Machine Learning and Optimization Laboratory

Incentivized, Privacy-preserving, and Personalized Distributed Machine Learning Framework for Medical Data

Semester project

by Felix Grimberg

Mary-Anne Hartley
Supervisor

Martin Jaggi
Co-supervisor

Sai Praneeth Reddy Karimireddy
Co-supervisor

Andres Colubri
Co-supervisor

09/06/2020

Abstract

Contents

1 Aim

This project is part of a larger endeavour to create a platform/framework for medical professionals around the globe, and espeicially in low-resource settings, to train machine learning models collaboratively. Main requirements to achieve this goal are identified and separated into necessary features and performance features.

1.1 Necessary features

Data privacy. First and foremost, the proposed framework must safeguard each participant from the leakage and re-identification of individual patients stored within their data set. This implies maintaining data at its original location and limiting transfers, which motivates the use of distributed machine learning techniques.

Incentives and intellectual property. Secondly, it must present incentives for the collection of high-quality health data and protect the intellectual property of their owner.

Resilience. Moreover, it must be designed to work with incomplete and non-identically distributed data, as well as with arbitrary connection/disconnection patterns from participants.

Personalisation. Finally, it is important that the selection of training data, features, and model architecture, is interpretable to the users (i.e., medical professionals) and subjected to their judgement. To this end, users must be given the tools to ensure that the trained machine learning model is adapted to the particularities of their local population.

1.2 Performance features

Data analysis. For better model personalisation, the proposed framework should facilitate the selection of appropriate training data by providing and visualizing measures of similarity between data sets.

Feedback. Additionally, it should give its users feedback on the quality of their data collection. For instance, it could point out physiological or epidemiological anomalies (e.g., breathing rates being consistently reported higher than usual). Likewise, it could suggest which additional features should be recorded to efficiently maximize future model performance and thus increase their collaborative scope.

Environmental impact. The framework should also be optimised to quantify, minimise and possibly compensate for the environmental impact associated with training each machine learning model.

Communication efficiency. The communication effort exerted by each participating device should be kept as low as possible, to facilitate participation in low-infrastructure settings. This simultaneously serves to reduce the environmental impact of each training run.

Robustness. Finally, the design should be robust to a small fraction of non-collaborative participants. Different robustness models exist for various types of non-collaboration, ranging from the inadvertent provision and use of false data to fully adversarial behaviour.

2 Background

Available tools for reproducible data science

1. Overall problem
2. What people have done to solve this problem → literature

- Record software versions, hardware and dates and times (e.g., using watermark[1]).

[1] https://pypi.org/project/watermark/

- Better yet, use virtual environments / containers. Google colab and Renku are convenient services that provide their own standardized virtual environment. Modifications to that environment can be recorded in the same notebook that also contains the code.
- Use version control! I.e., git or dvc[2]
- Maybe use a standard project structure (files/folders)
- Sumatra (probably not worth trying out in the scope of my semester project): "for each experiment that you conduct through Sumatra, this software will act like a 'save game state' often found in videogames."[3]

Available tools for (privacy-preserving) distributed machine learning

1. Overall problem
2. What people have done to solve this problem → literature

- [8]: Overview of the FL problem, including a discussion of *all* aspects discussed in this paper
- [1]: Secure and byzantine-robust aggregation scheme. Relies on securely computing the pairwise distance between the gradients of each client at each round, and using them to select a byzantine-robust subset of clients at each round (e.g. via multi-KRUM [2]).
- In terms of methods(references in ERC grant application):
  - Local SGD
  - Secure multi-party computation (could be incorporated in all of my proposed methods, I believe)
  - Differential Privacy (Don't have time to learn how it works)
  - Homomorphic Encryption: Not in our focus.
- The open-source Python library PySyft[4] "using Federated Learning, Differential Privacy, and Encrypted Computation (like Multi-Party Computation (MPC) and Homomorphic Encryption (HE)) within the main Deep Learning frameworks like PyTorch and TensorFlow."
- `https://github.com/epfml/collaborative-learning-benchmark` Doesn't do privacy-preserving stuff, but they simulate various types of non-iid-ness for CV and NLP tasks.

Available tools for (privacy-preserving, distributed) data analysis

1. Overall problem
2. What people have done to solve this problem → literature

- [13]: SQL based data analysis platform that integrates differential privacy. Each user can make queries only within their allotted ϵ-privacy budget. Users need not have knowledge of any privacy-related topics. Similar work: [5]
- [14]: Presents methods and metrics for privacy-preserving ML (supervised and unsupervised) and KDD (knowledge discovery from data) at different stages of the data lifecycle: data collection, data publication and data output (i.e. after a model has been trained). Includes a discussion of SMC and HE as methods of distributed privacy.

[2] https://dvc.org/
[3] https://datascience.stackexchange.com/questions/758/tools-and-protocol-for-reproducible-data-science-using-python
[4] https://github.com/OpenMined/PySyft

Existing methods for model personalisation / training data selection (TDS) and varying degrees of inequality between data sets.

