Is Your Data Relevant?: Dynamic Selection of Relevant Data for Federated Learning

Lokesh Nagalapatti[*1†], **Ruhi Sharma Mittal**[*2], **Ramasuri Narayanam**[3 †]

[1] IIT Bombay
[2] IBM Research AI
[3] Adobe Research India
nlokesh@cse.iitb.ac.in, ruhi.sharma@in.ibm.com, rnarayanam@adobe.com

Abstract

Federated Learning (FL) is a machine learning paradigm in which multiple clients participate to collectively learn a global machine learning model at the central server. Clients share updates derived from their local data to the server such that their privacy is not compromised. The server aggregates these updates and applies it to the global model. It is plausible that not all the data owned by each client is relevant to the server's learning objective. The updates incorporated from irrelevant data could be detrimental to the global model. The task of selecting relevant data is explored in traditional machine learning settings where the assumption is that all the data is available in one place. In FL settings, the data is distributed across multiple clients and the server can't introspect it. This precludes the application of traditional solutions to selecting relevant data here. In this paper, we propose an approach called Federated Learning with Relevant Data ($FLRD$), that facilitates clients to derive updates using relevant data. Each client learns a model called Relevant Data Selector (RDS) that is private to itself to do the selection. This in turn helps in building an effective global model. We perform experiments with multiple real-world datasets to demonstrate the efficacy of our solution. The results show (a) the capability of $FLRD$ to identify relevant data samples at each client locally and (b) the superiority of the global model learned by $FLRD$ over other baseline algorithms.

Introduction

The paradigm of Federated learning (FL) is introduced in (McMahan et al. 2017) where multiple clients collectively train a global model which we refer to as *Globally Learned Model (GLM)* in this paper. The data needed to train GLM is available or distributed across a set of clients. The local data possessed by each client is private to itself and is not shared with other clients or the server. The client(s) compute updates using their local data and shares it to the server where they are aggregated and applied to GLM.

It is plausible that not all data owned by each client is relevant to the GLM's objective (Toneva et al. 2018). Using irrelevant data to derive the clients' updates might be detrimental to the GLM. Hence it is important to develop an approach that selects relevant data at each client and thereby derive meaningful updates for GLM in each iteration of FL. The topic of selecting relevant data is explored in traditional machine learning settings (Ghorbani and Zou 2019; Yoon, Arik, and Pfister 2020) where the assumption is that all the data is available in one place. In FL settings, given that the data is distributed across multiple clients and the server can't introspect it due to privacy constraints, the traditional solutions to select relevant data are not applicable here.

We call data samples that are favorable to GLM as *relevant data* and those that are detrimental to GLM as *irrelevant data*. Irrelevance in data can arise due to a variety of reasons (Toneva et al. 2018; Wang, Wang, and Li 2019; Tuor et al. 2021; Zhu and Wu 2004). However, to accurately learn the GLM, it is worthwhile for each client to derive the updates only using its relevant subset of data. Hence, there is a need to introduce a mechanism that facilitates the clients to select relevant data from its local training data. This mechanism should necessarily adhere to the privacy requirements of the FL framework. In this work, we assume that no clients participate with adversarial intent; i.e., no client wants to purposefully corrupt the GLM.

The desired characteristic of a relevant data selection mechanism is that the relevance of a data sample should not be the same across multiple communication rounds. The value of a data point should change according to the state of GLM that varies across rounds. The updates computed by a client in communication round t is thus relevant to the server. Further as the GLM converges to a local optimum, value of data points also converge. In summary, the relevant data selection mechanism should be able to adapt to the dynamics of the FL environment that changes over time. We refer this dependence on time as Dynamic in the term *Dynamic Data Selection*.

Motivation for Our Work: we conduct an experiment to motivate the importance of detecting irrelevant/noisy data samples at each client in FL settings. We partition the Iris (Dua and Graff 2019) dataset among a server (S) and two clients (C_1,C_2). The server uses its data samples as test data. We use the FedAvg algorithm mentioned in (McMahan et al. 2017) to train a two-layered neural network (GLM) at the server. Then, we apply FedAvg to two cases: (a) After introducing 20% closed-set noise at each client as mentioned in (Wang et al. 2018) by flipping the labels of data samples; (b)

[*]These authors contributed equally.
[†]Work done while at IBM Research AI

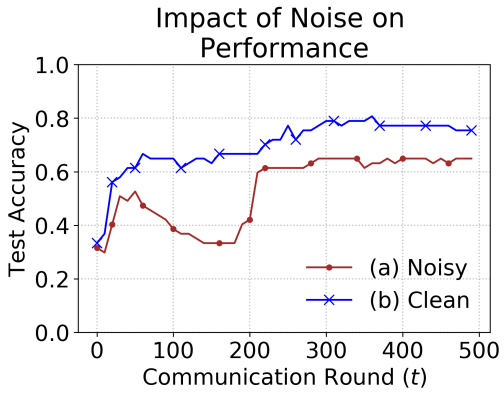

Figure 1: Impact of Noise on GLM in Federated Learning

After removing the noisy data samples introduced in case (a).

Figure 1 shows the performance of GLM on server's test data across 500 communication rounds. It is evident from the figure that the GLM trained in case (b) outperforms the GLM trained in case (a). This is despite the fact that clients have 20% lesser data in case (b). This experiment demonstrates the importance of identifying irrelevant/noisy data samples at each client and excluding them while deriving updates to collectively train GLM. In the subsequent sections, we propose an approach called *Federated Learning with Relevant Data ($FLRD$)* that enables clients to select data relevant to the server's objective which in-turn results in better performance of GLM.

