. For the purposes of this report three dataset and model combinations were chosen. The first is a synthetic dataset with highly non-IID data paired with a simple linear classifier. The second is the Vehicle dataset constructed by Duarte and Hu [2004] from a sensor network, paired with a linear SVM used for binary classification. The final data set is FMNIST [Xiao et al., 2017], paired with with a dense neural network.

3.3.2 Experimental Design

The experimental design intends to compare p-FedAvg variants against q-FedAvg with $q = 0$, corresponding to standard FedAvg, and against q-FedAvg with with q set to the optimal value found by Li et al. [2019] for each task. The versions of p-FedAvg used are the gini-based arithmetically weighed p-FedAvg, harmonic p-FedAvg and two variants of geometrically weighed p-FedAvg with $z \in \{0.85, 0.5\}$. There are several avenues of comparison.

3.4 Fairness

All p-FedAvg versions will have their average accuracy, variance, top 10% worst accuracy and top 10% best accuracy compared for the three datasets. The expectation is that p-FedAvg and q-FedAvg with $q > 0$ will have a similar average accuracy to FedAvg (i.e q-FedAvg with $q = 0$) whils having a lower variance and better worst 10% results. The effect is expected to be least pronounced for the harmonic version of p-FedAvg and most pronounced for the geometric version. Importantly, since geometric p-FedAvg most closely resembles q-FedAvg in terms of having one value which needs to be tuned it will be used to test the hypothesis that relative FFL allows for better transferability across domains. As such, instead of tuning the geometric term for each application the same values will be kept across with the expectation that similar improvements in fairness will be seen.

Besides these direct measurements, the accuracy distribution of clients will be graphed for each p-FedAvg variant in comparison with q-FedAvg at $q = 0$ and at the optimal q. Doing this for all three datasets would results in 12 total graphs, as such only the distribution for one dataset will be graphed in the main text with the rest relegated to the appendix.

3.5 Convergence Speed

While all discussion thus far has focused on the final fairness outcome of the experiments, it is inevitable that using a fair weighing scheme will affect the speed of convergence for a particular application and thus the communication efficiency of the FL algorithm. To understand the impact on this dimension, the accuracies per round of p-FedAvg and q-FedAvg conditions are graphed against one-another on the most relevant datasets.

4 Evaluation

4.1 Fairness Comparison

Table 1 shows the testing accuracy results for all tested experimental conditions. It shows that p-FedAvg variations are indeed capable of reducing accuracy variance and improving worst 10% performance—for the Vehicle dataset—when compared to the $q = 0$ condition. However, they fail to outperform q-FedAvg with any q value on any other metric for the other datasets. Furthermore, the mentioned improvements for the Vehicle dataset against $q = 0$ are meager at best. These experimental results immediately disprove any potential benefits that p-FedAvg was hypothesised to have over q-FedAvg. In the case of $q = 0$ one of the primary reasons for this is likely the tuned aggregation-algorithm-level learning-rate of q-FedAvg which p-FedAvg did not benefit from given the impracticality of doing a large parameter sweep with limited computational resources. For the optimal q it is rather clear that the extensively tested q value itself provides a very large advantage over the parameter-less gini and harmonic p-FedAvg variants. Most importantly, the hypothesis that the geometric p-FedAvg parameter would show better generalizsability than q is not supported by the current evidence as geometric p-FedAvg shows very high variance in terms of average accuracy between dataset/model combinations.

In terms of the relation between p-FedAvg variations, the assumptions made during algorithm creation hold up much better. Harmonic p-FedAvg shows the lowest variability between datasets, as expected given the logarithmic rate of growth of its weights. This comes at the cost of having a very low impact on the overall training process. Gini and geometric have a larger, but not still remarkably low, effect on variance and the worst 10% client performance. While gini p-FedAvg causes a 10% drop in average accuracy on the more complex non-Vehicle tasks, the magnitude of the impact for geometric p-FedAvg is largely dependent on the specific geometric term. As expected, the choice between a geometric term of 0.5 and 0.85 results in wildly varying outcomes with 0.5 doing more