1. What people have done to solve this problem → literature

- [8]: Overview of the various ways in which the assumption of IID data can be violated. Also, overview of existing methods to do FL effectively on non-IID data (i.e., personalization methods):
  - **Featurization.** include personal context information as features
  - **Multi-task learning.** One task per client, or one task per cluster of clients.
  - **Local fine-tuning.** of a trained global model
  - **Learning-to-Learn (Meta-learning).** optimize a learning algorithm by sampling over learning tasks
  - **Model-agnostic Meta-learning.** meta-learn a global model to use as starting point for learning a good local model in only a few local gradient steps.
- [11]: Three different methods (describe exactly our situation with learning theory notation)
  - **Hypothesis-based user clustering.** This is a form of multi-task learning. Find the best set of q hypothesis, which minimizes the total loss summed over all k distributions when each distribution picks out the hypothesis that works best among the q options -¿ q clusters based on which of the q hypotheses works best.
  - **Data interpolation.** Minimize the loss for a weighted combination of local data and global data. → more data means model generalizes better, but it is less specific.
  - **Model interpolation.** used in virtual keyboards. Jointly optimize (by alternating steps) the global model, the local models, and the proportion global-to-local for each client. More exactly: Each round, find the best local model and proportion by grid-search over the porportions. The "local" model is more something like a difference/correction model unless the proportion is full-local. Then do a gradient descent step for the global model.
- [12]: Model interpolation (similar to [11])
- [9]: A selection distribution generator (SDG) and a predictor are optimized via reinforcement learning (RL) by taking each other's outputs as input.
  - Self-optimizing system for training data selection → no manual thresholding needed, but also no immediately obvious way to involve users.
  - Proven method (achieved competitive performance in their paper).
  - Effective TDS can reduce the environmental impact of model training.
  - Designed for a neural network consisting of a feature extractor followed by a classifier (Where the SDG is also a neural net). The selected training data and the guidance set are transformed into a feature representation by the feature extractor. The SDG is rewarded according to the improvement of one of the following distribution discrepancy measures (refer to their cited sources):
    - Jensen-Shannon divergence (= arithmetic mean of the Kulback-Leibler divergence of each distribution with the arithmetic mean of the distributions)
    - Maximum mean discrepancy
    - Symmetric Rényi divergence
    - Guidance loss... I don't exactly understand what it punishes/rewards.
  - Not immediately obvious how to use their contribution in a way that makes individual decisions explainable (model predictions and selection of specific training data), e.g., for medical applications.
- [15]: This Google patent proposes adapting a machine learning model based on information collected locally on the device, such as use patterns or voice samples of the user. It suggests:

  - "selecting a Subset of a machine learning model to load into memory"
  - "adjusting a classification threshold value of the machine learning model"
  - "normalizing a feature output of the machine learning model"

- [16]: This paper proposes an online active learning (OAL) approach for human activity recognition (HAR) where the user is asked to annotate their own tasks (parsimoniously, because they will not comply otherwise). It addresses a different problem of personalisation: instead of having labeled data from several sources and selecting a subset thereof, HAR is based on data from only one source, for which only a very restricted number of ground-truth labels can be generated.
- Model *personalisation* is different from model *selection* because we wish to select *rows*, not *columns* of the feature matrix.

Existing incentive schemes

1. Overall problem
2. What people have done to solve this problem → literature

- [18]: Designs a scheme for the efficient approximation of the influence of each user's provided data (i.e., how much their data has contributed to minimising the loss function), which could serve as a basis for rewarding users. This requires that the payer has a private test set on which to compute the loss / the influence. Under this assumption, their scheme incentivises truthful data collection and reporting.

OPTIONAL – Existing methods for model compression

1. Overall problem
2. What people have done to solve this problem → literature

- Gradient compression with error feedback (reference in ERC grant application)
- Model compression (references in ERC grant application):
  - model sparsification
  - weight quantization

3 Objectives

Within the scope of this project, our main focus will be on developing and evaluating methods for model personalisation. Throughout the project, we will be mindful of the other requirements identified in section 1 and seek methods that are compatible with, or even beneficial for, as many of them as possible. We will strive to achieve objectives 3.3 and 6.1 in a fully reproducible manner, while maintaining the confidentiality of the provided data sets.

3.1 Landscape analysis - 3 weeks

We will begin by exploring available tools for reproducible data science, privacy-preserving distributed machine learning, and data analysis, and investigating how these tools can be leveraged to efficiently realise the other objectives listed herein. This landscape analysis will also research existing incentive schemes, as well as methods for model personalisation and communication compression. Our findings will be documented in the background section of this report.

3.2 Model personalisation and data selection - 4 weeks

Often, each user collects data on a different population. For instance, a hospital in Freetown and an Ebola treatment center in rural Sierra Leone would see fairly different patients in general, which would again differ from the patients admitted in Ebola treatment facilities in the Democratic Republic of the Congo. While it is also highly useful to make global inference across populations, clinicians in the Freetown hospital may be more interested in being able to accurately predict outcomes for new patients admitted *to their hospital*, than in predicting outcomes for new patients admitted *to any facility in general*. They would still need to use each other's data collaboratively, simply because the amount of data collected by each user is limited.
We will propose a variety of novel ways to help users in optimising the selected model for their local population. Some of the proposed methods will involve ranking the data sets of other users, exploiting various notions of similarity to the local data set. Ideas that we will pursue are:

- **Ndoye:** Computing a (potentially adapted) Ndoye factor for each available user before training [17].
- **Influence:** Using a subset of the local data set as a test set, and then selecting other users according to how much they contribute to reducing the test loss. As an additional advantage, this would guarantee that the evaluation metrics of the resulting model measure its ability to make predictions on the local population (as opposed to the global population or some aggregated subset thereof).
- **Gradients:** Training a global model, where the mini-batch gradients computed by each user are used to define a notion of similarity. This concept is based on the idea that similar data sets will produce similar gradients in expectation over a mini-batch SGD step.

These ideas will be evaluated with respect to the following criteria:

- **User focus:** Freedom of choice left to the users and interpretability of the information presented to them.
- **Resilience:** Compatibility with the distributed machine learning setting, resilience to incomplete data and to unreliably connected users.
- **Incentives:** Potential to provide incentives and/or protect intellectual property.
- **Feedback:** Potential to generate feedback on data quality and feature collection for the user. This does not include feedback on the interoperability of data sets, as this can be done without any data selection method.
- **Privacy:** Compatibility with existing methods that ensure data privacy, such as secure multi-party computation (SMC).
- **Byzantine robustness:** Compatibility with existing methods that ensure robustness against byzantine participants. By their purpose, all effective data selection methods are robust against the mere poisoning of other users' data and highly susceptible to the poisoning of the requesting user's own data.
- **Complexity:** Computation and communication requirements.

We will then develop suitable ideas and formalize them into concrete methods to be described in the corresponding section of this report.

