This paper appears in ACM ASIA Conference on Computer and Communications Security (ACM **ASIACCS**) 2020.
Please feel free to contact us for questions or remarks. AsiaCCS'20, October 05–09, 2020, Taipei, Taiwan

Responsive Web User Interface to Recover Training Data from User Gradients in Federated Learning

Hans Albert Lianto*
Nanyang Technological University
Singapore
HANS0032@e.ntu.edu.sg

Yang Zhao*†
Nanyang Technological University
Singapore
S180049@e.ntu.edu.sg

Jun Zhao
Nanyang Technological University
Singapore
junzhao@ntu.edu.sg

arXiv:2006.04695v1 [cs.CR] 8 Jun 2020

ABSTRACT

Local differential privacy (LDP) is an emerging privacy standard to protect individual user data. One scenario where LDP can be applied is federated learning, where each user sends in his/her user gradients to an aggregator who uses these gradients to perform stochastic gradient descent. In a case where the aggregator is untrusted and LDP is not applied to each user gradient, the aggregator can recover sensitive user data from these gradients. In this paper, we present a new interactive web demo showcasing the power of local differential privacy by visualizing federated learning with local differential privacy. Moreover, the live demo shows how LDP can prevent untrusted aggregators from recovering sensitive training data. A measure called the *exp-hamming recovery* is also created to show the extent of how much data the aggregator can recover.

CCS CONCEPTS

• **Security and privacy** → **Distributed systems security**; • **Computing methodologies** → *Supervised learning by classification*; *Supervised learning by regression.*

KEYWORDS

Local differential privacy, Machine learning, Linear regression, Federated learning, Stochastic gradient decent.

ACM Reference Format:
Hans Albert Lianto, Yang Zhao, and Jun Zhao. 2020. Responsive Web User Interface to Recover Training Data from User Gradients in Federated Learning. In *AsiaCCS'20: The 15th ACM ASIA Conference on Computer and Communications Security, October 05–09, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

1.1 Local differential privacy

Local differential privacy is an emerging model based on differential privacy where each record or tuple in a database, even when exposed to adversaries (which could include the data curator or aggregator), does not give the selected user's sensitive information to them. Hence even the data curator or aggregator is not trusted with true information from each entry in the dataset. Local differential privacy was first implemented by Google in its RAPPOR technology in their paper in [4].

Local differential privacy (ϵ-LDP) [4] is defined as follows:

Definition 1. *A random function f satisfies ϵ-local differential privacy if and only if, for any two tuples t and $t^{'}$ in the domain of f, and for any output t^* of f:*

$$Pr[f(t) = t^*] \leq e^{\epsilon} \times Pr[f(t^{'}) = t^*]$$

where parameter ϵ controls the privacy-utility trade-off (the higher the ϵ, the higher the tuple utility and the lower the tuple privacy).

Many companies store enterprise data in relational form with RDBMS software. For example, Airbnb and Uber store their data using mySQL [1]; Netflix and Instagram store their data using PostgreSQL [2]. Data stored in this relational form come in a set of tuples or records with the same schema. Each tuple or record have the same number of 'dimensions' or rows. LDP-perturbed data can hence also be stored in these relational forms.

Methods to perturb this multidimensional data to satisfy ϵ-LDP have been proposed, including Duchi *et al.*'s recent proposal in [3] and Wang's improved proposal (the Piecewise Mechanism and Hybrid Mechanism) in [5]. These methods will be used in the web demo outlined in subsequent sections of the paper.

1.2 Local differential privacy applied in federated learning

One practical application for local differential privacy is in machine learning algorithms, particularly in federated or centralized distributed learning. As illustrated in Fig. 1, each user is considered a node that trains gradients independently; each node then sends these gradients to the parameter server for aggregation [6]. The data that these users make use of to obtain user gradients is sensitive information and should not be leaked.

It can be shown in the subsequent web demo that without local differential privacy, a significant portion of the training data could be leaked by the untrusted aggregator in any federated learning setting where each user submits their gradients for training.

2 DEMO OVERVIEW

A web user interface (called ldp-machine-learning) is created to simulate stochastic gradient descent in federated learning, and for the aggregator to recover training data from each user. A screenshot

*Both authors contributed equally to the paper. The order of names is alphabetical.
†Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
AsiaCCS'20, October 05–09, 2020, Taipei, Taiwan
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Figure 1: Federated learning flowchart

of the demo is shown in Fig. 2 and the demo is publicly available in https://ldp-machine-learning.herokuapp.com/.