Dataset	Objective	Average(%)	Worst 10%(%)	Best 10%(%)	Variance
Synthetic	q = 0	**72.8**	12.1	100	0.08
	q = 1	70.3	**20.0**	100	**0.06**
	Gini	56.8	0.00	100	0.144
	Harmonic	66.6	0.00	100	0.141
	Geo = 0.5	38.6	0.00	100	0.154
	Geo = 0.85	62.1	0.00	100	0.132
Vehicle	q = 0	85.5	41.9	**97.4**	0.034
	q = 5	**86.6**	**66.8**	95.2	**0.007**
	Gini	85.6	44.1	97.1	0.031
	Harmonic	85.5	41.6	97.3	0.034
	Geo = 0.5	85.1	45.7	96.2	0.028
	Geo = 0.85	85.7	43.7	96.9	0.032
FMNIST	q = 0	69.5	NaN	NaN	**0.015**
	q = 15	**79.0**	NaN	NaN	0.047
	Gini	50.7	NaN	NaN	0.350
	Harmonic	63.3	NaN	NaN	0.064
	Geo = 0.5	53.5	NaN	NaN	0.396
	Geo = 0.85	65.8	NaN	NaN	0.100

Table 1: Results for all datasets. Shown in bold is the best value for a column in a given dataset. The FMNIST application only has three clients, meaning that its worst% and best% metrics are not defined—nonetheless the variance results still serve as a proxy for the effectiveness of FFL.

to homogenise accuracy than 0.85. However, the slight decrease in variance brought by setting the geometric term to 0.5 comes with an upwards of a 24% drop in accuracy for the synthetic benchmark which violates the need to maintain rough accuracy-parity for FFL to be practically useful.

The impact of p-FedAvg variants on specific clients becomes more clear when inspecting the accuracy distribution for the synthetic benchmark in fig.1 and fig.2. The intended behaviour of an FFL algorithm, a largely homogeneous accuracy distribution without any immense peaks, is shown by q-FedAvg with $q = 1$ in fig.1. Unlike $q = 1$, gini p-FedAvg and harmonic p-FedAvg have large parts of their average accuracy determined by a group of clients with near 100% accuracy—although this is worse under the harmonic weighing. The geometric p-FedAvg shown in fig.2 with a parameter of 0.85 behaves similarly to the previous two versions. Setting the geometric term to 0.5% does in fact produce a "fair distribution" albeit with much worse average outcomes. The same trends hold under the Vehicle dataset in fig.5 and fig.6—relegated to the appendix.

4.2 Convergence Comparison

Figure 3 shows how the different p-FedAvg variations compare to q-FedAvg in terms of testing accuracy of the global model after each aggregation round. These results are mixed when compared to the fairness section as p-FedAvg is much more competitive at the global-model level than on a client-by-client basis. For q-fedAvg, a $q = 1$ is strongly associated with a colder start in terms of global model accuracy while no p-FedAvg, even the most extreme geometric one, shares this behaviour. This is potentially caused by the client model step-size tuning done by q-FedAvg for $q > 0$, since it relies on a mere estimation of the Lipchitz constant as a bound for the step-size. Despite this fact, the accuracy rapidly improves for q-FedAvg with $q = 1$ after the early training rounds. While the most performant algorithm remains q-FedAvg with $q = 0$, the results of harmonic p-FedAvg—as the least "fairness" focused variation—stay competitive with q-FedAvg with $q = 1$

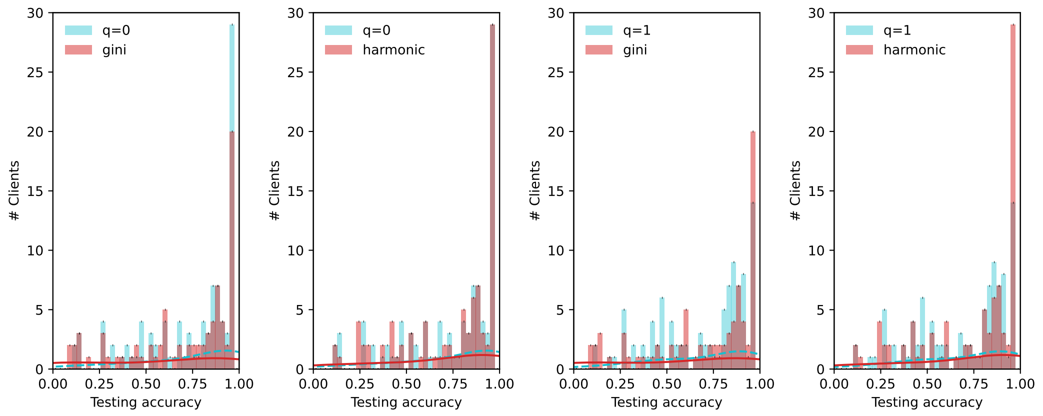

Figure 1: Fairness results comparing gini p-FedAvg and harmonic p-FedAvg against q-FedAvg for the the multi linear regression model trained on the Syntetic dataset . The q-values used are q=0 (Equivalent to normal FedAvg) and q=1 (The optimal q as reported by Li et al. [2019])

