Approaches to address the Data Skew Problem in Federated Learning

Dinesh C. Verma[a], Graham White[b], Simon Julier[c], Stephen Pasteris[c], Supriyo Chakraborty[a], and Greg Cirincione [d]

[a]IBM T. J. Watson Research Center, PO Box 218, Yorktown Heights, NY, U.S.A.
[b]IBM Research, Hursley Park, Hursley, SO212JN, U.K.
[c]University College, 66 Gower St, London, WC1E6BT, U.K.
[d]U.S. Army Research Lab, 2800 Powder Mill Road, Adelphi, MD, U.S.A.

ABSTRACT

A Federated Learning approach consists of creating an AI model from multiple data sources, without moving large amounts of data across to a central environment. Federated learning can be very useful in a tactical coalition environment, where data can be collected individually by each of the coalition partners, but network connectivity is inadequate to move the data to a central environment. However, such data collected is often dirty and imperfect. The data can be imbalanced, and in some cases, some classes can be completely missing from some coalition partners. Under these conditions, traditional approaches for federated learning can result in models that are highly inaccurate. In this paper, we propose approaches that can result in good machine learning models even in the environments where the data may be highly skewed, and study their performance under different environments.

Keywords: Federated Learning, AI, Data Skew, Coalition Operations, Future Battlespace

1. INTRODUCTION

In many scenarios where a machine learning model needs to be created from training data, the training data may be distributed across multiple locations in a wide-area network, and cannot be moved easily to a central location. Some of the reasons for preventing the movement of training data include: the lack of sufficient network connectivity, or policy/regulatory restrictions which prevent free unfettered movement of data. In order to create a model that can incorporate insights form all the different locations where the training data is present, the alternative approach is for each site to train a local model on data available to it. The models can be exchanged and merged together to create a final model that incorporates the insights present from the data at all the sites. We refer to this approach for building models as federated learning.

Federated Learning has been shown to be useful in many contexts such as consortia of health-care providers,[1] sharing insights among multiple government agencies,[2] coalition operations,[3] wireless communication networks ,[4] etc. However, most of the current work in federated learning deals with environments when the training data for the learning is randomly split across all the sites. In real world, the training data is usually skewed heavily with data having widely varying characteristics at different sites. In those cases the naive implementation of federated learning algorithms will result in an erroneous model. Proper safeguards need to be taken in federated learning to protect against this very realistic scenario.

In this paper, we look at three classes of machine learning problem and discuss how data skew can impact the model built using federated learning in each of them. We propose two effective mechanisms which can result in federated learning creating the correct model in the presence of the skewed data. We refer to these mechanisms as (i) bounds-aware fusion and (ii) bounds-expanding data exchange.

The rest of the paper is structured as follows. The next section provides an overview of the architecture for federated learning assumed in this paper, which is followed by three sections looking at the problem of data skew in machine learning based function estimation, classification on tabular data, and classification problems on non-tabular data. Finally, we summarize our results and identify areas for future research.

2. FEDERATED LEARNING OVERVIEW

For this paper, we assume a scenario for federated learning which is as shown in Figure 1. There are multiple sites that hold different portions of the training data needed to create an AI model. The different sites are connected by means of a slow network, or otherwise unable to send data to each other due to regulatory or policy constraints. However, the sites are allowed to create AI models using their local data, and exchange the AI models with each other. One of the sites acts as a fusion site, whose task is to take each of the models provided by the other sites, and come up with an integrated model which is obtained by merging all of the models together.

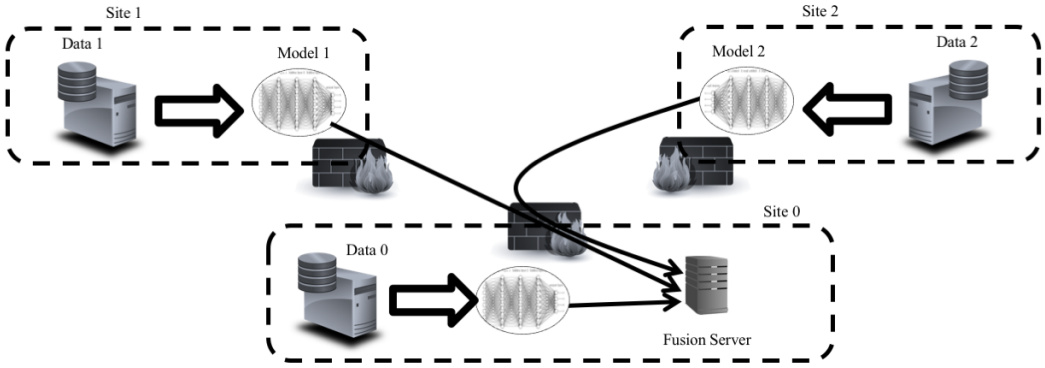

Figure 1. The typical environment for federated learning.

The environment shown in Figure 1 can be encountered in many scenarios. As an example, the three sites could be members of a military coalition which have collected different surveillance footage, and want to combine the models they have each built individually to have a better model to detect enemy operations. In another case, the three sites could be hospitals who have medical records of their patients, and want to mine the data collaboratively but are unable to share the data due to regulatory restrictions. In yet another scenario, each site could be the national site of a large bank which wants to mine the overall data for risk analysis, but is unable to transfer huge amounts of data to each other due to the costs and risks of moving data across large overseas links.

In each of these scenarios, one can envision a Fusion Agent at different sites, which interacts with a Fusion Server. The Fusion Agent is responsible for learning the model locally, and for interacting with the Fusion Server. The interaction with the Fusion Server can lead to coordination on some aspects of learning, e.g. all sites may choose to learn a model with the same architecture. Having the same architecture means each site is learning the same type of model e.g. all are learning a decision tree or all are learning a neural network with the same number of neurons, layers and identical activation functions. Assuming a homogeneous architecture of models simplifies the task of fusion, although it is possible to convert some types of architectures into another.[5,6]

The resulting architecture is that of a standard client-server architecture, where the client is at each of the sites, responsible for training of the model on the local data, and the server is responsible for the fusion of the models from each of the clients. We assume that the server also has access to some validation data, which can be used for analysis of the models sent by the different Fusion Agents. Site 0 has both a Fusion Client and a Fusion Server, and part of the training data at Site 0 can be used as validation by the Fusion Server. The abstracted architecture in this case would look like that in Figure 2.

In the environment shown in Figure 2, we can have the Fusion Server train many different types of models. Among these models are decision trees, clusters, rule-engines, Gaussian Mixture Models, Support Vector Machines, Neural Networks.[7] The models trained in this manner can be used for a variety of AI-enabled tasks. In this paper, we consider two of the tasks, function modeling and classification.

Function modeling is the task of estimating a function which can predict a numeric output from a given set of numeric inputs, while classification is the task of mapping a set of inputs into one of more discrete categories.