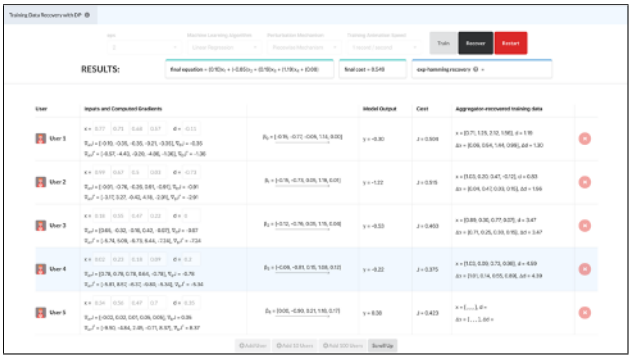

Figure 2: Screenshot of ldp-machine-learning demo

In the demo, when all users fill in the training data and the Train button is started, one epoch of stochastic gradient descent takes place. Unbeknownst to the 'users' illustrated in the GUI, the aggregator uses the gradients from the user and the current parameters of the model to backtrack and recover the training user data, which poses a security risk. Features of the UI are outlined in the next section.

2.1 Animation and animation speed

The main animation in the demo occurs when the Train and Recover buttons are clicked. When the Train button is clicked, training proceeds and the current cost or training accuracy of the machine learning model is shown beside each user sending his or her gradients to the aggregator. When training is finished, the final model cost and accuracy are outputted on screen. When the Recover button is clicked, the untrusted aggregator attempts to recover sensitive user training data from the gradients sent by each user. At the end of recovery, the average *exp-hamming recovery* for a user is outputted on screen. Normally, the training and recovery animation occurs at a rate of one user per second; however, changing the Training Animation Speed input from "1 record / second" to "Instant" in the UI input form will remove the animation and immediately display training and recovery results when training finishes.

2.2 Machine learning algorithms and LDP perturbation mechanisms

The demo features federated learning with various machine learning algorithms such as linear regression, logistic regression and support vector machine, which can be toggled, as shown in Fig. 3.

Moreover, as shown in Fig. 3, the LDP perturbation mechanism to perturb user gradients can be toggled from 4 options: Laplace mechanism, Duchi *et al.* mechanism, Piecewise mechanism and Hybrid mechanism; the privacy budget ϵ of the LDP algorithm can also be toggled.

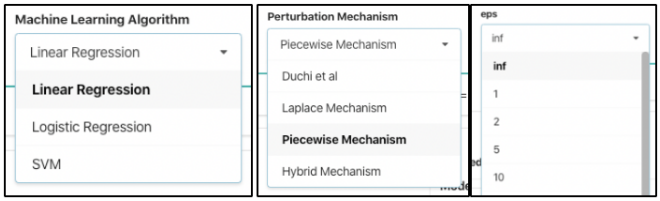

Figure 3: Toggling between different ML algorithms, LDP algorithms and privacy budgets

2.3 Other features and specifications

An 'Add 10 Users' button and an 'Add 100 Users' button were added to automatically generate new training data. A scroll up button from the bottom page is added for easier navigation. The buttons are shown in Fig. 4.

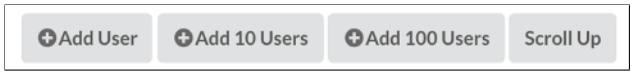

Figure 4: The add users and scroll up buttons

Each user's training data is generated via the following equation:

$$d = -0.55x_1 - 0.82x_2 + 0.07x_3 + 0.95x_4 + 0.31$$

for linear regression. For logistic regression and SVM it is:

$$d = \begin{cases} 1, if\ -0.55x_1 - 0.82x_2 + 0.07x_3 + 0.95x_4 + 0.31 > 0 \\ -1, otherwise \end{cases}$$

From these two equations, the ideal weights for the model are $[-0.55, -0.82, 0.07, 0.95, 0.31]$. The closer the model weights are to the ideal weights, the better the model is. Initial model weights and training data are generated with a pseudorandom seed via an extension in the JavaScript Math library.

2.4 Exp-hamming recovery

Before the formal definition of the *exp-hamming recovery* is discussed, it is imperative to identify what it means when it is "more difficult" for an aggregator to recover user training data. Consider a user's training data to be the vector $\vec{x} = (x_1, x_2, ..., x_n)$ with n dimensions. Now consider an aggregator's recovered training data to be the vector $\vec{x_r} = (x_{1r}, x_{2r}, \cdot, x_{nr})$. The natural convention is that it is more difficult to recover $\vec{x}$ if $\vec{x_r}$ is farther from $\vec{x}$. Hence a distance metric is needed; naturally, the higher the value of this metric is, the more difficult it is for an aggregator to recover the user's training data. Manhattan distance ($\ell 1$-norm) is the distance metric used for the definition of the *exp-hamming recovery*.

The *exp-hamming recovery* is designed so that the more (respectively less) difficult it is for the aggregator for recover training data, the lower (respectively higher) the *exp-hamming recovery* should be. The *exp-hamming recovery* is hence formally defined as follows:

$$E = exp(-k(\|\vec{x} - \vec{x_r}\|_1)$$

where k is a customizable constant (its value is important for accurate interpretation), and $\|\cdot\|_1$ represents the Manhattan distance.