3.3 Application to non-iid medical data - 3 weeks

Next, we will implement these methods and apply them to a medical data set collected on 577 patients in Sierra Leone during the 2013 - 2015 West African epidemic of Ebola Virus Disease (EVD) [6]. When a larger or more complete data set is needed, we will use the publicly available Titanic dataset [10]. The distributed machine learning problem will be simulated on a single machine. We will split the available data across simulated devices in a variety of ways to emulate unequal data set size, unequal distribution of features, unequal distribution of labels, unequal distribution of missing entries, and varying degrees of inequality between data sets.
We will evaluate how well each method performs for each split by using it to train and evaluate a set of models using cross-validation. Two baseline methods will also be evaluated, where models are trained once using only

the local data set, and once using all available data. Evaluation metrics, such as root mean squared error or classification accuracy, will be computed on the local test set for each of the trained models. We will visualize these metrics graphically per method for the various splits and models using grouped bar charts, where each bar is a stacked bar chart showing the results obtained using the method under consideration versus either of the baseline methods. An example of such a grouped stacked bar chart is shown in Figure 1. Similar grouped stacked bar charts could also be made per data split to facilitate method selection for a given use case. Pairs of methods could be compared and contrasted by aligning their individual grouped-stacked bar charts in the fashion of a population pyramid (cf. Figure 2). Finally, instead of showing the improvement over the baseline, each stacked bar chart within the grouped stacked bar chart could be used to display the evaluation metric on the test set reached within a certain number of SGD steps, a certain communication traffic per participant, a certain collaborative scope, etc.

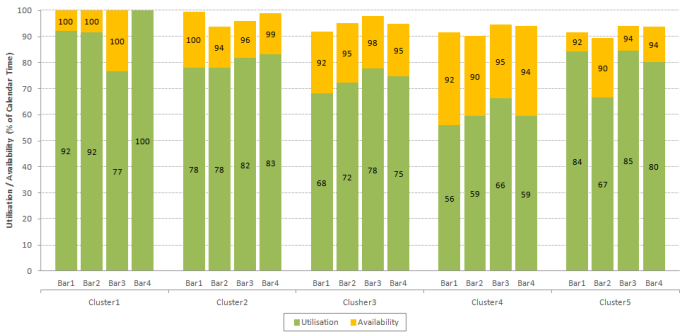

Figure 1: Example of a grouped stacked bar chart. Taken (unchanged) from MLD under CC BY-SA 4.0.

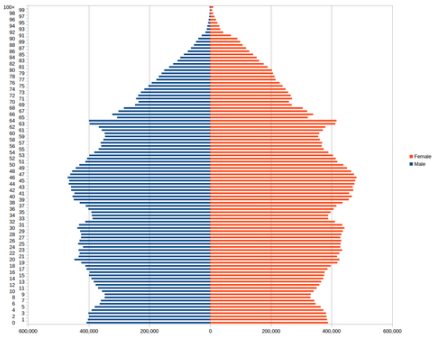

Figure 2: Example of a population pyramid. Taken (unchanged) from SkateTier under CC BY-SA 3.0.

We will also evaluate the methods qualitatively by comparing which data sets are deemed similar to each other for any given task, and analyzing how these patterns arise.

4 Model personalization and data selection

4.1 Setting and Objective

We consider a network of agents u_i, each collecting samples $\mathcal{S}_i = \left\{\mathbf{x}_i^{(n)}, y_i^{(n)}\right\}_{n=1,\dots,N_i}$ from an underlying distribution $\mathcal{D}_i$. One agent, u_0, is called the user and wishes to perform an inference task, such as (regularized) linear or logistic regression, to gain knowledge about the distribution $\mathcal{D}_0$ from which they collect data. For instance, u_0 could wish to predict the label y_0^{new} of a new sample after observing its features $\mathbf{x}_0^{new}$, using some kind of approximation of the conditional probability $p_0\left(y|\mathbf{x}\right)$.
However, the user u_0 knows that the number N_0 of samples they have collected is fairly small for the chosen inference task. Luckily, u_0 believes that *some* of the other agents u_i collect sufficiently interoperable[5] samples $\mathcal{S}_i$ from sufficiently similar underlying distributions $\mathcal{D}_i$, that the *true loss*[6] of u_0's model on $\mathcal{D}_0$ could be reduced by including $\mathcal{S}_i$ in the training process. Unfortunately, u_0 also knows that this is not the case for *all* agents. Simply including all agents' samples equally in the training process therefore would not help reduce the true loss of u_0's model.

[5]Within the context of an inference task, two data sets are called *interoperable* if they collect all features chosen for said inference task in the same way. It is sometimes possible to make two data sets interoperable by constructing the required features from other, originally available features. For example, a data set that records the date of symptom onset and the date of admission can be made interoperable with a data set that records the number of days since symptom onset at admission (w.r.t. an inference task for which this number of days is a relevant feature). For a general definition see [4].

[6]The *true loss* of a model on a distribution is formally defined in Equation 1. It is a quantity that can be obtained only analytically, and only if the model and the distribution are perfectly known. In practice, it is approximated by the *test loss* on a test set consisting of samples drawn from the distribution. This approximation is good if the samples of the test set are "sufficiently" numerous, drawn "sufficiently" independently, and "sufficiently" independent from the samples of the training set.

The task of *model personalization*, in a broader sense, is to give u_0 a training algorithm which discriminates between the available agents in some way to minimize the true loss of the resulting model on $\mathcal{D}_0$.
One class of model personalization methods relies on *data selection*, followed by standard decentralized training on the selected subset of data. The task of data selection is thus to help u_0 select a subset $\mathcal{U}_{train}$ of agents whose samples will be included in the training process. The optimal subset $\mathcal{U}^*_{train}$ is that which minimizes the true loss of the resulting model on $\mathcal{D}_0$.

Model personalization. Formally, we are given:

- A set of agents $u_i \ \forall i \in \mathcal{U} = \{0, 1, \ldots, N\}$
  - Each agent u_i has collected a set of samples (called *data set*): $\mathcal{S}_i = \left\{\mathbf{x}_i^{(n)}, y_i^{(n)}\right\}_{n=1,\ldots,N_i}$
  - Each agent u_i collects their samples from an (unknown) underlying distribution $\mathcal{D}_i$: $\left(\mathbf{x}_i^{(n)}, y_i^{(n)}\right) \overset{i.i.d.}{\sim} \mathcal{D}_i$
  - It is assumed that the label y_0 is not independent of the features $\mathbf{x}_0$ under $\mathcal{D}_0$: $p_0\left(y|\mathbf{x}\right) \neq p_0\left(y\right)$
- A class of models $\mathcal{M}$ s.t. $f : \mathcal{X} \to \mathcal{Y} \ \forall f \in \mathcal{M}$. For instance, $\mathcal{M}$ could be the class of linear models where $f\left(\mathbf{x}\right) = \mathbf{w}\mathbf{x}$, $\mathbf{w} \in \mathbb{R}^D$.
- A loss function: $\ell\left(y, \hat{y}\right)$. For instance, the loss function could be the mean squared error (MSE).