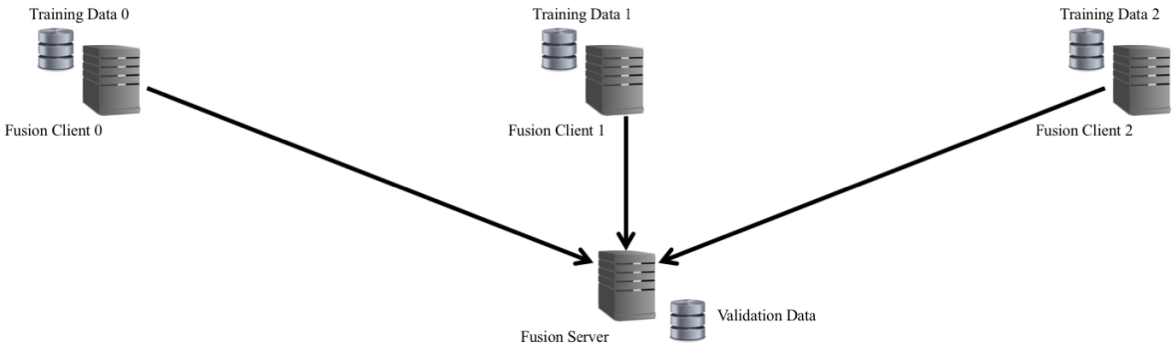

Figure 2. The client-server model for federated learning.

The classification task can be done on input that is tabular, or on input that is a non-tabular signal (e.g. a video, an image or a sound clip). Thus, there are three tasks associated with federated learning that we consider in the subsequent sections of this paper.

3. FUNCTION ESTIMATION

One common instance of a machine learning based task is the estimation of the function that best matches the available training data. We assume that the training data consists of multiple data points, each consisting of some N features $x_1, x_2 \ldots x_N$ and an output y. The goal of function estimation is to find the function f which best satisfies the relationship that $y = f(x_1, x_2 \ldots x_N)$. Function estimation can be viewed as a fundamental approach which characterizes a large number of machine learning problems. Several approaches in AI, such as training a neural network, training a decision tree, training a decision table, etc. can be viewed as alternative mechanisms for estimating a function.

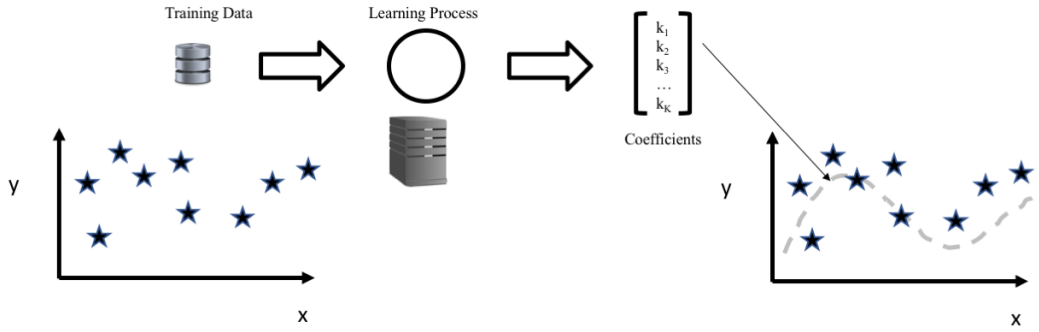

Figure 3. The abstracted operation for function estimation.

While many different algorithms for function estimation exist, and the choice of specific function estimation depends on the domain being studied, we can model the general task of function estimation as a mapping of the available training data set to a set of K coefficients as shown in Figure 3. The value and semantics of K depends on the type of curve-fitting being done.

As an example, if the function estimation is being done by means of linear regression among the features and the output variable, then there will be $N + 1$ coefficients (i.e. $K = N + 1$, and these would be the $N + 1$ values of $\alpha_0, \alpha_1 \ldots \alpha_N$ which provides the best fit for the relationship:

$$y = \alpha_0 + \sum_{i=1}^{i=N} \alpha_i x_i$$

If the function estimation is being done in a non-linear manner, then an alternative approach would be the estimation of a model where $K = (M + 1)N$, and the coefficients would be β_i, j which provide the best estimate for the relationship:

$$y = \sum_{i=1}^{i=N} \beta_0, ix_i + \sum_{i=1}^{i=N} \beta_i, jx_i{}^j$$

Yet another approach for function estimation use the kernel method, in which a set of functions (kernel functions) are assumed to define the basis for the space of all functions of interest. The estimated function would be computed as a linear combination of the kernel functions. The kernel functions $k_1 \dots k_K$ are functions defined on the features $x_1 \dots x_N$ and their linear combinations would cover all the other functions of interest in the curve-fitting exercise. The goal them is to come up with the K coeffients $\gamma_1 \dots \gamma_K$ which provides the best estimate for the relationship:

$$y = \sum_{i=1}^{i=N} \gamma_i k_i(x_1 \dots x_N)$$

As a simple example, for an output y with only one feature x, a kernel function set of $k_i(x) = \sin(ix)$ may be defined, and the resulting function be estimated as:

$$y = \sum_{i=1}^{i=K} k_{i,1} \sin(ix) + k_{i,2} \cos(ix)$$

Alternatively, when the kernel function can be defined as $k_i(x) = (x - a)^i$, where a is a pre-determined constant and the resulting function be estimated as:

$$y = \sum_{i=1}^{i=K} (x - a)^i$$

Regardless of the approach used for function estimation, the net result is the calculation of the K coefficients required to estimate the right function.

The problem of function estimation arises in many real world situations, ranging from determining the right control and configuration settings for physical control of equipment and machinery, make financial decisions such as estimating the default risk associated with a loan, predicting the value of a house, estimating the configuration that will maximize the performance of a data center or network, etc. In many of these real world situations, the training data can not be shared freely. As an example, multi-national finance and telecommunications companies may be prevented from moving data across national boundaries. In order to deal with such split in training data, federated learning techniques need to be deployed.

3.1 Federated Learning for Function Estimation

In order to deal with the challenge of distributed learning, a federated learning approach would be to estimate the function independently at each of the fusion clients, and then to aggregate those functions at the fusion server site.

The abstract process for fusion can be seen in Figure 4. Each of the site has some local data. At each iteration, each of the site selects a random subset of the local data, and runs a local function estimation based on that randomly selected data subset. This results in each fusion client computing a set of K coefficients. These K coefficients are then averaged together by the fusion server. The process is repeated for a predetermined number of iterations, with each iteration done over a randomly selected subset of the data at each site.

An intuitive rationale for using the averaging process is shown in Figure 5. At each site, the function estimation which is done over the available number of points results in an estimation of the ground truth of the function. The ground truth is shown as solid line in the left-most side of the figure. Each site has a set of points shown as stars in the figures in the middle and the right, and each site is estimating an approximation to the ground truth based on its learning algorithm. This learning process results in an estimated function at each site which is shown by the dashed line of each site.