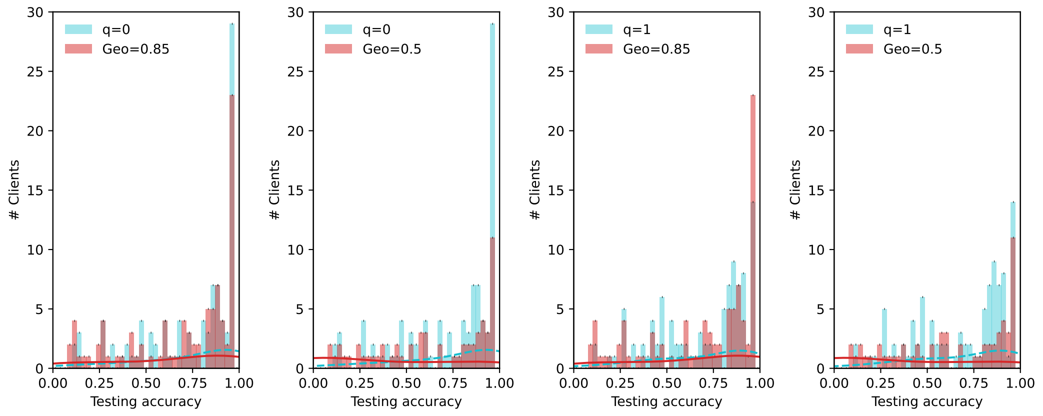

Figure 2: Fairness results comparing geometric p-FedAvg with the geometric term at $z \in \{0.85, 0.5\}$ against q-FedAvg for the linear classification model trained on the Synthetic dataset.

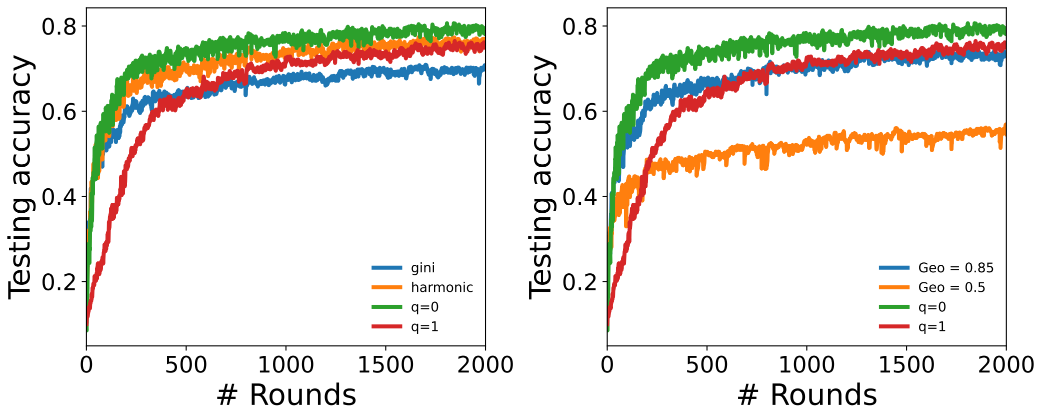

Figure 3: Convergence results comparing gini p-FedAvg, harmonic p-FedAvg and geometric p-FedAvg with parameters 0.5 and 0.85 against q-FedAvg for the the linear classification model trained on the Syntetic dataset.