We define the true loss $\mathcal{L}_{\mathcal{D}_0}\left(f\right)$ of a model $f \in \mathcal{M}$ on $\mathcal{D}_0$:

$$\mathcal{L}_{\mathcal{D}_0}\left(f\right) = \mathbb{E}_{\left(\mathbf{x}_0^{new}, y_0^{new}\right) \sim \mathcal{D}_0}\left[\ell\left(y^{new}, f\left(\mathbf{x}^{new}\right)\right)\right] \tag{1}$$

And we attempt to find a training algorithm $\mathcal{A}$ which, given the set $\mathcal{U}$ of users u_i with their individual data sets $\mathcal{S}_i$, the class of models $\mathcal{M}$, and the loss function, will produce the model $f = \mathcal{A}\left(\mathcal{U}, \mathcal{M}, \ell\right) \in \mathcal{M}$ which minimizes the true loss $\mathcal{L}_{\mathcal{D}_0}\left(f\right)$ on $\mathcal{D}_0$.

Data selection In addition to the model personalization problem, we are also given a training algorithm $\mathcal{A}$ which, given a training set $\mathcal{S}$ of samples, produces a model $f_{\mathcal{S}}$: $f_{\mathcal{S}} = \mathcal{A}\left(\mathcal{S}\right), \ \hat{y} = f_{\mathcal{S}}\left(\mathbf{x}\right)$. For instance, this could be ridge regression with a regularization parameter selected among a certain number of candidates to minimize the validation error in 4-fold cross-validation.
We introduce the following notation:

- A subset $\mathcal{U}_{train}$ of agents: $\mathcal{U}_{train} \subseteq \mathcal{U}$
- The corresponding training set: $\mathcal{S}_{train} = \bigcup_{j \in \mathcal{U}_{train}} \mathcal{S}_j$

And we aim to find $\mathcal{U}^*_{train}$:

$$\mathcal{U}^*_{train} = \underset{\mathcal{U}_{train} \subseteq \mathcal{U}}{\arg\min} \ \mathcal{L}_{\mathcal{D}_0}\left(f_{\mathcal{S}_{train}}\right)$$

Fictive example The agents could be individual hospitals dispersed across one or several regions. The aetiology of common, generic symptoms such as fever is highly dependent on geographic location, where the rural setting suffers more faecal-oral and vector borne diseases such as hepatitis A and malaria, while fevers in the urban setting tend to be related to respiratory disease. A medical professional (u_0) might want to train a diagnostic model to help diagnose the patients admitted to their urban hospital. The number N_0 of samples collected at their hospital, however, is too limited to yield a very good model, so u_0 considers using data from other hospitals. Knowing that rural populations suffer from fairly different problems than u_0's urban patients, u_0 pre-selects only other urban hospitals – in other words, u_0 performs manual data selection based on prior medical knowledge. However, u_0 suspects the presence of other confounding variables that could cause the samples collected by *some* of the other urban hospitals to negatively affect the true loss of the trained model on $\mathcal{D}_0$. In this example, the underlying distribution $\mathcal{D}_0$ describes the population of all possible patients of u_0.

	Ndoye	Influence	Gradients
User focus	o		-
Resilience	+		+
Incentives	+		o
Feedback	+		-
Privacy	+		+
Byzantine	-		o
Complexity	+		o

Table 1: Summarizing evaluation of the model personalization methods outlined in the remainder of this section, along the metrics introduced in subsection 3.2.

4.2 Summarizing evaluation of the proposed methods

Add methods from background in table

In the following discussion of proposed methods, we shall assume without loss of generality that the data sets $\mathcal{S}_i$ collected by each agent u_i are perfectly interoperable. Indeed, a non-interoperable data set $\mathcal{S}_i^-$ can always be made technically interoperable by filling in all missing features with arbitrary values[7]. The resulting data set $\mathcal{S}_i^+$ no longer truly corresponds to the distribution $\mathcal{D}_i$, and is likely less useful[8] than it would be if it had directly been sampled from $\mathcal{D}_i$ in an interoperable manner. This loss of usefulness is a consequence of the lack of interoperability in the original data set $\mathcal{S}_i^-$ and can only partially be leviated by filling in the missing features in an informed way (rather than arbitrarily).

4.3 The adapted Ndoye factor.

This method is subdivided into a data selection step, followed by a standard decentralized training task. It differs from the original Ndoye factor, in that the available data sets $\mathcal{S}_i$ are ranked by increasing $\mathcal{D}_i$-to-$\mathcal{D}_0$ transfer loss of the corresponding models $f_i = \mathcal{A}(\mathcal{S}_i)$:

$$\mathcal{L}_i^{transfer} = \mathcal{L}_{\mathcal{D}_0}(f_i) - \mathcal{L}_{\mathcal{D}_i}(f_i)$$

In practice, this transfer loss can only be approximated:

$$\hat{\mathcal{L}}_i^{transfer} = \mathcal{L}_{\mathcal{S}_0}(f_i) - \hat{\mathcal{L}}_{\mathcal{D}_i}^{CV}(f_i)$$