In summary, there is a strong need for each client to learn a function that provides an estimate of relevance value to each data point without compromising the privacy constraints. We have provided an implementation of our approach in the supplementary material.

The following are the major **contributions** of our work:

- Data relevance estimation that assigns a relevance score to each datum at each client adaptive to the state of GLM.
- Policy gradients based solution for Data relevance estimation in Federated Learning (FL) setting called $FLRD$.
- Extensive experimental evaluation to highlight the usefulness of our approach in handling multiple types of irrelevancies in data such as label-noise, attribute noise, etc.

Related Work

Recent literature shows (Toneva et al. 2018) that not all the samples in the training data are equally important for learning, especially in Deep Neural Networks. Datasets usually contain irrelevant data points due to low data quality, outliers, label noise, attribute noise, distribution mismatch between training data and test data, etc. Impact of attribute noise and its possible solutions is highlighted in (Zhu and Wu 2004; Petety, Tripathi, and Hemachandra 2019; Mannino, Yang, and Ryu 2009). Various types of label noise categories are mentioned in (Wang et al. 2018). Different solutions for model training in the presence of label noise are explored in (Patrini et al. 2017; Han et al. 2018; Ghosh, Kumar, and Sastry 2017; Veit et al. 2017; Vahdat 2017; Hendrycks et al. 2018; Yu et al. 2019). Apart from label noise and attribute noise, a subset of training data becomes irrelevant when there is a mismatch in its distribution w.r.t to test data (Zhu et al. 2019; Zagoruyko and Komodakis 2016). Outliers are also examples of irrelevant data points in training data (Bergman and Hoshen 2020). With irrelevant data points in the training data, higher performance may be achieved after removing that subset from the training data as mentioned in (Ferdowsi, Jagannathan, and Zawodniok 2014; Frenay and Verleysen 2014). Therefore, detection and selection of relevant data are important for performance improvement.

Data Valuation is an area of research that provides a usefulness value to each datum. Leave-one-out (LOO) strategy assigns value to a data sample proportional to the performance difference of a model trained on data with and without that data sample. The approach of using influence functions (Koh and Liang 2017) for data valuation provides a closed-form expression for an approximation to LOO objective. The valuation strategy based on Shapely values (Shapley 1953), a cooperative game theory concept, provides equitable value to each datum. (Ghorbani and Zou 2019) used Monte Carlo sampling approximation and gradient-based estimation to reduce the computational complexity of the approach. The policy gradient method of Reinforcement learning (RL) is used for data valuation in (Yoon, Arik, and Pfister 2020). All the above-mentioned approaches for data valuation requires entire training data at one place to assign value to each data sample. Hence, these strategies are not directly applicable to the FL setting where training data resides privately at multiple clients and can't be shared due to privacy and bandwidth constraints.

Client Selection in FL: Recently, the field of client selection in FL has gained significant interest from the research community. Selecting the clients with relevant data using Shapely-based valuation is studied in (Nagalapatti and Narayanam 2021). Client selection for efficient communication and faster convergence is studied in (Jee Cho et al. 2020; Cho, Wang, and Joshi 2020; Tang et al. 2021). FedProf (Wu et al. 2021) matches server data footprint (baseline) with clients data footprints to select clients at each round for efficient and faster convergence. All these works have shown that client selections aids in faster convergence and better performance of FL algorithms. These works focuses on identifying the clients/updates as relevant or irrelevant based on complete data possessed by the client. However, in practical scenarios clients' training data is a mix of relevant and irrelevant data samples. While learning the global shared model, it is important to know the relevance of each data point for the global learning task to improve the performance of the GLM.

Data Selection in FL: The field of selecting the relevant data w.r.t. the central server's learning objective in the FL setting is under-explored. The detection of noisy and irrelevant data in the FL setting is studied in (Tuor et al. 2021)

and we refer to this approach as *KSLoss* in this paper. For identifying the noisy data samples, the server trains a benchmark model on a noise-free validation dataset and shares it with clients. Each client then computes the loss of each data point that it owns and communicates it back to the server. The server then runs a hypothesis test called *Kolmogorov–Smirnov (KS) test* on the cumulative distribution function of the losses corresponding to the test dataset and all clients' datasets to find a global threshold and communicates it back to the clients. Each client then filters out noise from its private data by dropping the data points that have a loss greater than the communicated threshold. This approach requires the identification of irrelevant data before starting the training of GLM and hence relevance of data is constant across time. Further, this strategy requires the sharing of losses w.r.t each data sample to the server which due to privacy constraints is a strong assumption in the FL setting.

One other approach that uses loss values to detect irrelevant samples is explored in Active Federated Learning *(AFL)* (Goetz et al. 2019). Unlike the previous approach, this approach is dynamic in nature. In each round, each client k shares a value $v_k = \frac{1}{\sqrt{n_k}} \sum_{(x,y)\in D_k} l(x,y)$ where D_k is the private dataset that k owns. Then the server uses these values to construct a probability distribution and does a biased selection of clients using the distribution. We use the above two methods as baselines in our experiments and refer to them as $KSLoss$ and the AFL respectively.

Our Approach: FLRD

We first introduce the formal problem statement and then discuss our proposed solution approach in detail.