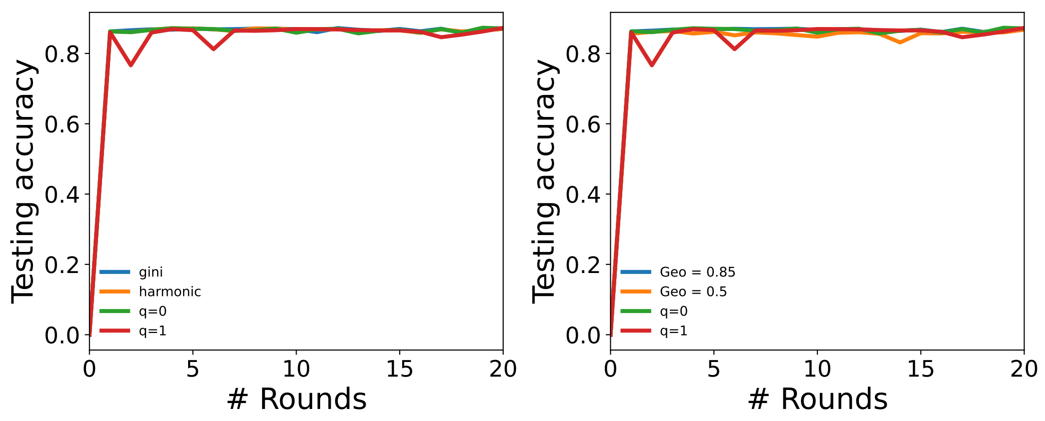

Figure 4: Convergence results comparing gini p-FedAvg, harmonic p-FedAvg and geometric p-FedAvg with parameters 0.5 and 0.85 against q-FedAvg for the the binary linear SVM model trained on the Vehicle dataset.

throughout while exceeding those of q-FedAvg with $q = 1$. A less encouraging trend is present for geometric p-FedAvg as both geometric term values fail to outperform either q-FedAvg.

4.3 Discussion

Using the relative order of elements was intended as a means of reducing the impact of task-specific loss values on aggregation algorithm behaviour and thus allow for more transferability across domains. As the large gaps between the three datasets show, inherent task difficulty and structure can paradoxically affect p-FedAvg more than q-FedAvg. The absolute gap in performance between p-FedAvg and q-FedAvg may be, as discussed, partially attributable to the better tuning of q-FedAvg, however, the much higher between-dataset variance is not explained by this fact. While task difficulty is relevant to any ML algorithm—the Vehicle application is easy enough that all algorithms converge to more-or-less the same value (fig4)—the very large performance collapse from the comparably difficult FMNIST to the synthetic dataset is unusual.

There are several potential explanations for this fact. The most plausible is that by taking away the loss-based scaling of objective functions, the algorithm is in fact deprived of important task-specific information and must instead rely on the task-independent weights. Rather than improving generalisability, p-FFL may in fact harm it. Another potential factor is that the automatic step-size scaling done by q-FedAvg may help independently of the q-value by simply attempting to adapt the learning process to the losses dynamically.

5 Conclusion

The goal of this work was the creation of an alternative form of Fair Federated Learning, p-FFL, by replacing the q-FFL fairness mechanism from one dependent on the absolute client loss values to a relative one based on their sorted order. While the proposed algorithms do indeed promote fairness to different extents, and thus represent a viable alternative formulation, their intended robustness against task heterogeneity proved to be lacking. This was shown across three datasets with four variations of p-FedAvg, and two of q-FedAvg. In fact, q-FedAvg was superior both in terms of reliability, absolute performance, and fairness to all p-FedAvg algorithms except for the one using the harmonic weighing. Consequently, the hypothesis that relative FL can provide improvements in robustness cannot be accepted.

There are a number of avenues through which p-FFL could be extended to a practically relevant FFL class. One is to close part of the implementation gap between p-FedAvg and q-FedAvg by tuning the aggregation learning rate of p-FedAvg similarly. In the same vein, dynamically changing the model learning rate similarly to how q-FedAvg uses the Lipschitz constant could improve convergence. Finally, a wider class of weightings could be explored, such as generalised Harmonic series.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Marco F Duarte and Yu Hen Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing*, 64(7):826–838, 2004.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Tian Lan, David Kao, Mung Chiang, and Ashutosh Sabharwal. An axiomatic theory of fairness in network resource allocation. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, 2010. doi: 10.1109/INFCOM.2010.5461911.

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

Tian Li, Maziar Sanjabi, and Virginia Smith. Fair resource allocation in federated learning. *ArXiv*, abs/1905.10497, 2019.

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.

Rachael Hwee Ling Sim, Yehong Zhang, Bryan Kian Hsiang Low, and Patrick Jaillet. Collaborative bayesian optimization with fair regret. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9691–9701. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/sim21b.html`.

John A Weymark. Generalized gini inequality indices. *Mathematical Social Sciences*, 1(4):409–430, 1981.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL `http://arxiv.org/abs/1708.07747`.