Above, $\mathcal{L}_{\mathcal{S}_0}(f_i)$ represents the average error of f_i on $\mathcal{S}_0$, while $\hat{\mathcal{L}}_{\mathcal{D}_i}^{CV}(f_i)$ corresponds to the approximation obtained by cross-validation. The user u_0 then uses this ranking to select a training set $\mathcal{S}_{train}$, e.g. by selecting $\mathcal{U}_{train}$ as the k agents with the lowest transfer loss or by selecting all agents whose transfer loss is below a given threshold.
It is clear why it is sensible to rank agents by increasing value of $\mathcal{L}_{\mathcal{D}_0}(f_i)$: If it is low, this most likely implies that $\mathcal{S}_i$ is well-suited as training data for the inference task specified by u_0. Unfortunately, $\mathcal{L}_{\mathcal{D}_0}(f_i)$ depends not only on the distribution $\mathcal{D}_i$ (as well as $\mathcal{D}_0$ and $\mathcal{A}$), but also on the size $|\mathcal{S}_i|$ (i.e., the number of samples contained in $\mathcal{S}_i$). However, the size $|\mathcal{S}_i|$ no longer matters when the data sets of several users are combined into $\mathcal{S}_{train}$. Thus, the influence of $|\mathcal{S}_i|$ is removed by considering the transfer loss $\mathcal{L}_i^{transfer}$ instead of just the loss $\mathcal{L}_{\mathcal{D}_0}(f_i)$.

User focus. This method can be explained intuitively as evaluating how successfully an inference made on $\mathcal{S}_i$ can be transferred to $\mathcal{S}_0$. Thus, users can understand the ranking process, albeit without gaining any information as to *why* any particular data set was ranked higher than another. Since the ranking happens

[7] In most cases, one can even fill in the missing features with reasonable, non-arbitrary values. This increases the usefulness of the resulting data set $\mathcal{S}_i^+$.

[8] In the context of the data selection problem, a data set is $\mathcal{S}_i$ is more useful than another data set $\mathcal{S}_j$, if u_i is more likely to be included in $\mathcal{U}_{train}^*$ than u_j.

before any decentralized training, the user is free to either investigate this question by means of further data analysis before manually selecting $\mathcal{U}_{train}$, or select $\mathcal{U}_{train}$ automatically.

Resilience. This method judges each data set as a whole, thus incomplete data is not particularly problematic. Further, unreliably connected agents can be dealt with quite easily, as any existing, resilient decentralized training scheme can be used. The ranking only requires one message from u_0 to each u_i and back. The user can be given the option to select a subset $\mathcal{U}_{train}$ and start training before all other agents have responded.

Incentives. Agents can be rewarded, either proportionally to their rank, or simply based on whether they were included in $\mathcal{U}_{train}$. Finally, u_0 can publish their selected $\mathcal{U}_{train}$, which would enable each agent to check whether their contribution is sufficiently diluted to protect their intellectual property. Since this happens before the training process begins, agents can withdraw from any proposed inference tasks that would impinge too much on their intellectual property.

Feedback. If the model weights of f_i are revealed to the user u_0, then u_0 can use them to spot differences between $\mathcal{S}_0$ and $\mathcal{S}_i$, at the expense of training a model $f_0 = \mathcal{A}(\mathcal{S}_0)$. The other agents can receive extensive feedback based on their ranking, whether or not they were included in $\mathcal{U}_{train}$, and which model weights were particularly different from f_0.

Privacy. $\mathcal{L}_{\mathcal{S}_0}(f_i)$ can be computed via SMC without revealing $\mathcal{S}_0$ to u_i nor f_i to u_0, albeit at a high communication cost.

Byzantine robustness. This method is not robust against byzantine agents. While a byzantine-robust federated or distributed training algorithm can be used, the user selection process relies on each agent to honestly and correctly report their cross-validation loss $\hat{\mathcal{L}}^{CV}_{\mathcal{D}_i}(f_i)$.

Complexity. In addition to the decentralized training process itself, this method merely requires each agent u_i to train one model f_i (entirely locally) and to send its weights, along with a singular loss value, back to the user u_0. If SMC is used to hide the weights of f_i from u_0, then the communication requirements increase drastically.

4.4 Influence-based model personalization

User focus.

Resilience.

Incentives.

Feedback.

Privacy.

Byzantine robustness.

Complexity.

4.5 Gradient-based model personalization

At the core of this method is a novel adaptation of federated training algorithms based on stochastic gradient descent (SGD). It is based on the secure (i.e., privacy-preserving) and byzantine-robust federated SGD-based machine learning framework proposed in [1]. Like said framework, this method relies on round-based gradient descent, and on securely computing distances between gradients through SMC at each round. The novelty lies in how these distances are used:
At each round, each agent u_i (including the user u_0) computes a gradient:

$$\mathbf{g}_i = \nabla_{\mathbf{w}} \mathcal{L}_{\mathcal{S}_i}(f)$$

Above, $\mathbf{w}$ stands for all learned parameters of the current model f. The distance between $\mathbf{g}_i$ and $\mathbf{g}_0$ (normalized) is used as a metric for how different $\mathcal{D}_i$ is from $\mathcal{D}_0$:

$$\mathbf{d}_{i,0}^{rel} = \frac{\|\mathbf{g}_i - \mathbf{g}_0\|}{\|\mathbf{g}_0\|} \in \mathbb{R}^+$$

If the $\mathbf{g}_i$ are full gradients (as opposed to stochastic gradients), then this measure treats all data sets equally, regardless of their size. In other words, individual samples are granted more weight if they come from a smaller data set. This can be mitigated in various ways. For example, each user's contribution could be weighted with two weights: one to favor larger data sets, and a different weight based on $\mathbf{d}_{i,0}^{rel}$. Instead, we propose algorithm 1 to calculate $\mathbf{d}_{i,0}^{rel}$ based on batches of fixed size, as explained after the examples.
Let us analyze what happens in a few examples, if $\mathbf{g}_i$ are indeed the full gradients, considering the entire set $\mathcal{S}_i$, as opposed to stochastic gradients. Let us further assume that each training loss function $\mathcal{L}_{\mathcal{S}_i}(f)$ is convex in the model parameters $\mathbf{w}$.