Problem Statement

In a typical FL setting, a dedicated node called *server (S)* devises the objective of a globally learned model (GLM) to be built. It decides the architecture of the model denoted by $f_\theta : X \rightarrow Y$. The tuple $(x, y) \in (X, Y)$ denotes a sample input-output pair. The objective of the server is modeled by a loss function $l(y, f_\theta(x))$. For classification problems, a popular choice for l is categorical cross-entropy loss. We assume that the server is accompanied with validation data D_V and test data D_{Test} respectively. Samples from D_V and D_{Test} are assumed to be IID drawn from the target distribution. The server doesn't use these to train the GLM rather only uses them to test its performance. Server learns the parameters from a set of n clients, say $\{C_1, \cdots, C_n\}$. Each client C_i has local data D_i which is private to itself and is not transferable to other clients or the server.

The progress of FL happens in a sequence of communication rounds $t \in [T]$. In each round t, each client C_i downloads the parameters of the GLM denoted by θ^t. It then derives a local update ($\delta^t_{b_i}$) with a subset of data (b_i) it has and shares it with the central server. $\delta^t_{b_i}$ is typically the gradient derived from a subset of client's local training data, i.e., the updates of client C_i is $\delta^t_{b_i} = -\eta_i \frac{\partial}{\partial \theta}(l(b_i))\big|_{\theta^t}$. In this case, $\theta^t + \delta^t_{b_i}$ forms the parameters of the Locally Learned Model LLM_i at the i^{th} client for communication round t with learning rate η_i. The central server collects all the local updates $\{\delta^t_{b_1}, \cdots, \delta^t_{b_n}\}$ and aggregates them using an aggregation function Agg to derive the aggregate vector $\delta^t_{agg} = Agg(\delta^t_{b_1}, \cdots, \delta^t_{b_n})$. A popular choice for the Agg function is Federated Averaging (FedAvg) (McMahan et al. 2017). The server then applies the aggregated updates to GLM and computes the parameters for the next communication round using $\theta^{t+1} = \theta^t + \delta^t_{agg}$. The data D_i owned by each client C_i is generally prone to be noisy. We call data samples that are benign to GLM as *relevant data* and those that are detrimental to GLM as *irrelevant data* respectively.

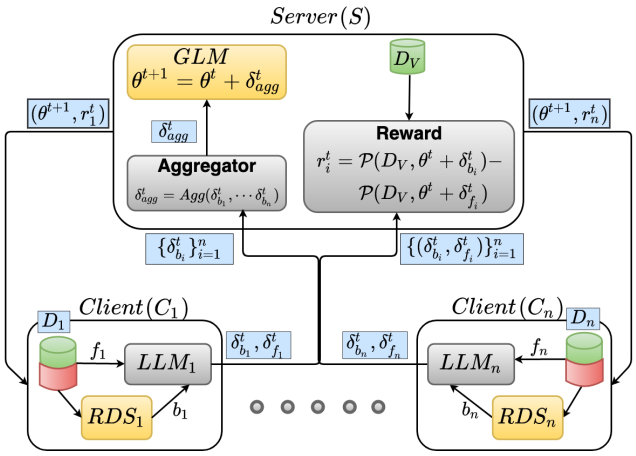

Figure 2: Schematic diagram of one communication round of proposed Federated Learning with Relevant Data ($FLRD$)

One interesting question in this regard is how each client learns a relevance prediction function locally that assigns Relevance Score (RS) to its data samples w.r.t. the GLM's objective? There are three key challenges associated with this problem: (1) The proposed solution should adhere to the privacy constraints imposed by the FL framework; (2) The client's local data is not a representative of the target distribution and hence designing a solution that relies only on client's local data may lead to sub-optimalities in the relevance prediction function and thereby it adversely affects GLM; and (3) As mentioned in Section , the relevance value of a data point $(x, y) \in D_i$ should vary as a function of time t.

Solution Approach

In our approach, each client learns a relevance prediction function locally that tries to predict the Relevance Score (RS) of each data point that it owns. RS ranges in $[0, 1]$. We call the relevance prediction function at each client C_i as Relevant Data Selector (RDS_i) which is local to C_i and is not revealed to other clients or the server. Hence, the strategy to select relevant data samples is determined solely by the client with the aid of the server as we will see in Section . In this paper, we use the terms value/relevance interchangeably.

Hence, in $FLRD$, there are $n + 1$ functions to be trained – one GLM at the server and n RDS_i functions, one each at

n clients. The state of GLM is accessible to both the server and the clients but the state of RDS_i is accessible only to the i^{th} client C_i. Both GLM and RDS_i are modeled as deep neural networks. The function signature for GLM and RDS_i is as follows.

$$GLM: \ f_\theta : X \rightarrow Y \tag{1}$$

$$RDS_i: \ g_{i\phi_i} : (X, Y) \rightarrow [0, 1] \tag{2}$$

$\{\theta, \phi_1, \cdots, \phi_n\}$ form the parameters to be learnt. In all our experiments, the parameters are the weights and bias vectors of fully connected neural networks. The overall objective of Federated Learning with Relevant data ($FLRD$) is given by:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \sum_{(x,y)\in D_i} g_{i\phi_i}(x, y) \cdot l(y, f_\theta(x)) \tag{3}$$

where the loss for each example is weighted by its corresponding relevance score. The target parameters θ^* are the ones that minimize the loss on unseen test data.

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{(x,y)\in D_{Test}} l(y, f_\theta(x)) \tag{4}$$

In literature, Equations 3 and 4 are also called as empirical risk and true risk respectively. We want $g_i(x, y)$ to be high for samples which when used to minimize θ will help us find a better $\hat{\theta}$ that is closer to θ^*. In other words, g_i should assign high values to relevant data samples. We interpret the output of g_i (between $[0, 1]$) as probability of the sample being relevant. Hence in each client C_i, for each $(x, y) \in D_i$, $g_i(x, y)$ acts as a Bernoulli random variable that denotes the event of a sample being relevant to GLM. We call this probability $g_i(x, y)$ as Relevance Score (RS) of the sample (x, y). Once g_i is learned, each client C_i samples a mini-batch b_i of examples according to relevance scores and derives the update $\delta^t_{b_i}$ using them to share with the server. Because g_i aids in the selection of relevant samples, we call it a Relevant Data Selector (RDS_i). The steps carried out in one round of $FLRD$ is elucidated in Figure 2.