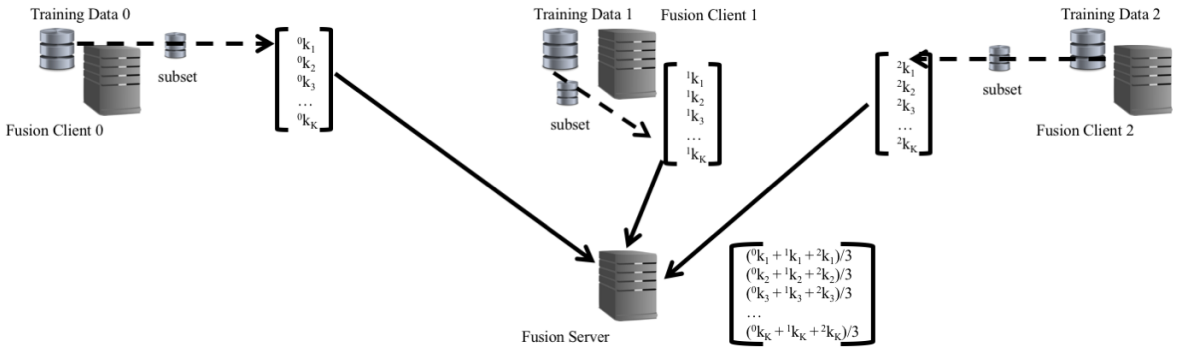

Figure 4. One iteration for the Federated Version of Function Fusion.

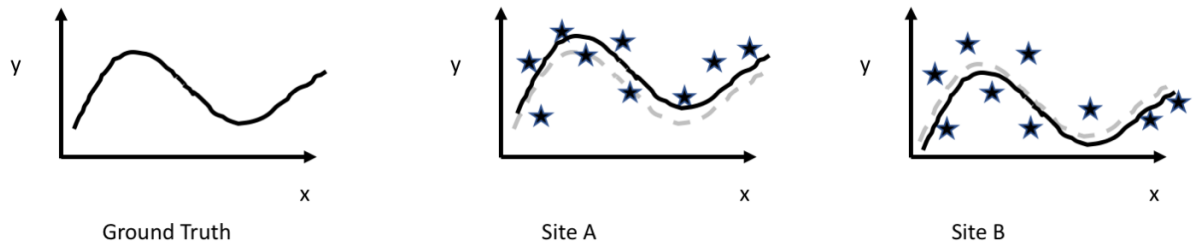

Figure 5. Justification for Operation of Federated Version of Function Fusion with Evenly Distributed Data.

If each site is learning the function independently, then the estimate would differ from the ground truth by means of an error. If we have the averaging done over several different sites, and we average for a large enough number of times, then we will expect those errors in the estimates to cancel out, resulting in a better approximation to the ground truth.

A random subset selection at each of the sites is required so that the averaging process has a higher chance of eliminating the errors from the ground truth. If there are only a small number of sites, e.g. 3 sites, then averaging the 3 errors is not likely to result in elimination of the error. However, if averaged over a larger number of iterations, e.g. 100 averages, the error is likely to be eliminated. Therefore, taking random subsets of the data at each site, and averaging them together is more likely to result in an accurate estimation.

3.2 Data Skew Issue in Function Estimation

The approach described in Section 3.1 works well is the function estimation is done over data points that are representative of the entire range of the function. However, if the data points that are collected at each of the site refer to different ranges in the value of the function, then the averaging process is likely to lead to an incorrect result.

The situation with data skew is illustrated in Figure 6. Two sites are each estimating the function from the data available. The ground truth is shown on the bottom left-hand side. However, if the data available at each site has only got a portion of the function of interest, and the function does not have the same form at those portions, the averaging process can result in a very erroneous process. In the illustrative case, since one site will be estimating a positive correlation between the dependent and independent variable, the other site will be estimating a negative correlation, this would indicate that the averaging process may result in a relationship which will be completely wrong, as shown in the estimate on the right hand side bottom.

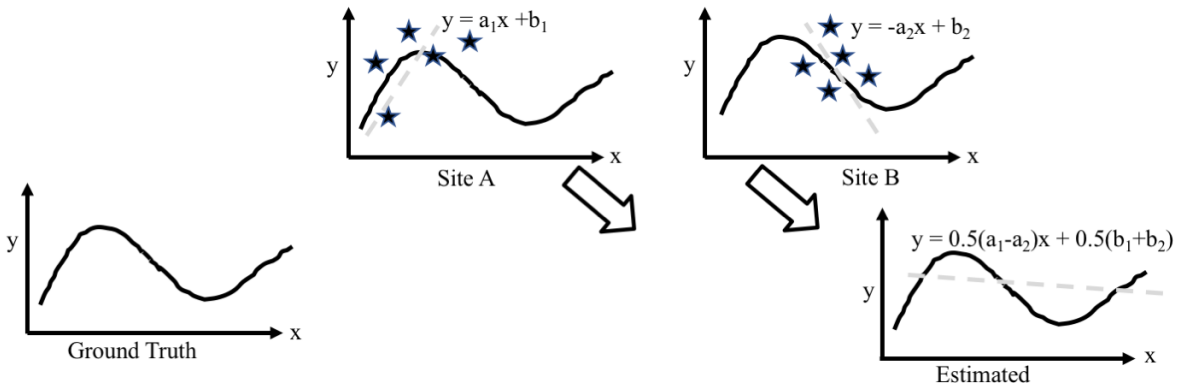

Figure 6. Problem in Function Estimation Federation with Data Skew

The problem in performing the fusion is that the averaging is being done for function estimates that were applicable under different conditions. The averaging of the different estimated functions is the right approach when the estimates are applicable for a common region of dependent variables, and not the right approach when the estimates are for different regions of the space.

In order to deal with data skew, the fusion server needs to be aware of the region of applicability of the estimate. In this case, the fusion server can determine the regions where the estimates are applicable, and do piece-wise fusion, i.e. average only those parts of the system where the estimates from different sites are valid.

3.3 Bounds-Aware Fusion

When the aggregation is done in this manner, one could create an aggregation of estimated functions in areas where there is a common overlap, but avoid areas where function estimation is done over different ranges of the functions. In this case, the estimation would result in an estimate which is closer to the ground truth.