Example 1 *Suppose the model has just been randomly initialized to f^0 and performs very suboptimally for both $\mathcal{S}_i$ and $\mathcal{S}_0$. Then, the normalized distance between the gradients will be very low. Formally:*

$$\min\left\{\mathcal{L}_{\mathcal{S}_0}\left(f^0\right), \mathcal{L}_{\mathcal{S}_i}\left(f^0\right)\right\} \gg \max\left\{\mathcal{L}_{\mathcal{S}_0}\left(f_{\mathcal{S}_i}\right), \mathcal{L}_{\mathcal{S}_i}\left(f_{\mathcal{S}_0}\right)\right\}, \quad f_{\mathcal{S}_i} = \underset{f \in \mathcal{M}}{\arg\min}\ \mathcal{L}_{\mathcal{S}_i}(f)$$

$$\Rightarrow \quad \|\mathbf{g}_i - \mathbf{g}_0\| \ll \|\mathbf{g}_0\|$$

$$\Rightarrow \quad \mathbf{d}_{i,0}^{rel} \ll 1$$

Therefore, both agents u_i and u_0 are fully included at this stage of the training process, because the distance $\mathbf{d}_{i,0}^{rel}$ is close to 0.
However, as the training process progresses and the model gradually performs better, the distance between the gradients steadily increases. We can analyze the edge cases where f is either the global model or the local model:

Example 2 *Suppose f is the global model for 2 agents, u_0 and u_i:*

$$f = \underset{f' \in \mathcal{M}}{\arg\min}\ \mathcal{L}_{\mathcal{S}_0 \cup \mathcal{S}_i}(f'), \quad \mathcal{L}_{\mathcal{S}_0 \cup \mathcal{S}_i}(f) = \frac{|\mathcal{S}_0|\mathcal{L}_{\mathcal{S}_0}(f) + |\mathcal{S}_i|\mathcal{L}_{\mathcal{S}_i}(f)}{|\mathcal{S}_0| + |\mathcal{S}_i|}$$

$$\Rightarrow \quad \mathbf{0} = \nabla_{\mathbf{w}} \mathcal{L}_{\mathcal{S}_0 \cup \mathcal{S}_i}(f) = \frac{|\mathcal{S}_i|}{|\mathcal{S}_0| + |\mathcal{S}_i|}\left(\frac{|\mathcal{S}_0|}{|\mathcal{S}_i|}\mathbf{g}_0 + \mathbf{g}_i\right)$$

$$\Rightarrow \quad \mathbf{g}_i = -\frac{|\mathcal{S}_0|}{|\mathcal{S}_i|}\mathbf{g}_0$$

$$\Rightarrow \quad \mathbf{d}_{i,0}^{rel} = \frac{\|\mathbf{g}_i - \mathbf{g}_0\|}{\|\mathbf{g}_0\|} = \frac{\left\|\left(-\frac{|\mathcal{S}_0|}{|\mathcal{S}_i|} - 1\right)\mathbf{g}_0\right\|}{\|\mathbf{g}_0\|} = 1 + \frac{|\mathcal{S}_0|}{|\mathcal{S}_i|}$$

In this case, whether (and to which degree) u_i should participate in the training depends on the sizes $|\mathcal{S}_0|$ and $|\mathcal{S}_i|$: Indeed, if $|\mathcal{S}_0| \ll |\mathcal{S}_i|$, then $\mathbf{d}_{i,0}^{rel} \approx 1$, indicating that it would be useful to incorporate u_i further in the training process. This is sensible, considering that the much larger number of samples in $\mathcal{S}_i$ could help reduce

the generalization error substantially. Inversely, if $|\mathcal{S}_0| \gg |\mathcal{S}_i|$, then it is not useful to include $\mathcal{S}_i$ in training and we are better off only using the (much more numerous) samples collected by u_0. Correspondingly, this leads to $\mathbf{d}_{i,0}^{rel} \gg 1$.

Example 3 *Now let us extend the previous edge case to more agents:*

$$
\begin{aligned}
& f = \underset{f' \in \mathcal{M}}{\arg\min}\ \mathcal{L}_{global}\left(f'\right), \quad \mathcal{L}_{global}\left(f\right) = \frac{\sum_{i \in \mathcal{U}} |\mathcal{S}_i| \cdot \mathcal{L}_{\mathcal{S}_i}\left(f\right)}{\sum_{i \in \mathcal{U}} |\mathcal{S}_i|} \\
\Rightarrow \quad & \mathbf{0} = \nabla_{\mathbf{w}} \mathcal{L}_{global}\left(f\right) \\
\Rightarrow \quad & \exists i \in \mathcal{U} : \langle \mathbf{g}_i, \mathbf{g}_0 \rangle < 0 \\
\Rightarrow \quad & \exists i \in \mathcal{U} : \|\mathbf{g}_i - \mathbf{g}_0\| > \|\mathbf{g}_0\| \\
\Rightarrow \quad & \exists i \in \mathcal{U} : \mathbf{d}_{i,0}^{rel} > 1
\end{aligned}
$$

In this setting, 1 is indeed the best lower bound we can prove. To illustrate this, imagine a task with N agents whose gradients are $\mathbf{g}_{i,i>0} = -\frac{1}{N-1}\mathbf{g}_0$ at the global model f. Then $N \to \infty \Rightarrow \mathbf{g}_i \to \mathbf{0} \Rightarrow \mathbf{d}_{i,0}^{rel} \to 1$. In that case, the low distance values suggest we should include them all in the training (in other words: stick close to the global model) because this greatly increases the number of available samples.
In general, in this example, relying on the distance measures implies moving towards the model best suited for a weighted subset of data sets (those with $\mathbf{d}_{i,0}^{rel} \approx 1$) and neglecting data sets that are in stark disagreement with $\mathcal{S}_0$ (those with $\mathbf{d}_{i,0}^{rel} \gg 1$).