Training GLM: We train GLM using the standard federated averaging algorithm (McMahan et al. 2017). Assuming that RDS_i is available at each client C_i, it first samples a mini-batch ($b_i \subseteq D_i$) of training data according to the relevant scores. It then fits a Local model LLM_i with b_i. To do so, at each communication round t, the client first downloads the GLM parameters θ^t from the server and does E number of successive Gradient Descent steps i.e., δ^t_i is obtained by performing E successive gradient steps locally at C_i using the data b_i. We use local learning rate $\eta_i = 0.01$ and $E = 10$ for all clients in all experiments. The γ^E_i forms the parameters of the local model LLM_i after E gradient steps. Finally, the update shared by C_i is computed as $\delta^t_{b_i} = \gamma^E_i - \theta^t$. The server then collects the updates from all the clients and aggregates them using Agg function. In our experiments, we use mean/average function for Agg. The server computes the average δ^t_{agg} and applies it to the GLM as follows.

$$\delta^t_{agg} = \frac{1}{n} \sum_{i=1}^{n} \delta^t_{b_i} \tag{5}$$

$$\theta^{t+1} = \theta^t + \delta^t_{agg} \tag{6}$$

This completes one round of GLM training. The clients will now download θ^{t+1} and participate in the next round using it. Next, we see how to train RDS_i that helps clients in selecting the best mini-batch for subsequent communication rounds.

Training RDS_i: Our solution to model RDS_i is inspired from (Yoon, Arik, and Pfister 2020). We use policy gradients to train RDS_i. In particular, we treat RDS_i as a local policy network of client C_i that helps in selecting the relevant mini-batch b_i. We use REINFORCE (Williams 1992) algorithm to optimize the policy gradients and the reward signal to train the policy network RDS_i is obtained from the server.

As mentioned in Section , the objective of each RDS_i module is to assign high values to relevant data samples local to the client C_i. To train RDS_i, we need feedback that measures the goodness of the relevance scores assigned by it. One way of doing it is to measure the performance gain in GLM when the update $\delta^t_{b_i}$ is incorporated in it. But computing the performance of GLM locally at each client is not advisable because, to get a good estimate of the performance, we need the evaluation of GLM to be done on samples that represent the server's target distribution. However, in FL, the client's data distribution is significantly different from the server's target distribution and hence the performance measured on local data samples possessed by the clients is not a good indicator of the actual test performance. Therefore, we need the intervention of the server to assist the clients in terms of getting feedback/reward for training RDS_i. However, due to privacy constraints, it is not appreciable for the clients to share the state of the RDS_i model to the server. We conduct an experiment to further validate this point [1]. At this juncture, there are two possible solutions. (1) the server makes validation data D_V public and each client uses it to compute the feedback locally. (2) D_V is private to the server and the server computes the feedback on it and communicates it back to clients subject to privacy constraints. While (1) is straightforward, it is not applicable in all cases. Especially if D_V involves sensitive data, regulations like General Data Protection Regulation (Yang et al. 2019), makes it prohibitive. Therefore it is more appreciable to develop solution approaches that follow (2).

In our approach, D_V is private to the server and is only used to compute the reward and not to train GLM. The only additional requirement made by us is that each client C_i in addition to sending the update $\delta^t_{b_i}$, sends one more update $\delta^t_{f_i}$, which is also obtained by performing E gradient steps locally at C_i using a subset $f_i (\subseteq D_i)$ that is sampled uniformly at random (not using mini-batch b_i). Hence the updates sent by each client C_i is a tuple of two entries $(\delta^t_{b_i}, \delta^t_{f_i})$. Now, server computes the reward r^t_i as follows.

$$r^t_i(b_i) = \mathcal{P}(\theta^t + \delta^t_{b_i}) - \mathcal{P}(\theta^t + \delta^t_{f_i}) \tag{7}$$

$$\mathcal{P}(\theta) = \frac{1}{|D_V|} \sum_{(x,y)\in D_V} \mathcal{I}(y == f_\theta(x)) \tag{8}$$

where $\mathcal{P}(\theta)$ is a performance measure on validation set with parameters of GLM as θ and $\mathcal{I}$ is the indicator function.

[1] refer to Section B in the supplementary material

Equation 8 shows classification accuracy as an example for the performance measure. Note that if D_V is indeed public, there is no need for client to share $\delta^t_{f_i}$ to the server and instead it can compute r^t_i locally. Our hypothesis is that if b_i is actually relevant, then $\delta^t_{b_i}$ is likely to add more value to GLM than $\delta^t_{f_i}$. Now, we see how each client C_i updates RDS_i using r^t_i.