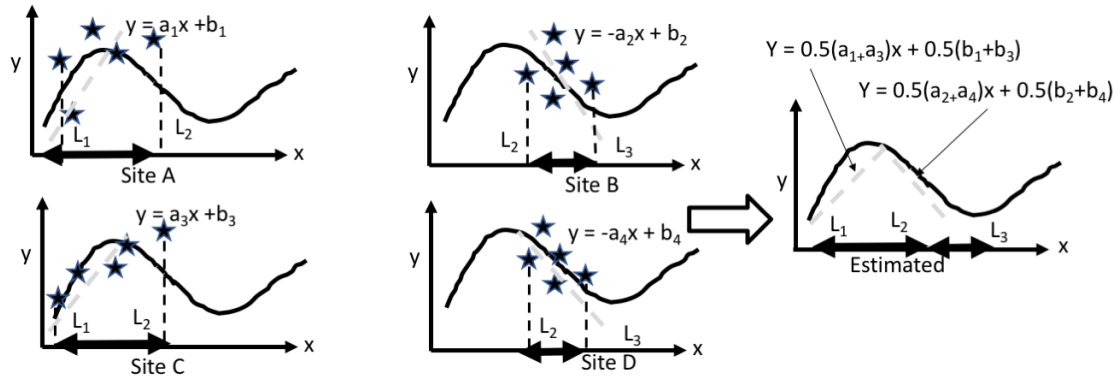

Figure 7. Handling the data skew problem by being aware of the bounds

The right way to perform function fusion with data skew is illustrated in Figure 7. Four sites are shown in the figure, and the function estimated at each site is shown using the grey shaded line. The dark black line shows the estimated function. Each site also finds the limits under which the estimate is valid, shown as limits L_1 and L_2 for sites A and C, and as L_2 and L_3 for sites B and D respectively. For function estimation, models from sites A and C are fused together, and models from sites B and D are fused together in the region of applicability to result in the aggregate function shown at the very right in Figure 7. The function estimate is different between the region $L_1 - - L_2$ and $L_2 - - L_3$. This estimate is much closer to the ground truth than the blind aggregation.

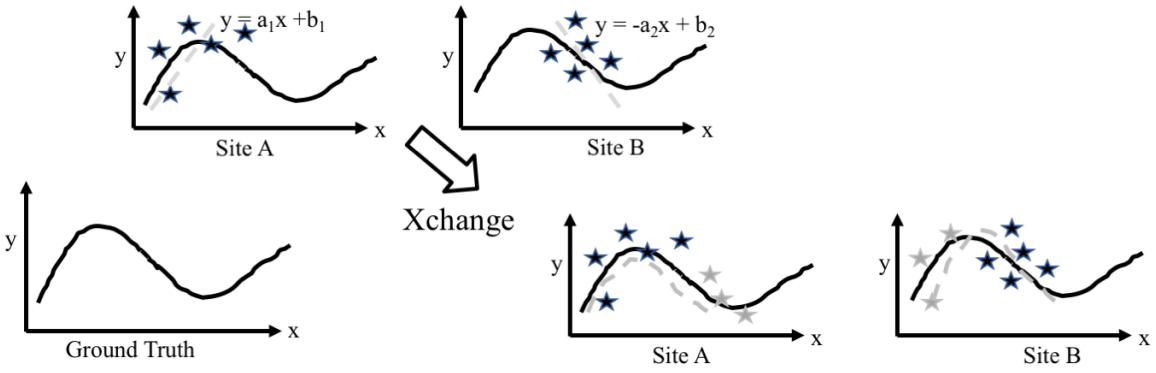

Figure 8. Handling the data skew problem with data exchange

3.4 Bounds Expanding Data Exchange

An alternative approach to handle data skew would be to exchange some data so that the bounds of each of the sites can be expanded so that they can become the same. When different sites determine their bounds, they can ask other sites for a few representative data points that extend their boundaries of applicable region to cover missing points. These extra points can then be used to extrapolate the function in a better manner, and the aggregation, which is now done over the same common region, can be closer to the ground truth. The abstract approach for using this is shown in Fig. 8

The data exchange may be done by means of exchange of a selected number of points, or it may be done in a more sophisticated manner. As an example of a more sophisticated approach, each site may find out the bounds of the input space over which it has data, which other sites do not. Instead of sending some points out to the other sites, it can train a function which can predict the values on that bounds, and exchange that function with the other site. In many cases, this function exchange may be more efficient.

3.5 Experimental Validation

To check for the effectiveness of the two approaches for handling data skew in practice, let us consider a simple experiment in the analysis of the characteristic curve of a electronic devices. An electronic device can be tested in multiple sites to determine its characteristic curves. Due to differences in the measurement equipment, not all of the sites may measure the characteristic for all possible ranges for which the device may be operating. Thus, different sites may be measuring the results for a different ranges of the independent parameters.

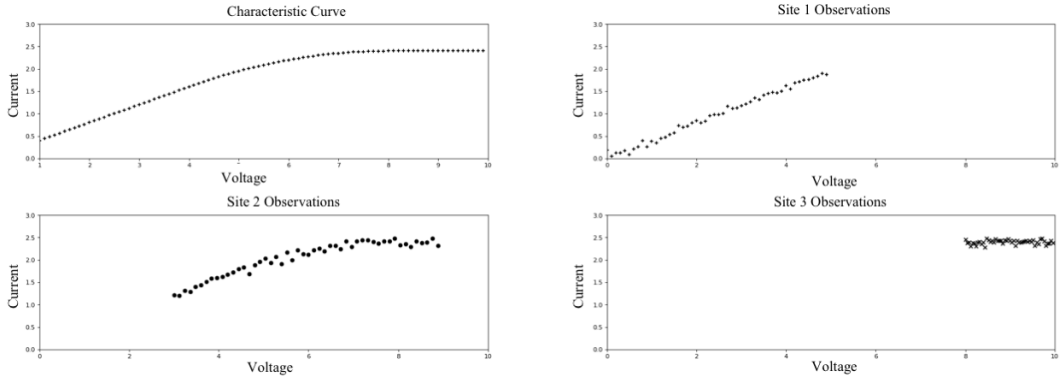

Figure 9. The upper left hand figure shows the characteristic curve of a MOFSET. Three sites measure it for different range of voltages, which are slightly overlapping, and result in the three set of measurements shown in the other three figures.

The characteristic curve for a typical metal-oxide semiconductor field-effect transistor (MOFSET) is shown in Fig. 9. Let us assume that instances of a batch production of this electronic device is tested at different sites, each of which are operating independently, and chose to test the device over different ranges of the independent parameter, which is the voltage in this case. The dependent variable, current, is measured at each of these points. The three sites may not be able to share the raw data, but are willing to share the results. Although this example may seem somewhat artificial, there are many instances in which physical phenomenon is measured at different intervals by different groups, and the information can not be easily shared due to a variety of considerations.

The performance of the three difference algorithms discussed earlier is shown in Figure 10. The first algorithm (Naive algorithm) simply divides each sited data set into multiple batches, and tries to combine them together. Each site runs a non-polynomial best fit algorithm on its locally available data. A quadratic equation was attempted to fit into the ground truth since all sites were aware that a quadratic equation could capture the relationship in this case. However, given that the different parts of the measurements have different relationships, the resulting fused model is fairly inaccurate. However, exchanging some information (about 10%) in this case, allows the sites to extend their applicability bounds, and the naive fusion algorithm works well after this data exchange. An equally good algorithm is obtained when the model being fit is aware of the bounds of applicability of each of the sites, and combines them only in the areas where they overlap.

