Technical Whitepaper

Differential Privacy in LINE Federated Learning

v1.0, September 2023

LINE Corporation

Copyright

This document is provided for informational purposes only. LINE Corp. has endeavored to verify the completeness and accuracy of information contained in this document, but it does not take the responsibility for possible errors or omissions in this document. Therefore, the responsibility for the usage of this document or the results of the usage falls entirely upon the user, and LINE Corp. does not make any explicit or implicit guarantee regarding this.

Software products or merchandises mentioned in this document, including relevant URL information, conform to the copyright laws of their respective owners. The user is solely responsible for any results occurred by not complying with applicable laws.

LINE Corp. may modify the details of this document without prior notice.

Contents

Introduction

Federated Learning (FL)[18] is a collaborative machine learning method in which clients never share the raw data but send the model update information (e.g. gradients or model difference) to the server for the update of the global model. The global model is kept up-to-date with a large number of the gradients from the participants of FL. Therefore, new users can tap into such a globally up-to-date model whenever they join LINE.

FL preserves the privacy of users with the federated protocol where clients never share the raw data, but only share the model update information with the server. Exposing only the model update information like gradients would seem sufficient to protect the privacy of individual users. However, several studies have shown that the raw data can be reproduced from the gradients[15,24]. As we place the highest priority on the privacy of individual users, we have adopted differential privacy as a measure of rigorous privacy guarantees.[9,11].

Differential privacy (DP)[9,11] is the golden standard privacy notion used in various products, services and statistical surveys[2,8,13]. DP is achieved by injecting a particular noise. The noise achieving DP makes any outputs computed from user data indistinguishable. The indistinguishability is quantified by privacy parameters like ϵ and δ.

Local differential privacy[7,19] has been a widely accepted privacy standard that makes it hard to distinguish randomized responses crafted from any inputs to the extent quatified by privacy parameters ϵ and δ.

One way to implement FL under differential privacy is to introduce a privacy mechanism for each client to preserve their own privacy, and then each client sends the noise-injected model update information to the server[18,22]. Since the collected model update information is indistinguishable among a batch of user reports, the server cannot infer individual properties, but estimates the global characteristics from uploaded user reports to update the global model.

Find the following example of FL implementation on the LINE app for sticker recommendation. Through the FL processes, LINE reinforces privacy and, at the same time, improves the usability of machine learning features like sticker recommendation as the ML model lists up sticker candidates without explicitly receiving the local "activity log" such as "Sent Sticker Data". While preserving the privacy of users via differential privacy mechanism, LINE performs federated learning to improve the efficacy of the sticker recommendation. See the details on the LINE's Security and Privacy [a].

This white paper offers in-depth details on differential privacy in LINE's federated learning platform. The target audience of this white paper is the engineers and developers who focus on machine learning or security, presumably with a strong understanding of statistics, data engineering and machine learning.

[a] https://linecorp.com/en/security/article/459

LINE Federated Learning Platform

LINE introduced a federated learning platform to improve the performance of recommendations. The first application of the federated learning platform is "sticker recommendation". This section gives an overview of the FL process between LINE apps and LINE's server.

Overview

Under the federated learning framework, machine learning features run on the clients, using preference information, as personalized local machine learning models are stored on the clients. These features offer users prompt personalization. Sticker recommendation is the first use-case of this federated learning technology (Refer to the LINE's Security & Privacy [a] for more information on sticker suggestion).

The federated learning technology reduces the information LINE receives from users by processing their action histories like "Sent Sticker Data" on their device, and keeps raw user data on their devices, which reinforces user privacy preservation.

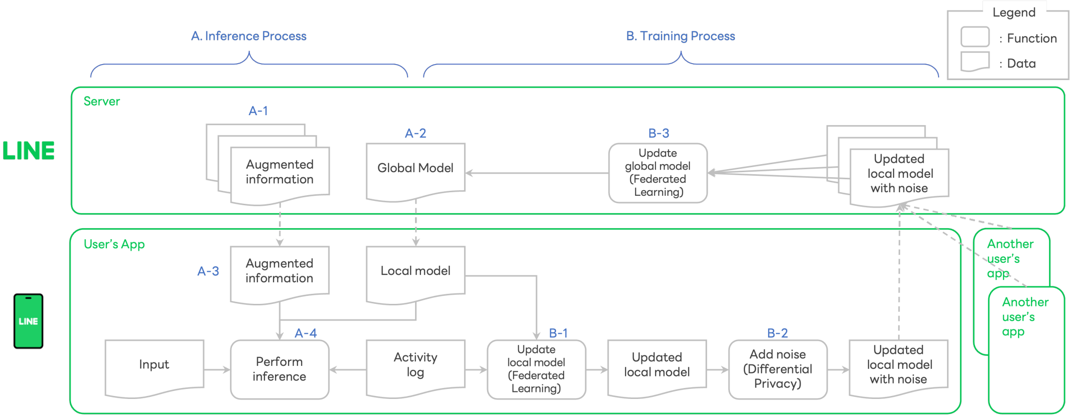

Figure 2.1: Overview of LINE Federated Learning for Sticker recommendation. See the details on the LINE's Security & Privacy [a].

The federated learning process is comprised of two sub-processes: "A. Inference" and "B. Training". Figure 2.1 shows the federated learning process for sticker recommendation.