The utility function that is to be maximized at each client C_i is

$$\max_{\phi_i} J(\phi_i) = \mathbb{E}_{\alpha \sim \pi_i}[r^t_i(b_i, f_i)] \tag{9}$$

where $\alpha \subseteq D_i$ and π_i is a probability distribution over 2^{D_i} given by

$$\pi_i(\alpha|D_i) = \prod_{(x,y)\in\alpha} g_{\phi_i}(x,y) \cdot \prod_{(x,y)\in D_i \setminus \alpha} \big[1 - g_{\phi_i}(x,y)\big] \tag{10}$$

Now, the client derives policy gradients for RDS_i as follows

$$\nabla_{\phi_i} J(\phi_i) = \nabla_{\phi_i} \sum_{\alpha \in 2^{D_i}} r^t_i(\alpha) \cdot \pi_i(\alpha|D_i) \tag{11}$$

We use Policy Gradients Theorem (Sutton and Barto 2018) to get an approximation of $\nabla_{\phi_i} J(\phi_i)$ as follows.

$$\nabla_{\phi_i} \hat{J}(\phi_i) = r^t_i(b_i) \cdot \big[\pi_i(b_i|D_i)\big] \cdot \big[\nabla_{\phi_i} log(\pi_i(b_i|D_i))\big] \tag{12}$$

where we approximate the expectation using one sample subset b_i that we use for deriving $\delta^t_{b_i}$. The gradients in Equation 12 can be obtained easily because g_{ϕ_i} is a neural network function which is differentiable. Finally, the client C_i updates RDS_i using

$$\phi_i = \phi_i + \zeta_i \nabla_{\phi_i} \hat{J}(\phi_i) \tag{13}$$

where ζ_i is the local learning rate to update RDS_i. We set $\zeta_i = 0.01$ in all our experiments.

Experiments

We evaluate the efficacy of our approach $FLRD$ (Federated Learning with Relevant Data) on multiple datasets by considering various scenarios of irrelevant data samples.

Datasets: We use eight public datasets to evaluate our approach. 7 datasets are tabular: (1) Adult, (2) Mushroom, (3) Iris, (4) Balance, (5) Monk, (6) Segment, (7) Contraceptive, and the last is an image dataset: (8) Flower [2]. The datasets are obtained from UCI (Dua and Graff 2019) and KEEL (Alcala-Fdez et al. 2010) repositories. For the Flower dataset, we extract a feature vector of 2048 dimensions per image using InceptionV3 (Chollet et al. 2018). Other details of datasets and data pre-processing are included in Section A of the supplementary material.

Irrelevant Data Samples at Each Client: To demonstrate the effectiveness of $FLRD$, we experiment with vanilla datasets (without introducing any noise) and noisy datasets. *We consider three types of noise: attribute noise, closed-set label noise, and open-set label noise.* For attribute noise, we use public datasets with 5% noise from Keel repository (Alcala-Fdez et al. 2010). To introduce $x\%$ closed-set label (Wang et al. 2018) noise, we randomly flip labels of $x\%$ data samples at each client. In open-set label noise, we assign $x\%$ out of distribution samples to each client and label them randomly. We refer the reader to (Wang et al. 2018) for more details about noise injection strategies.

Data Partitioning Among Server and Clients: For our experiments, given a dataset, we assign 10%, 20% of data samples as validation data D_V and test data D_{Test} respectively to the server. We distribute the remaining 70% samples among clients equally.

Model Architecture and Hyperparameters: For our experiments, at each client C_i we use a five-layered neural network (NN) of 100 neurons each as Relevant Data Selector (RDS_i) with a learning rate of 0.01. We select accuracy as the performance evaluation metric $\mathcal{P}$. We use a two-layered neural network of 100 dimensions each as Global Learning Model (GLM). For GLM also we consider the learning rate of 0.01 with a decay rate of 0.995 at every 50^{th} communication round. For both the networks, we use stochastic gradient descent (SGD) optimization algorithm.

Baselines: We illustrate our empirical findings in a variety of experiments. We compare our approach $FLRD$ with three baselines: (1) Standard Federated Averaging algorithm (McMahan et al. 2017) which we refer to as FL (2) Static filtering of data points according to their losses (Tuor et al. 2021) which we refer to as KSLoss, and (3) Dynamic filtering of clients according to their cumulative losses (Goetz et al. 2019) which we refer to as AFL.

Detection of Irrelevant Data Samples

In this experiment, data samples of the Adult dataset are partitioned among 10 clients ($C_1, C_2, \cdots, C_{10}$) and a server (S) as per the strategy explained earlier. To introduce the irrelevant data samples, we use a closed-set label noise strategy. The irrelevant data percentage in clients C_1, C_2 is 10%; C_3, C_4 is 20%; C_5, C_6 is 30%; C_7, C_8 is 40%; and C_9, C_{10} is 50%. Figure 3 shows the relevance score (RS) of data samples learned using RDS_i at each clients C_i after 100 communication rounds. In the interest of space, we show results only for odd clients. The green bars correspond to non-noisy data samples whereas the red bars correspond to noisy data samples. It is evident from the figure that the relevance scores of non-noisy samples are higher than that of noisy samples across clients. Hence, using our approach, the RDS_i of each client i can differentiate between noisy and non-noise data samples at each client. Please refer to the section G in supplementary material for graphs of all clients for Adult, Flower, and Mushroom datasets. In all these results, we observe that RDS_i assigns high values to non-noisy data samples consistently across all clients. We further note that due to the dynamic nature of $FLRD$, the magnitude of relevance keeps changing across communication rounds. However, the trend between noise and non-noise samples remains consistent.