This metric naturally makes sense because, if $\vec{x} = \vec{x_r}$, there is full recovery of user training data, meaning that $E = 1$. If $\|\vec{x} - \vec{x_r}\|_1 = \infty$, meaning that $\vec{x}$ and $\vec{x_r}$ are infinitely apart, there is no information gained by the aggregator of what the user training data is like, meaning that $E = 0$. One can choose a value of k such that $E < 1/e \approx 0.368$ should the Manhattan distance $\|\vec{x} - \vec{x_r}\|_1 > 1/k$. This critical value 0.368 for *exp-hamming recovery* would hence be a good heuristic to whether the aggregator has enough data to be able to make a good guess at the user's real training data or not. In the demo, the value of k used is 0.5.

2.5 Privacy budget allocation and training specifications

For privacy budget management in each user, the privacy budget is allocated equally to each gradient value. Since there are 5 total values to perturb in the demo, if the privacy budget allocated to each user is ϵ each of these values are perturbed with privacy budget $\epsilon/5$ For training, for all the machine learning algorithms, stochastic gradient descent proceeds at a constant learning rate of $\alpha = 0.01$. A feature to customize this learning rate may be added to a later version of the demo, but the customization of this hyperparameter is not necessary at this stage.

3 IMPLEMENTATION

Fig. 5 shows the architecture diagram for the web UI, which shows that the site's UI and main logic operate entirely on the front-end. This is to eliminate the latency from API calls to a backend server, which would slow down the site performance. In addition to that, the UML State Diagram for the UI is shown in Fig. 6.

The development library used for the frontend is React, an open-source JavaScript library. Throughout development, a conscious effort is made to separate the part of the application corresponding to its appearance and to its logic. The appearance is maintained within the project's 'components/' directory, while the site logic lies mainly in the 'utils/' directory. The version control system for the project is git and project code is stored on GitHub. Moreover, the demo is deployed and served on Heroku, a cloud platform-as-a-service.

4 CONCLUSIONS

In this paper, a web GUI is made to illustrate the above results and the power of locally differentially private mechanisms to perturb data. The demo is publicly accessible so potential researchers in the field of differential privacy are able to understand local differential privacy in the context of applying it in machine learning algorithms.

The GUI is easily extensible to other machine learning algorithms and LDP perturbation mechanisms in future work. Moreover, training hyperparameters such as batch size and learning rates can be added to the demo for better training results that nearly match experimental setups with real datasets.

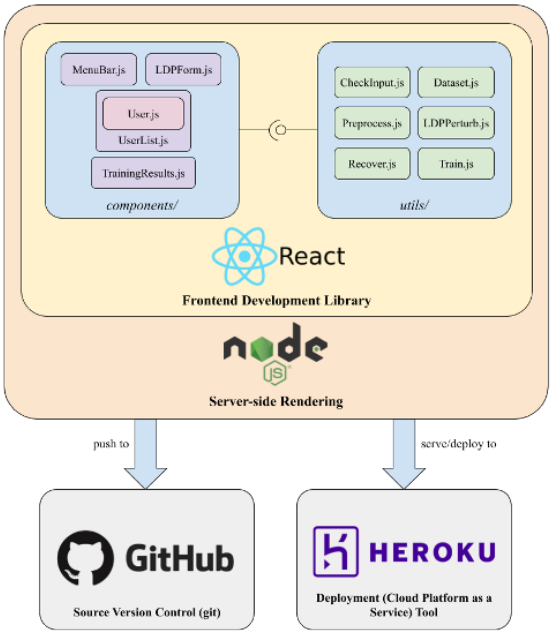

Figure 5: Software architecture diagram for ldp-machine-learning

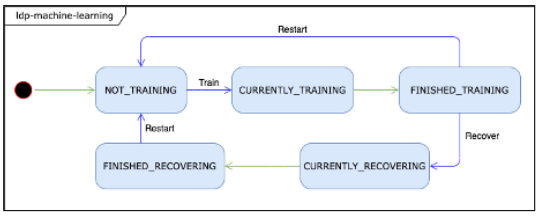

Figure 6: UML State Machine Diagram from ldp-machine-learning

REFERENCES

[1] [n.d.]. Why developers like MySQL. https://stackshare.io/mysql https://stackshare.io/mysql.
[2] [n.d.]. Why developers like PostgreSQL. https://stackshare.io/postgresql https://stackshare.io/postgresql.
[3] John C Duchi, Michael I Jordan, and Martin J Wainwright. 2018. Minimax optimal procedures for locally private estimation. *J. Amer. Statist. Assoc.* 113, 521 (2018), 182–201.
[4] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 1054–1067.
[5] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and analyzing multidimensional data with local differential privacy. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 638–649.
[6] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*. 14747–14756.