Example 4 *Suppose now that f is the local model of u_0:*

$$
\begin{aligned}
& f = \underset{f' \in \mathcal{M}}{\arg\min}\ \mathcal{L}_{\mathcal{S}_0}\left(f'\right) \\
\Rightarrow \quad & \mathbf{g}_0 = \nabla_{\mathbf{w}} \mathcal{L}_{\mathcal{S}_0}\left(f\right) = \mathbf{0} \\
\Rightarrow \quad & \forall i \in \mathcal{U} : \mathbf{d}_{i,0}^{rel} \text{ undefined } (+\infty)
\end{aligned}
$$

Under the stated assumptions of convexity and full gradient descent, all distances $\mathbf{d}_{i,0}^{rel}$ grow without an upper bound as the trained model approaches the local model. In other words, when we move too close to the local model, all data sets start to appear very different from $\mathcal{S}_0$. Consequently, using any decreasing function of $\mathbf{d}_{i,0}^{rel}$ as a similarity metric creates the danger of converging to the local model. One possible strategy to prevent this is to stop training before convergence. Intuitively, training should not be stopped as long as the gradients fit into a D-dimensional cone[9].

Weight Erosion. Due to the problems highlighted by these examples, we propose an adapted SGD scheme in algorithm 1, which can be implemented securely using the two-server framework. Each agent u_i is initially given a weight of $\alpha_i^0 = 1$, and then computes a minibatch gradient[10] at each round. u_i's weight is decreased by a small amount $\Delta\alpha$ that depends on $\mathbf{d}_{i,0}^{rel}$. The batch size is fixed across all agents, such that all samples are initially treated equally in the first epoch. However, as the number of rounds increases, samples from smaller data sets will be seen more often than samples from larger data sets, because each agent uses the same number of samples per round. This leads to an over-representation of samples from smaller data sets. To counteract this, $\Delta\alpha$ is made to depend also on the average number of times each sample in $\mathcal{S}_i$ has been used. algorithm 1 relies on four hyperparameters: The number of rounds r_{max}, the batch size b, the distance penalty factor p_d, and the size penalty factor p_s The former two are already inherent to SGD, while the penalty factors are introduced

[9] If all gradients fit into a cone, then it is possible to improve the loss on all data sets by taking a step in the opposite direction of the axis of the cone.
[10] Alternatively, the agents can compute local SGD updates, which are used in the same way.

to calculate $\Delta\alpha$.

Algorithm 1: WEIGHT EROSION

Data: A set of users $u_i, i \in \mathcal{U}$, with associated data sets $\mathcal{S}_i$, a model class $\mathcal{M}$, and a gradient-based machine learning algorithm $\mathcal{A}$.
Result: A personalized machine learning model f.
Set a number of rounds r_{max}, a batch size b, a distance penalty factor p_d, and a size penalty factor p_s ;
Initialize $\alpha_i^0 \leftarrow 1 \forall i \in \mathcal{U}$;
Randomly initialize the model $f \in \mathcal{M}$;
for r *from* 1 *to* r_{max} **do**
  for $i \in \mathcal{U}$, *starting with* $i = 0$ **do**
    Select a batch of size b from $\mathcal{S}_i$ and compute a gradient $\mathbf{g}_i$;
    Compute the distance $\mathbf{d}_{i,0}^{rel}$;
    $\Delta\alpha \leftarrow \left(1 + p_s \left\lfloor \frac{(r-1)b}{|\mathcal{S}_i|} \right\rfloor\right) p_d \mathbf{d}_{i,0}^{rel}$;
    $\alpha_i^r \leftarrow \max\left\{0, \alpha_i^{r-1} - \Delta\alpha\right\}$;
  end
  $\bar{\mathbf{g}} \leftarrow \frac{\sum_{i \in \mathcal{U}} \alpha_i^r \mathbf{g}_i}{\sum_{i \in \mathcal{U}} \alpha_i^r}$;
  Update the model f based on $\bar{\mathbf{g}}$;
end

User focus. With this approach, the user does not learn why a certain subset of the available data sets were selected, even if they understand the selection process. They learn nothing beyond "These are the data sets for which $p^i(y|\mathbf{x})$ is similar to the local set". To make things worse, the selection is only revealed to them after the model has been trained.

Resilience. Like the algorithm proposed in [1], algorithm 1 does not break if users appear or disappear between subsequent rounds. If this is expected, $\Delta\alpha$ can be modified by replacing r with the number of rounds in which u_i has actually participated. When a new agent appears after the first round, their weight should not be initialized to 1, but rather to the mean or median weight of all agents.

Incentives. Agents could be rewarded according to how many rounds they participated in, or according to the sum of their weights in each of these rounds. The examples give us an intuition that the result of algorithm 1 lies somewhere between the global model and u_0's local model. As long as the number and data set size of other users in $\mathcal{U}$ is sufficient to dilute each individual agent's contribution, the algorithm therefore does not impinge anyone's intellectual property. This condition on $\mathcal{U}$ can be verified by each agent before engaging in a round of training.

Feedback. Unfortunately, algorithm 1 does not produce any information that lends itself to giving feedback on the user's data collection. The latter could be used

Privacy. A secure (i.e., privacy-preserving) and byzantine-robust SGD protocol based on a framework with two non-colluding servers is proposed in [1], which relies on secret sharing to compute and reveal the distances between gradients through SMC. Weight erosion can be seamlessly integrated into that protocol as an aggregation rule, instead of the byzantine-robust aggregation rule KRUM [2].

Byzantine robustness. Since weight erosion would *replace* the byzantine-robust aggregation rule in the setting of [1], the byzantine robustness property is lost. While byzantine inputs can negatively affect the training process, a byzantine agent u_i would likely see their weight α_i decline fast.

Complexity. This method relies on computing distances between gradients at each round. However, contrary to [1], only $|\mathcal{U}|$ distances are computed instead of $|\mathcal{U}|^2$.

5 Application case study

We choose to implement the Weight Erosion scheme from subsection 4.5. Why? In this section, ...