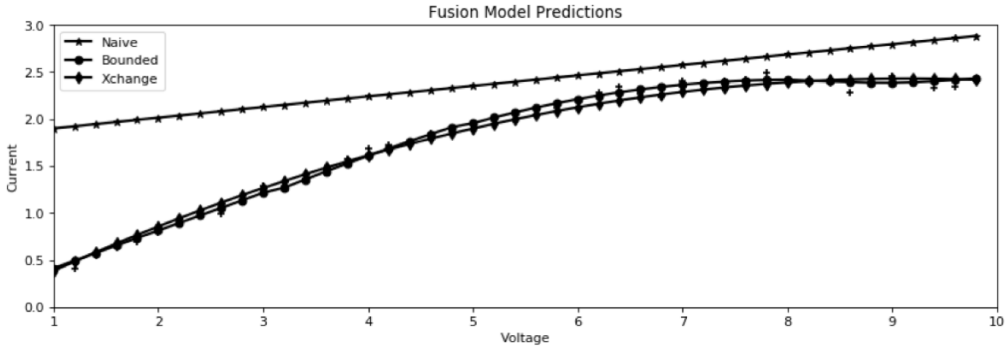

Figure 10. The results of the different fusion algorithms on the MOFSET data. Ignoring the ranges for which data has been collected results in an inaccurate model marked as the Naive algorithm. The ground truth, combined from all sites is marked with + symbols. The other two approaches, being aware of bounds, and exchanging small amounts of data, result in a more accurate approach.

In summary, there are two basic approaches for dealing with data skew in function estimation. The first, bounds-aware fusion, consists of computing applicable limits and aggregating AI models from different sites only when they are applicable in the same region of the input space. The second, bounds-expanding exchange, is to exchange a small amount of data so that all sites have the same applicable region of applicability for their estimated models. Both of these approaches result in an fused model that is more accurate than the Naive approach of fusing parameters.

4. CLASSIFICATION ON TABULAR DATA

Another common instance of a machine learning task is the classification of an input from tabular data. In this instance we assume a set of tabular data (as may commonly be found in an spreadsheet or a relational database) is used for training a classifier. Most columns of this table denote one or more features, and one column denotes the output or class. Subsequently, predictions of class/output can be made from the model on previously unseen rows of the table. In this scenario each column in the tabular data set can be considered as a feature upon which the model could be trained. It may additionally be the case that feature engineering, feature normalization or feature regularization may need to take place upon the tabular data. As set out in Section 2, we assume a homogeneous architecture such that the same process is applied to the data at each site and fed into the same training algorithm.

4.1 Data Skew Issues for Tabular Data

The approach used to tackle the problem of data skew in the domain of classification on tabular data may differ depending on the type(s) of skew found within the data. We propose four broad tabular data skew issues where: (1) there is a data imbalance between partners; (2) one or more partners has missing classes; (3) one or more partners have missing features; (4) one or more partners have missing values. These issues need not occur in isolation and may be found in any combination.

4.1.1 Data Imbalance

This data skew issue occurs when one or both partners have vast amounts more (or less) training data than another partner. For example, it may be the case that one partner has thousands of training samples for class 1 and the other partner has a few tens of training samples for this class. When this is the case, the resulting model built via federated learning can be very poor[2] and techniques are needed in order to address data imbalance issues in this environment.

4.1.2 Missing Classes

This data skew issue occurs when one or both partners have training data representing a class that is not present within the data of the other partner. Figure 11 illustrates the case where at site 1 there is a complete set of features and values but no training data exists for class 4. Similarly, at site 2 there is a complete set of features and values but no training data exists for class 3. These missing classes can occur in any volume or combination. When used in isolation this would result in site 1 not being able to classify class 4 from their model and site 2 not being able to classify class 3 from their model. Techniques are needed such that both site 1 and site 2 are able to classify samples from classes missing in their training data.

Site 1

Feature 1	Feature 2	Feature 3	Class
S1 FA	S1 FB	S1 FC	1
S1 FD	S1 FE	S1 FF	2
S1 FG	S1 FH	S1 FI	3

Site 2

Feature 1	Feature 2	Feature 3	Class
S2 FA	S2 FB	S2 FC	1
S2 FD	S2 FE	S2 FF	3
S2 FG	S2 FH	S2 FI	4

Figure 11. Missing classes in tabular data.

4.1.3 Missing Features

This data skew issue occurs when one or both partners have training data containing features (columns of tabular data) that are not present within the data of the other partner. Figure 12 illustrates the case where site 1 has features 1 and 2 but is missing feature 3. Similarly, site 2 has features 1 and 3 but is missing feature 2. These missing features can occur in any volume or combination. When used in isolation this would result in a model being created that does not "know of" or require the missing feature(s). Techniques are needed such that both site 1 and site 2 are able to classify from samples with missing features.

Site 1

Feature 1	Feature 2		Class
S1 FA	S1 FB		1
S1 FD	S1 FE		2
S1 FG	S1 FH		3

Site 2

Feature 1		Feature 3	Class
S2 FA		S2 FC	1
S2 FD		S2 FF	2
S2 FG		S2 FI	3

Figure 12. Missing features in tabular data.

4.1.4 Missing Values

This data skew issue occurs when one or both partners have training data where some values are missing throughout their feature set. Figure 13 illustrates the case where site 1 is missing feature 2 for class 1 and feature 3 for class 3. Similarly, site 2 is missing feature 3 for class 2. When used in isolation a technique would be required to address this problem and similarly for federated learning, techniques are needed such that missing values can be addressed.

Site 1

Feature 1	Feature 2	Feature 3	Class
S1 FA		S1 FC	1
S1 FD	S1 FE	S1 FF	2
S1 FG	S1 FH		3

Site 2

Feature 1	Feature 2	Feature 3	Class
S2 FA	S2 FB	S2 FC	1
S2 FD	S2 FE		2
S2 FG	S2 FH	S2 FI	3

Figure 13. Missing values in tabular data.

4.2 Addressing Data Skew in Tabular Data

The natural choice to address the data skew issues described in Section 4.1 would be to transfer training data between partners. However, with the assumption that this is not possible due to the reasons described in Section 2, other approaches are needed. These approaches need to address the issues in alternative ways that conform to the requirements of low bandwidth connections between partners or regulatory issues.

4.2.1 Bounds-Aware Fusion

As in the case of function-estimation, a logical choice for addressing the data skew issues in tabular data would be to identify common regions in the feature-space between data sets at different sites and attempt to fuse the model only among sites that have data on the common regions. using the common data. In this approach, a method for discovering where there is overlap in training data between partners would be required. This could be implemented on the fusion server as a prior requirement before federated training begins. It would be implemented such that the fusion server would require information from each partner in advance of training that describes the range of data they have for each feature. This would allow a simple calculation to take place to work out where the overlap exists in the feature space and information sent back to the partners from the fusion server would configure the training to take place using the discovered overlap.

This method would result in a set of common trained models for the benefit of all partners that would take advantage of the combined training data. For regions of the feature space where there would be overlapping data among different sites, an AI model which fuses the model trained on different sites will be used. In regions of the feature space where there is no overlap, i.e. only one site has data, that model will be used. For performing inference on a point which is not in any of the regions of the training data set, the nearest AI model would be used.