[2] https://www.tensorflow.org/datasets/catalog/tf_flowers

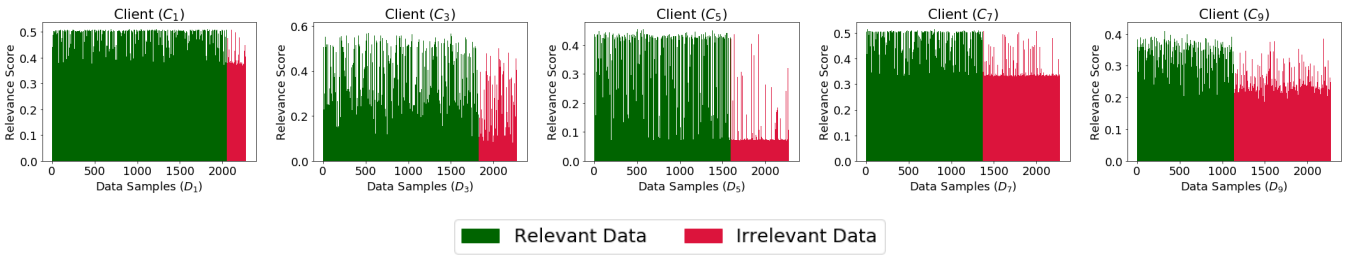

Figure 3: Relevance scores of clients using Adult dataset with closed-set label noise obtained after 100 communication rounds. The noise percentage in client C_1, C_2 is 10%; C_3, C_4 is 20%; C_5, C_6 is 30%; C_7, C_8 is 40%; C_9, C_{10} is 50%. The data samples are sorted for the representational purpose only; however, in the training data the samples are shuffled.

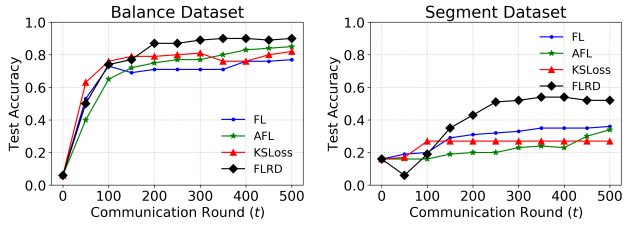

Figure 4: Performance of GLM on test data using $FLRD$ and other baselines across multiple communication rounds with datasets having 5% attribute noise.

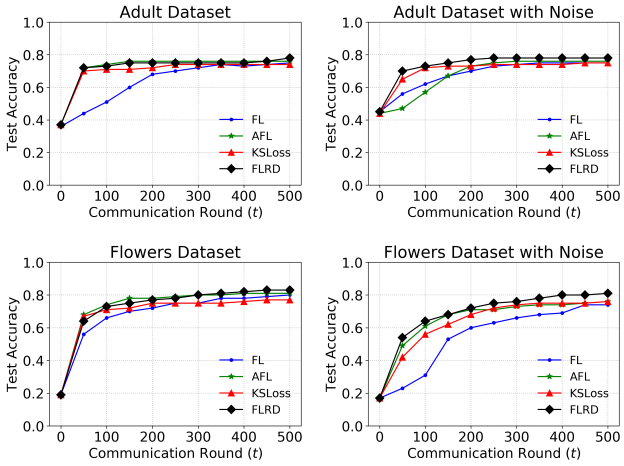

Figure 5: Performance of GLM on D_{Test} using $FLRD$ and other baselines across multiple communication rounds with the original dataset (without noise) and noisy dataset (with noise).

FLRD vs Baselines: Attribute Noise

In this experiment, we show the impact of $FLRD$ with attribute noise. We use 4 datasets with 5% attribute noise: Balance, Contraceptive, Monk, and Segment obtained from (Alcala-Fdez et al. 2010). The training data contains attribute noise whereas D_{Test} and D_V are noise-free. The noisy training data samples are divided among 10 clients and we run all the algorithms for 500 communication rounds independently. Figure 4 shows that even in presence of attribute noise at each client, the performance of GLM using $FLRD$ is significantly better than the performance of other baselines. A close competitor to $FLRD$ on all but the Segment dataset is AFL. This observation underlines the need for dynamic data valuation in a federated learning setting. Please refer section E of supplementary material for results on Contraceptive and Monk datasets.

FLRD vs Baselines: Closed-Set Label Noise

In this experiment, we demonstrate the impact of RDS_i on the performance of GLM. To do so, we run $FLRD$ and the other three baselines independently on two versions of the dataset viz. original and corrupted. For original, we use the ground truth dataset as is and for corrupted, we introduce closed-set noise where noise percentage is randomly taken from $\{5\%, 7\%, \cdots, 25\%\}$ at each of the 10 clients. Ideally, if RDS_i learns to detect irrelevant samples correctly, each client would send updates derived only from relevant data to the server which in turn would lead to a better GLM. Figure 5 shows that GLM trained using $FLRD$ outperforms GLM trained using other baselines in both original and corrupted datasets. We further note that because FL does not take countermeasures to mitigate the impact of noise on GLM, its performance is much less on the corrupted datasets. Please refer to Section C for results on Mushroom dataset. These results signify that at each client identifying relevant data samples and then using only those to send updates to the server is important for building an efficient GLM. Please refer to section D of supplementary material for results on open-set label noise.