5.1 Implementation

The Weight Erosion scheme (algorithm 1) was implemented on top of an early version of a 2020 `Google colab` project by Sai Praneeth Karimireddy (cf. link[11]), where he created a framework for federated learning simulations based on `JAX` and `Haiku` [3, 7].
How exactly did I do it / What exactly did I do / implemented on top of an early version of Sai Praneeth Karimireddy's Jax FL project on colab https://tinyurl.com/Karimireddy-Jax-FL

5.2 Case study

Present inference problem(s) and data set(s)

5.3 Results

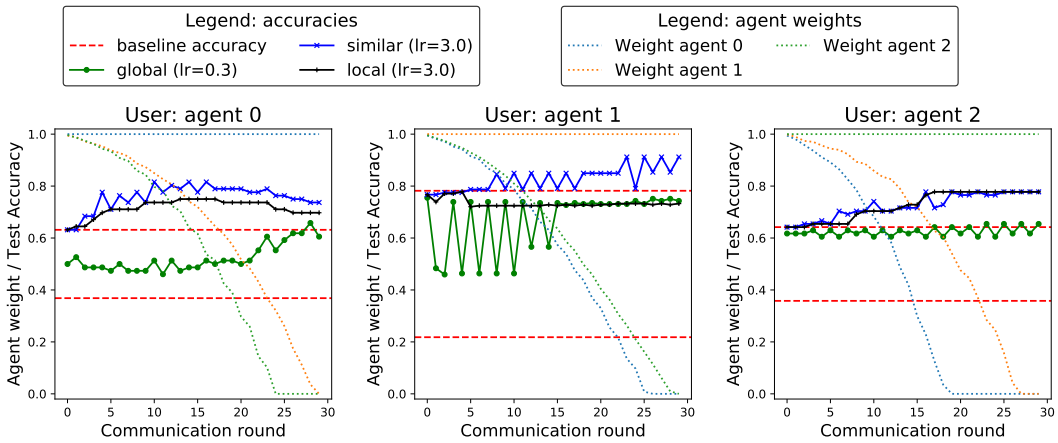

Figure 3: Simulation of federated training on the Ebola data set, split across 3 agents by age group:
agent 0: 0 - 20 years old / agent 1: 21 - 40 years old / agent 2: 41+ years old.
Accuracy on the user's test set.
`Global`: the model trained on all agents' data without Weight Erosion,
`Weight Erosion`: the model trained using all agents with Weight Erosion,
`Local`: the model trained only on the user's data.
In red (dashed), the accuracy obtained by predicting always EVD(+) or always EVD(-). The pointed lines represent the weight α_i of each agent. The learning rates were tuned independently for each aggregation scheme (`Global`, `Weight Erosion`, `Local`), and they are displayed in the legend.
$p_d = 0.01$
$p_s = 0.2$
One batch of 125 samples per round and agent (The user's training set contains only one such batch, the other agents' training sets contain two batches each.)
seed = 278

[11] Short link to the project: `https://tinyurl.com/Karimireddy-Jax-FL`

6 Outlook

- Implement other methods
- Refine gradient-based similarity some more, try it out on other problems
- Rigorously define similarity

6.1 Resilience test - If time allows

Additionally, we will emulate a set of participants who are spread across time zones and experience various levels of connection bandwidth and reliability by specifying individual (non-arbitrary) random connection/disconnection patterns for each simulated device. We will use these to test the resilience of the proposed methods.

Mention EDP application. E.g., when talking about the datasets used.

References

[1] Anonymous Authors. "Secure Byzantine-Robust Machine Learning". In: (2020).

[2] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. "Machine learning with adversaries: Byzantine tolerant gradient descent". In: *Advances in Neural Information Processing Systems*. 2017, pp. 119–129.

[3] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.1.55. 2018. URL: http://github.com/google/jax.

[4] *Definition: Interoperability*. URL: http://interoperability-definition.info/en/ (visited on 06/01/2020).

[5] Moritz Hardt and Guy N Rothblum. "A multiplicative weights mechanism for privacy-preserving data analysis". In: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE. 2010, pp. 61–70.

[6] Mary-Anne Hartley et al. "Predicting Ebola infection: A malaria-sensitive triage score for Ebola virus disease". In: *PLoS neglected tropical diseases* 11.2 (2017).

[7] Tom Hennigan et al. *Haiku: Sonnet for JAX*. Version 0.0.1. 2020. URL: http://github.com/deepmind/dm-haiku.

[8] Peter Kairouz et al. "Advances and Open Problems in Federated Learning". In: *arXiv:1912.04977 [cs, stat]* (Dec. 10, 2019). arXiv: 1912.04977. URL: http://arxiv.org/abs/1912.04977 (visited on 06/04/2020).

[9] Miaofeng Liu et al. "Reinforced Training Data Selection for Domain Adaptation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics, 2019, pp. 1957–1968. DOI: 10.18653/v1/P19-1189. URL: https://www.aclweb.org/anthology/P19-1189 (visited on 03/27/2020).

[10] Miaofeng Liu et al. *Reinforced Training Data Selection for Domain Adaptation*. 2019. DOI: 10.18653/v1/P19-1189. URL: https://www.aclweb.org/anthology/P19-1189 (visited on 03/27/2020).

[11] Yishay Mansour et al. "Three Approaches for Personalization with Applications to Federated Learning". In: *arXiv:2002.10619 [cs, stat]* (Feb. 24, 2020). arXiv: 2002.10619. URL: http://arxiv.org/abs/2002.10619 (visited on 06/03/2020).

[12] Yishay Mansour et al. "Three Approaches for Personalization with Applications to Federated Learning". In: *arXiv:2002.10619 [cs, stat]* (Feb. 24, 2020). arXiv: 2002.10619. URL: http://arxiv.org/abs/2002.10619 (visited on 06/03/2020).

[13] Frank D McSherry. "Privacy integrated queries: an extensible platform for privacy-preserving data analysis". In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 2009, pp. 19–30.

[14] Ricardo Mendes and João P Vilela. "Privacy-preserving data mining: methods, metrics, and applications". In: *IEEE Access* 5 (2017), pp. 10562–10582.

[15] Xu Miao. *Personalized machine learning models*. US Patent App. 14/105,650. June 2015.

[16] Tudor Miu, Paolo Missier, and Thomas Plötz. "Bootstrapping personalised human activity recognition models using online active learning". In: *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*. IEEE. 2015, pp. 1138–1147.

[17] Mohamed Ndoye et al. "Collaborative privacy". In: (2020). URL: https://www.mndoye.com/collaborativeprivacy.pdf.

[18] Adam Richardson, Aris Filos-Ratsikas, and Boi Faltings. "Rewarding High-Quality Data via Influence Functions". In: *arXiv preprint arXiv:1908.11598* (2019).