An example of the models which will be used among three different sites when classification is done on a site with two independent features is shown in Fig. 14. The feature region on which an AI model can be trained for each of the three sites is shown at the top of the figure. The bottom shows the resulting fusion which is bounds-aware. In regions of the feature space where two or more sites have a common data, their AI models are fused together. On the other hand, in the regions where there is only one applicable model that is used. A few exemplar regions where fusion is done are shown in the figure.

4.2.2 Bounds Expanding Data Exchange

It has been shown previously[2] that an approach of limited data exchange can help increase accuracy when data is skewed and not all classes and features are present across partner sites. A data exchange total of less than 2% may be enough to significantly increase the accuracy of a model trained using federated learning techniques. This process would work well for cases of missing classes among some partners. Their training data could be enhanced with new classes or features via a limited exchange of data with the partner(s) having the required

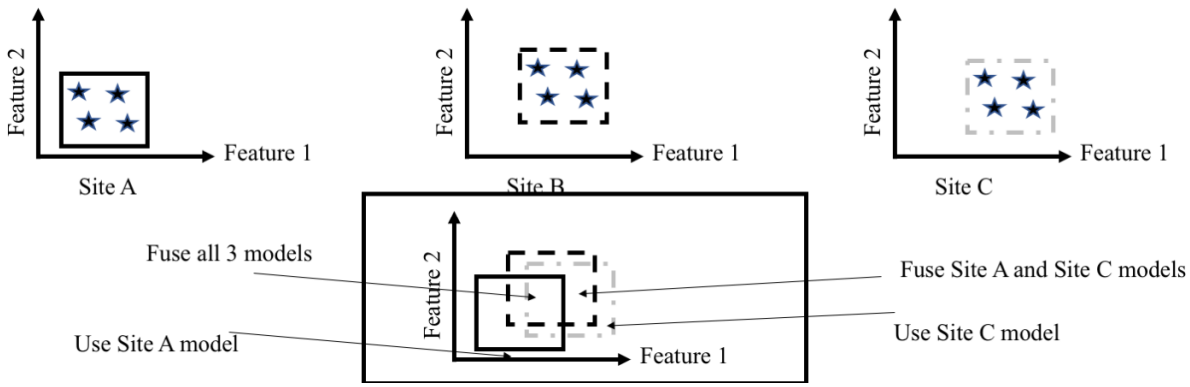

Figure 14. Bounds Aware Fusion for Featured Data Set – Example

classes. This method does rely on some form of data exchange being possible due to limited bandwidth and possible due to regulatory issues.

The primary motivation for the data exchange would be to ensure that all sites have data in the feature space that allows their model to be applicable in the same region. This would enable the fusion algorithm to work properly.

4.2.3 Data Reconstruction for Bounds Expanding Data Exchange

Another method for tackling the issues of data skew in tabular data would be for data reconstruction to take place. This would occur at all partner sites where there are missing features, classes or values in the training data. The intuition behind this is such that there is enough combined knowledge across partners in the federated learning environment that missing data at some partners can be reconstructed using the combined knowledge but without data exchange. There are several possible methods that make this possible as outlined below: estimation; generative networks and oversampling.

Estimation: in order to tackle the problem of missing classes, features or values, a simple method of estimation can be used to reconstruct data at each partner site. For example, for a missing value, the other values for that feature can be used within the context of the other available features to estimate what the missing value should be set to. This method would be simple to implement Within a single site and by extension logically simple in a federated scenario. However, in the federated scenario the missing values would need to be estimated prior to the beginning of training. This may require the fusion server to be able to communicate estimated values from all partners involved.

Estimating missing values, features or classes within a single site can be implemented by numerous approaches such as a simple average (mean/median) through to creating a model of a feature using the other features available within the tabular data. The model can be created via any learning mechanism but a simple approach such as a logistic regression is likely to be sufficient. The estimation process is complicated slightly when moving away from single site to the federated learning scenario (without data exchange) since several estimation techniques must now be merged in order to provide the required values. This can be treated in the same way as any other merge across sites and values between partners can be merged via numerous approaches fro the simple averaging approach to the more complex model-building approach.

Oversampling: for cases of data imbalance across partners, it may be possible to reconstruct training data at the site with less data using traditional oversampling techniques such as Synthetic Minority Oversampling (SMOTE)[8] and its subsequent improvements such as Borerline-SMOTE[9] or ADASYN.[10]

Using oversampling at a single site would involve implementing the traditional algorithms in the way they were originally intended. There may be some complexities introduced when moving these algorithms to the federated learning environment but these would be solved by simple engineering at the fusion server. The technique would not require data to be exchanged between partners since averages are taken across a set of samples for a given

set of features within the data set. Hence, two methods emerge for oversampling. One in which oversampling takes place independently at each site with the syntetic samples being swapped among partners. The other in which oversampling takes place across the federated learning environment using the fusion server to orchestrate the selecting of samples and swapping of averages over the selected data set.

GAN: the use of Generative Adversarial Networks (GANs)[11] has become more advanced and commonplace since their inception in 2010. In fact, this technique for generating data has become highly advanced in the field of imagery in particular with the ability for the network to "imagine" new versions of existing images[12] or even entirely new images.

GANs can be used as an alternative approach to oversampling in order to address the data imbalance issue. This would be made possible by their ability to create entirely new training samples at a site where there is relatively little training data. Using this method, the site that has a large amount of training data would use their data to train a GAN. This trained model would be passed to the partner(s) with few training samples such that they can run the model and produce new training samples at their local site. This method would not require any form of data exchange between sites and could reliably reproduce training samples to address the data imbalance problem.

The GAN technique could also be used to more specifically address the problems of missing classes, features or values. In this instance, instead of training the GAN to reproduce an entire training sample, it is trained simply to take the available information (features, classes or values) from the partner site that has these things missing and to output the required data. Similarly to the method where a GAN is trained and sent across the network to create new training samples, the GAN would be trained and sent across the network specifically for the purpose of addressing the missing features, classes or values problem.

Similarly to the use of oversampling, the use of GANs to generate data would need some implementation specifics in order to function correctly in the federated learning environment. The fusion server would be configured to assess which partners are presented with which particular data skew issues and prior to full federated training taking place, each site may be requested to build a GAN to a particular specification. The next step would be for the fusion server to swap the GANs between partners and instruct the partners to run the networks in order to generate training data. Once this step is complete, the federated learning process can continue and complete as normal using the a combination of both the original and the generated data across all partner sites.