Impact of Removing Data Samples With High/Low Relevance Score

In this experiment, we show that removing data samples with high relevance scores (RS) deteriorates GLM performance whereas removing data samples with low RS helps to improve it. To do so, we take a dataset and split it across the server and 10 clients. We introduce 15% closed-set label noise in each client. Then we run $FLRD$ for 500 communication rounds and record the RS of data samples at each client. Recall that RS of samples of C_i is given by RDS_i. We sort the data samples locally at each client in descending

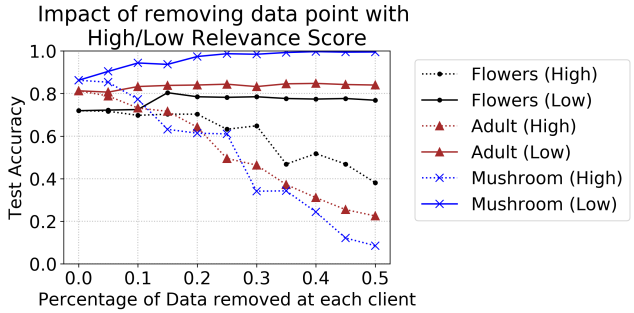

Figure 6: Performance of GLM on D_{Test} after removing data samples with the high/low relevance score at each client.

order of their corresponding RS. Then we conduct two experiments viz. High, Low. For High, we remove $x\%$ samples with high RS at each client. Then, we train the GLM using the standard FedAvg algorithm (McMahan et al. 2017). We run FedAvg for 500 communication rounds. We repeat this for various x percentages: $\{5\%, 10\%, \cdots, 50\%\}$. For Low scenario also, we follow a setup similar to High except that we remove $x\%$ samples with low RS at each client.
As shown in Figure 6, it is evident that removing samples with high RS indeed affects the performance of GLM adversely. On contrary, removing samples with low RS improves GLM performance. We can consistently observe that removing as many as 50% samples with low RS scores didn't affect the GLM at all. Whereas removing even $10-20\%$ samples with high RS has a noticeable negative impact on GLM. We observe this trend consistently across multiple datasets: Flower, Adult, and Mushroom.

Parameter Sensitivity: Noise Percentage

In this experiment, we vary the noise percentage across $\{10\%, 20\%, 30\%, 40\%\}$ and check its impact on the performance of GLM. We partition the Mushroom dataset across a server and 10 clients and introduce closed-set label noise appropriately. Then we train GLM using FL and $FLRD$ and measure the performance on D_{Test} across 500 communication rounds. As we can see in Figure 7, $FLRD$ consistently outperforms FL. This shows the robustness of $FLRD$ to both low and high noisy datasets.

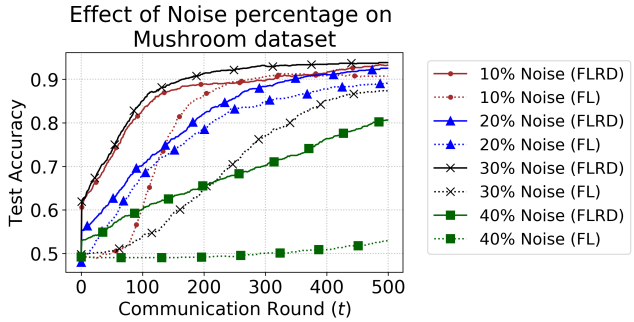

Figure 7: Parameter Sensitivity: noise percentage

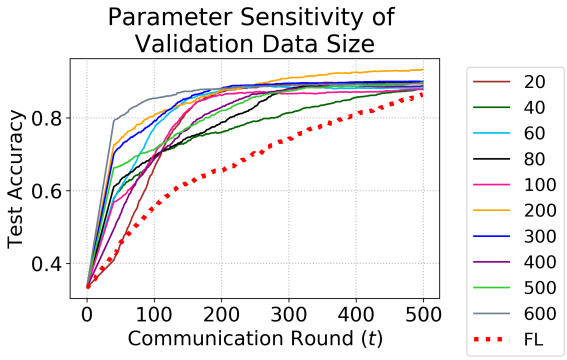

Figure 8: Parameter Sensitivity: validation dataset size

Parameter Sensitivity: Validation Dataset Size

Our approach $FLRD$ requires a validation set (D_V), using which the server calculates reward for each client's update. To provide a correct reward, the validation data needs to be of high quality, noise-free, follow the same distribution as test data, etc. Collecting such samples is costly. Hence $FLRD$ must work well in cases where we don't have abundant validation samples. In this experiment, we show the sensitivity of our approach to the size of validation data. We work with the Mushroom dataset and split it across the Server and 10 clients. We introduce 20% closed-set label noise in each client. Then, we run $FLRD$ multiple times with the varying number of validation samples in the server. In particular, we vary the size of validation data from 20 to 600 samples and record the performance of $FLRD$ across 500 communication rounds. As we can see in Figure 8, $FLRD$ outperforms FL even when validation samples are scarce. However, the convergence of GLM using $FLRD$ is fast when the size of the validation dataset is large.

Conclusion and Future Work

In this paper, we proposed an approach called $FLRD$ that uses policy gradients to train a module called RDS_i. We showed that RDS_i is instrumental in performing dynamic data valuation in Federated Learning (FL) to select relevant data for sharing updates at each client which in turn helps in improving the performance of GLM. Through extensive experimental analysis using multiple real-world datasets and various types of irrelevance scenarios, we demonstrated the effectiveness of our approach over other baselines. We like to extend the proposed $FLRD$ framework to active learning settings where we assume that each client possess a small amount of labeled data and a large amount of unlabeled data. The problem then is to use a variant of RDS_i to assign a high probability to those samples which when used in training post annotation (through an annotation service by incurring an appropriate cost) would be effective in training GLM. Clients cannot afford invocations of the annotation service above a cost budget and hence the problem becomes even more challenging. Current proposal for RDS_i doesn't take cost of exploration into account and We like to design appropriate solution approaches as part of future work.

References

Alcala-Fdez, J.; Fernández, A.; Luengo, J.; Derrac, J.; Garc'ia, S.; Sanchez, L.; and Herrera, F. 2010. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17: 255–287.

Bergman, L.; and Hoshen, Y. 2020. Classification-Based Anomaly Detection for General Data. *ArXiv*, abs/2005.02359.