A Appendix

A.1 Algorithm

Algorithm 1 Baseline p-FedAvg

```
Input: $g$, $\phi$, $T$, $\eta$, $E$, $w^0$, $M$
 1: for each round t = 0,1...T-1 do
 2:     Sever attempts to select a set $S_t$ of $M$ devices probabilistically weighed by their sample sizes
 3:     for for each client $k \in S_t$ do
 4:         Client trains $w^t$ locally for E epochs with step-size $\eta$
 5:         if Client is trained successfully then
 6:             Client returns their trained model $w_k^{t+1}$ and the size of their dataset $n_k$ and loss $F_k$
 7:         end if
 8:     end for
 9:     Server generates the weight vector $\mathbf{p} = g(c)$, where c is the number of received models
10:     Server sorts $\mathbf{w}$ based on the sorted order of client losses $\phi(\mathbf{F})$
11:     Server multiplies client parameters by their weight and aggregates them
12:     $\mathbf{w} = \mathbf{p}^T \cdot \mathbf{w}$
13:     $w^{t+1} = \sum_{k=1}^{c} w_k^{t+1}$
14: end for
```

A.2 Vehicle Fairness Data

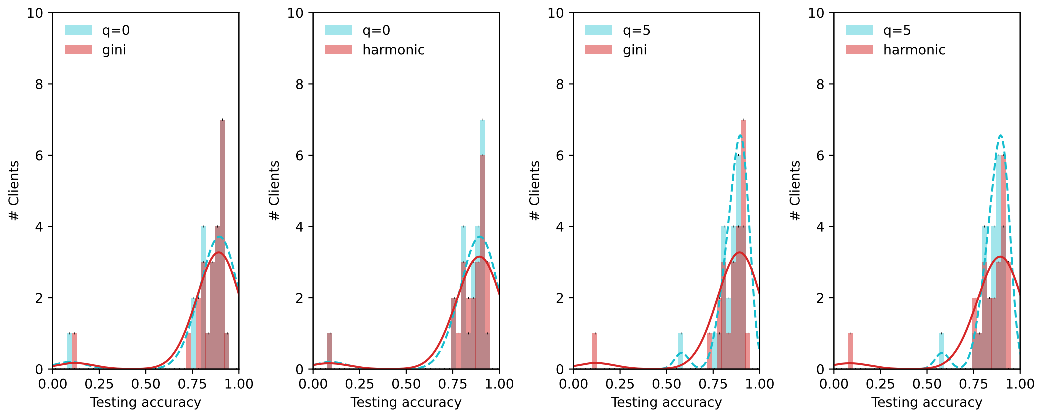

Figure 5: Fairness results comparing gini p-FedAvg and harmonic p-FedAvg against q-FedAvg for the SVM model trained on the Vehicle dataset. The q-values used are q=0 (Equivalent to normal FedAvg) and q=5 (The optimal q as reported by Li et al. [2019])

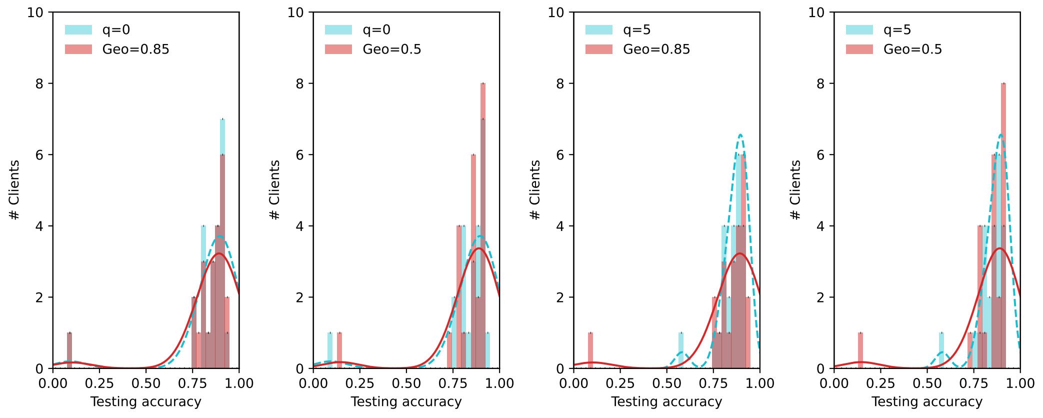

Figure 6: Fairness results comparing geometric p-FedAvg with the geometric term at $z \in \{0.85, 0.5\}$ against q-FedAvg for the SVM model trained on the Vehicle dataset. The q-values used are q=0 (Equivalent to normal FedAvg) and q=5 (The optimal q as reported by Li et al. [2019])