4.3 Experimentation

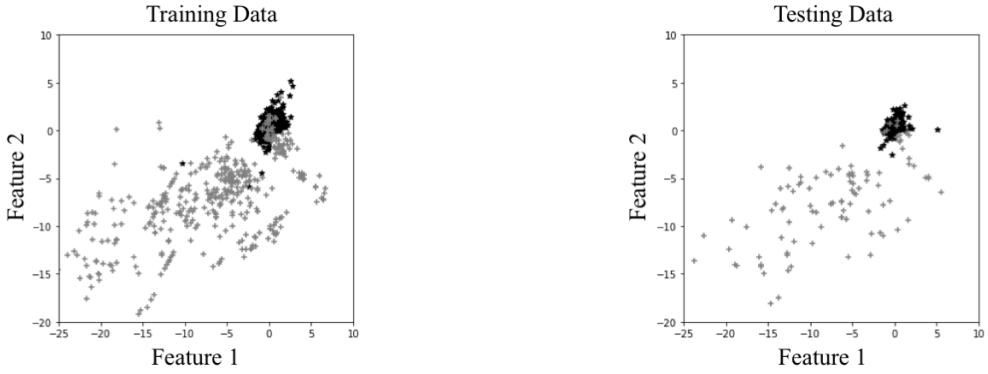

Figure 15. The training and testing datasets for Tabular Classification Experiments. The black marks show good credit card transactions, while the grey marks are fraudulent transactions.

In order to assess the effectiveness of the approaches discussed above, we ran some simple experiments on a typical example of tabular data, namely the analysis of credit card loan risk. Data on European Credit Card risk available from kaggle [13] which was contributed by ULB (Universit Libre de Bruxelles) [14] was used as the data set to create a classifier with two binary outputs. The dataset consists of 30 anonymized features which are

markers of credit card risk. Since our goal is to examine the impact of data skew, rather than tryig to find the best credit card risk algorithm, we trained a random forest on the dataset to identify the most significant features in the provided data set. Two features stand out as being the most important to predict the risk with relative importance of 0.54 and 0.10 respectively, while the third feature and below had the relative importance of only 0.04 or less. Therefore, we reduced the dataset to consist of only these two features. Reduction to two features also has the advantage that the dataset can be visualized more easily. Furthermore, to improve the fact that the dataset is originally unbalanced (fraud represented less than 0.5% of original data), we only selected a core set of non fraudulent credit card transactions which are roughly twice in number to that of fraudulent transactions, resulting in a synthetic data set with two features, a binary output indicator, and about 1500 points, with one third of it being fraudulent transactions.

The resulting data set was then broken into 80% set for training and the remaining 20% for testing. These two data sets are shown in Fig. 15. The black marks show normal transactions and the grey marks show fraudulent transactions. The training data set was then split in four sites corresponding to the four quadrants of the feature space in the training data set. The resulting data set at each of the four sites was as shown in Fig. 16. The data is highly skewed with three of the four sites consisting only of fraudulent transactions, while the fourth site has a better mix with about 800 normal transactions and 100 fraudulent transactions.

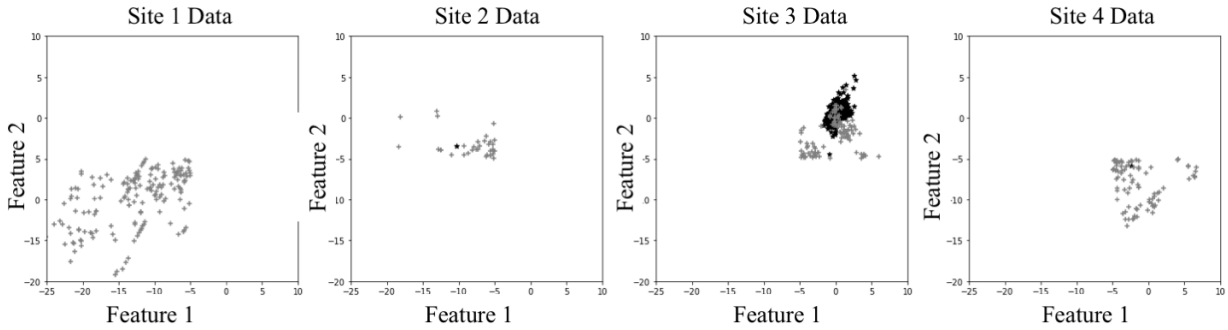

Figure 16. The split data set for four different sites. The black marks show good credit card transactions, while the grey marks are fraudulent transactions. The data is highly skewed.

The AI model used to discriminate between fraudulent and normal transactions for this data set was a 3-layered neural network, using 16, 4 and 1 neurons respectively. The first two layers were rectified linear units, while the last layer used a sigmoid activation function, indicating the probability that the transaction was not fraudulent. Each site trained its own model, as well as used the different approaches outlined above to create a fused model. 10 iterations over the process were used to get an better sense of the accuracy of the federated learning process.

We applied the same two approaches of Bound-Aware Fusion and Bound-Expanding data exchange to help build a better model. The results for this are shown in Table 1. The first few rows list the accuracy on training data if each site used only its local data. That result does not vary significantly across different runs of the system. The different approaches for using model fusion work differently. The accuracy of Naive fusion of the neural network which uses existing algorithms for fusing neural networks did not work too well in this case. The approach using the bounds of overlap (which were no overlap in this case), works much better than the data exchange or the naive approach. The data exchange method outperforms the naive approach of fusing neural networks. Furthermore, it has a lot less variability than the other fusion approaches.

5. GENERAL CLASSIFICATION PROBLEM

In the case of the general classification, the input for classification may be an input signal that does not come predefined into specific features. Examples of such inputs include images, sounds, vibration data, sensor data, and any other kind of complex input that may be available for analysis. The classification problem depends on

Table 1. Model Accuracy.

Approach	Accuracy Range	Mean Accuracy	Standard Deviation
Site 1 alone	37% - 37%	37%	0%
Site 2 alone	37% - 37%	37%	0%
Site 3 alone	63% - 63%	63%	0%
Site 4 alone	37% - 37%	37%	0%
Naive Fusion	63% - 97%	71%	11%
Data Exchange	63% - 94%	84%	13%
Bounds Aware Method	91% - 91%	91%	0%

the nature of the problem, e.g. sensor data from a phone may be used to classify if a person is walking, running, sitting or standing.

As in the case of tabular data, classification problems done during federated learning can degrade significantly if the classes available at different locations are partitioned. The challenges arise both due to data imbalance and due to the fact that the classes may be partitioned. With partitioned data, models may not remember the classes that are available and thus the federation process may fail. This is similar to the case of the data skew during the function estimation where one needs to validate that the context in which fusion is happening is consistent across different sites before one does the actual fusion operation.

A specific scenario which could be problematic for federated learning is the one where the training data is split across several different locations, so that each location has an independent data set. When such a situation is encountered, the accuracy of federated learning can degrade significantly .[2] The results from previous work [2] showing the impact of this degradation on a set of images with 10 classes, each class split among 10 different sites is shown in Figure 17.