Cho, Y. J.; Wang, J.; and Joshi, G. 2020. Client Selection in Federated Learning: Convergence Analysis and Power-of-Choice Selection Strategies. *ArXiv*, abs/2010.01243.

Chollet, F.; et al. 2018. Keras: The python deep learning library. *Astrophysics source code library*, ascl–1806.

Dua, D.; and Graff, C. 2019. UCI Machine Learning Repository [http://archive. ics. uci. edu/ml]. Irvine, CA: University of California. *School of Information and Computer Science*, 25: 27.

Ferdowsi, H.; Jagannathan, S.; and Zawodniok, M. 2014. An Online Outlier Identification and Removal Scheme for Improving Fault Detection Performance. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5): 908–919.

Frenay, B.; and Verleysen, M. 2014. Classification in the Presence of Label Noise: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5): 845–869.

Ghorbani, A.; and Zou, J. 2019. Data Shapley: Equitable Valuation of Data for Machine Learning. In *International Conference on Machine Learning*, 2242–2251.

Ghosh, A.; Kumar, H.; and Sastry, P. S. 2017. Robust Loss Functions under Label Noise for Deep Neural Networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, 1919–1925. AAAI Press.

Goetz, J.; Malik, K.; Bui, D.; Moon, S.; Liu, H.; and Kumar, A. 2019. Active Federated Learning. *CoRR*, abs/1909.12641.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*, 8535–8545.

Hendrycks, D.; Mazeika, M.; Wilson, D.; and Gimpel, K. 2018. Using Trusted Data to Train Deep Networks on Labels Corrupted by Severe Noise. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*, 10456–10465. Curran Associates, Inc.

Jee Cho, Y.; Gupta, S.; Joshi, G.; and Yağan, O. 2020. Bandit-based Communication-Efficient Client Selection Strategies for Federated Learning. In *2020 54th Asilomar Conference on Signals, Systems, and Computers*, 1066–1069.

Koh, P. W.; and Liang, P. 2017. Understanding Black-box Predictions via Influence Functions. volume 70 of *Proceedings of Machine Learning Research*, 1885–1894. International Convention Centre, Sydney, Australia: PMLR.

Mannino, M.; Yang, Y.; and Ryu, Y. 2009. Classification algorithm sensitivity to training data with non representative attribute noise. *Decision Support Systems*, 46(3): 743 – 751. Wireless in the Healthcare.

McMahan, H.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

Nagalapatti, L.; and Narayanam, R. 2021. Game of Gradients: Mitigating Irrelevant Clients in Federated Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10): 9046–9054.

Patrini, G.; Rozza, A.; Menon, A.; Nock, R.; and Qu, L. 2017. Making Deep Neural Networks Robust to Label Noise: A Loss Correction Approach. 2233–2241.

Petety, A.; Tripathi, S.; and Hemachandra, N. 2019. Attribute noise robust binary classification. arXiv:1911.07875.

Shapley, L. S. 1953. A value for n-person games. *Contributions to the Theory of Games*, 2(28): 307–317.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book. ISBN 0262039249.

Tang, M.; Ning, X.; Wang, Y.; Wang, Y.; and Chen, Y. 2021. FedGP: Correlation-Based Active Client Selection for Heterogeneous Federated Learning. *ArXiv*, abs/2103.13822.

Toneva, M.; Sordoni, A.; des Combes, R. T.; Trischler, A.; Bengio, Y.; and Gordon, G. J. 2018. An Empirical Study of Example Forgetting during Deep Neural Network Learning. *CoRR*, abs/1812.05159.

Tuor, T.; Wang, S.; Ko, B. J.; Liu, C.; and Leung, K. K. 2021. Overcoming noisy and irrelevant data in federated learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 5020–5027. IEEE.

Vahdat, A. 2017. Toward Robustness against Label Noise in Training Deep Discriminative Neural Networks. *CoRR*, abs/1706.00038.

Veit, A.; Alldrin, N.; Chechik, G.; Krasin, I.; Gupta, A.; and Belongie, S. 2017. Learning From Noisy Large-Scale Datasets With Minimal Supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 839–847.

Wang, L.; Wang, W.; and Li, B. 2019. CMFL: Mitigating Communication Overhead for Federated Learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 954–964.

Wang, Y.; Liu, W.; Ma, X.; Bailey, J.; Zha, H.; Song, L.; and Xia, S. 2018. Iterative Learning with Open-set Noisy Labels. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8688–8696.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4): 229–256.

Wu, W.; He, L.; Lin, W.; and Mao, R. 2021. FedProf: Efficient Federated Learning with Data Representation Profiling. arXiv:2102.01733.

Yang, Q.; Liu, Y.; Chen, T.; and Tong, Y. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2): 1–19.

Yoon, J.; Arik, S.; and Pfister, T. 2020. Data valuation using reinforcement learning. In *International Conference on Machine Learning*, 10842–10851. PMLR.

Yu, X.; Han, B.; Yao, J.; Niu, G.; Tsang, I.; and Sugiyama, M. 2019. How does Disagreement Help Generalization against Label Corruption? *ArXiv*, abs/1901.04215.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. arXiv:1605.07146.

Zhu, L.; Arik, S. O.; Yang, Y.; and Pfister, T. 2019. Learning to Transfer Learn: Reinforcement Learning-Based Selection for Adaptive Transfer Learning. arXiv:1908.11406.

Zhu, X.; and Wu, X. 2004. Class Noise vs. Attribute Noise: A Quantitative Study. *Artificial Intelligence Review*, 22(3): 177–210.