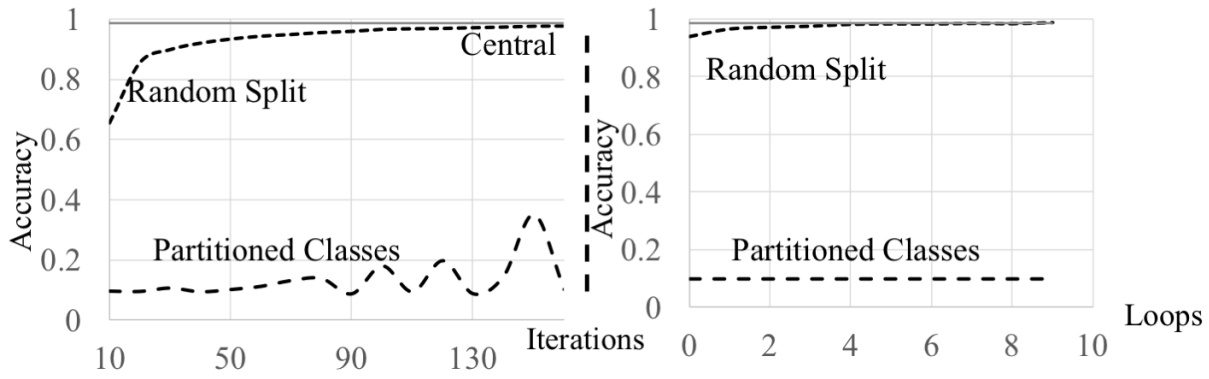

Figure 17. Problem in Classification with Partitioned data using two flavors of federated learning. In each flavor, the accuracy of federation without taking into account the differences in the classes present at different sites is very low.

Unlike the case of function estimation or tabular data, it is hard to define the context where the data available for classification can be determined as not being applicable, or being applicable. The determination of this would require the ability to have negative samples in the local training data set, or for a neural network to have the ability to state that it is encountering a data sample which it has never encountered before.

One way out of the problem is to enable the exchange of a small amount of data among different sites. This is the Bounds Expanding Data Exchange. Even with a small amount of exchange of data samples among different sites, a significant gain in accuracy can be obtained. For the example shown in Figure 17, each site has about

6,000 data points. The gain in accuracy that happens when only 16–64 data points are provided to other nodes by each node is shown in Figure 18. While this exchange reflects less than a percentage of the data points that are available, it improves the accuracy of the federated model significantly. This approach would work for those environments where the exchange of a small amount of data would be permitted.

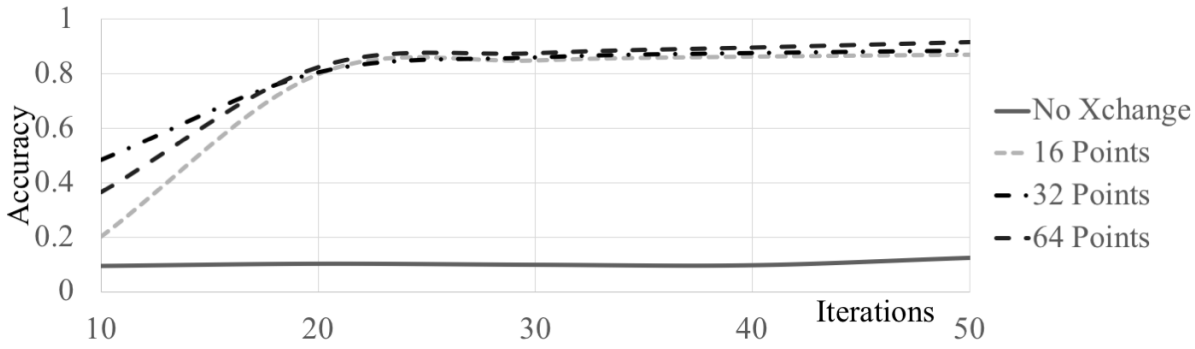

Figure 18. Solving the problem of data skew with a small amount of data exchange.

One of the open problems that remains for federated fusion is whether we can find a representation that will be able to characterize the boundaries of applicability of each of the models, like we were able to do in the case of function estimation or classification on tabular data. That would allow data skew issues to be addressed in cases where no data exchange is permitted. However, we are not aware of a good way to understand the bounds where a neural network is applicable.

6. CONCLUSIONS

In this paper, we have examined the challenges associated with performing federated learning in the situation where the data at each of the sites holding a part of the training data is highly skewed. We have examined three machine learning problems, function estimation, classification models built on tabular data, and classification models built on raw data such as images or audio. We have shown that an approach for bounds-aware fusion and bounds-expanding data expansion work as effective strategies for countering the problem of data skew.

While both approaches can be used for the problems of function estimation, and for classification on tabular data, we could only use the bounds-expanding data exchange for classification on raw data. In future work, we will explore effective ways to design and implement the approach for bounds-aware fusion on classification on raw data.

ACKNOWLEDGMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C., and Shi, W., "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics* **112**, 59–67 (2018).

[2] Verma, D., Julier, S., and Cirincione, G., "Federated ai for building ai solutions across multiple agencies," in [*AAAI FSS-18: Artificial Intelligence in Government and Public Sector, Arlington, VA, USA*], (2018).

[3] Verma, D., Chakraborty, S., Calo, S., Julier, S., and Pasteris, S., "An algorithm for model fusion for distributed learning," in [*Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX*], **10635**, 106350O, International Society for Optics and Photonics (2018).

[4] Zhu, G., Liu, D., Du, Y., You, C., Zhang, J., and Huang, K., "Towards an intelligent edge: Wireless communication meets machine learning," *arXiv preprint arXiv:1809.00343* (2018).

[5] Park, Y., "A mapping from linear tree classifiers to neural net classifiers," in [*Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*], **1**, 94–100, IEEE (1994).

[6] Setiono, R. and Leow, W. K., "On mapping decision trees and neural networks," *Knowledge-Based Systems* **12**(3), 95–99 (1999).

[7] Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., and Chan, K., "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," *arXiv preprint arXiv:1804.05271* (2018).

[8] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P., "Smote: synthetic minority oversampling technique," *Journal of artificial intelligence research* **16**, 321–357 (2002).

[9] Han, H., Wang, W.-Y., and Mao, B.-H., "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in [*International conference on intelligent computing*], 878–887, Springer (2005).

[10] He, H., Bai, Y., Garcia, E. A., and Li, S., "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in [*2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*], 1322–1328, IEEE (2008).

[11] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets," in [*Advances in Neural Information Processing Systems 27*], Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., eds., 2672–2680, Curran Associates, Inc. (2014).

[12] Bau, D., Zhu, J.-Y., Strobelt, H., Bolei, Z., Tenenbaum, J. B., Freeman, W. T., and Torralba, A., "Gan dissection: Visualizing and understanding generative adversarial networks," in [*Proceedings of the International Conference on Learning Representations (ICLR)*], (2019).

[13] "Kaggle dataset on credit card fraud." https://www.kaggle.com/mlg-ulb/creditcardfraud. Accessed: 2019-02-28.

[14] Dal Pozzolo, A., Caelen, O., Johnson, R. A., and Bontempi, G., "Calibrating probability with undersampling for unbalanced classification," in [*2015 IEEE Symposium Series on Computational Intelligence*], 159–166, IEEE (2015).