Inference Process

A-1. LINE Server generates "augmented information" for the target machine learning task.

A-2. LINE Server also generates a "global model".

A-3. Each LINE app on the user device downloads the "augmented information" and "global model" from the server.

A-4. LINE apps extract the local user "activity log" along with users actions. LINE apps use the "activity log" and the "local model" to infer the preferable items from the set of candidate items. This step takes place on the user device.

Training Process

B-1. Randomly selected LINE apps perform model training on the user devices, using the local "activity log", and update the local model on their devices. LINE apps remove the "activity log" after the "activity log" is used for local model training. That means any "activity log" is used only once.

B-2. As an additional post process, LINE apps add the noise to the model, using the "Differential Privacy" technology. When sending the updated local model, LINE apps also drop the user identifier to send the minimum amount of information. This privacy-preserving process aims to make others difficult to estimate the actual activities by the user via the updated local model.

B-3. The server receives updated local models from multiple users, randomly selected in B-1. The models are averaged and used to update the global model. The collaborative machine learning method that involves both client devices and server(s) is called "Federated Learning".

Algorithm Overview

This section introduces an overview and mathmatical notations of the federated learning algorithm with differential privacy guarantees.

Client-side Algorithm

1. Let θ_i be the client i's local model parameters. Copy the downloaded global model parameters θ_{global} to the local parameters as $\theta_i \leftarrow \theta_{global}$.
2. Train the local model. Let θ'_i be the trained model parameter from θ_i.
3. Compute the model difference Δ_i as $\Delta_i \leftarrow \theta'_i - \theta_i$.
4. Randomize the model difference as $\Delta^*_i \leftarrow \mathcal{R}(\Delta_i)$, using the local randomizer $\mathcal{R}$ to satisfy (ϵ_0, δ_0)-LDP (See the details under the section on "Client-side Local Randomizer").
5. Report the randomized model difference Δ^*_i to the server.

Server-side Algorithm

When the server receives a batch of randomized model differences, the server computes the average over the batch, having m randomized model differences.

$$\bar{\Delta}^* = \frac{1}{m} \sum_{i \in [m]} \Delta^*_i \tag{3.1}$$

Then, the server updates the global model as follows:

$$\theta_{global} \leftarrow \theta_{global} + \bar{\Delta}^* \tag{3.2}$$

The server repeats this process for each batch.

Privacy Model

We attempt to preserve the privacy of local data for each user. Especially, we employ a privacy enhancing mechanism to keep each model update information (i.e. model difference) sent from each client differentially private in the federated learning process above.

This section introduces local differential privacy, and describes the privacy model we used, namely event-level local differential privacy, and its privacy composition.

Local Differential Privacy

Local differential privacy[7,19] has been a widely accepted privacy standard that makes it hard to distinguish randomized responses crafted from any inputs to the extent quantified by privacy parameters ϵ and δ.

Definition 1. (ϵ, δ)-Local Differential Privacy. Given privacy parameters $\epsilon \in \mathbb{R}_0^+$ and $\delta \in [0,1]$, a randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{S}$ satisfies (ϵ, δ)-local differential privacy if, for any pair of inputs $X, X' \in \mathcal{X}$ and any subset of outputs $S \subseteq \mathcal{S}$, it holds that

$$\Pr[\mathcal{M}(X) \in S] \leq \exp(\epsilon) \Pr[\mathcal{M}(X') \in S] + \delta \tag{4.1}$$

To utilize DP properly, we need to consider a privacy model including client-side data management, privacy budget accounting, and design of local randomizer.

Event-level Local Differential Privacy

LINE apps drop user identifiers when reporting the randomized model update information for "de-identification". That means the server cannot link multiple reports from a single client since the explicit identifier of the user is removed. In addition, the local data (i.e. "activity log") is used only once for training and immediately discarded. Such "disjoint data usage" makes capturing the history of any user's activities untraceable and enables parallel privacy composition[23] over the sequential observations from a single client.

Each client ensures (ϵ_o, δ_0)-LDP for a single output (i.e. model difference) on each client. Assume a reporting period as an event, a client gives *event-level local differential privacy* for each output by local randomizer. Event-level differential privacy is defined in[10]. Event-level LDP is an extension of it in consideration of local privacy.

Another paradigm of preserving the privacy of client's outputs is user-level LDP which ensures LDP over all the client's outputs. However, we do not have practical solutions for satisfying the user-level LDP for the unlimited size of the sequences. Therefore, we justify the use of the event-level LDP with the privacy-enhancing features defined as "de-identification" and "disjoint data usage".

We also introduce a random sampling of participants for a single round of federated learning. This sampling strategy makes it hard to track activities of a specific user.

Privacy Composition

As this federated learning platform employs "de-identification" and "disjoint data usage", we consider any randomized submissions from clients are independent. Thus, we account for the privacy consumptions per event.

As described in the federated learning process above, we assume that a set of training samples is removed from the client whenever the set is used to generate a model difference. That means a client never utilizes any training samples twice. Between any two reporting periods, the set of training samples do not have any redundant data. At each reporting period, the client produces a randomized model difference from a single set of training samples that the client has never used.

Due to the assumption above, we compute the privacy composition of a single client over the all reporting periods by the parallel composition theorem[23]. That is, the total privacy consumption for any set of training samples is always (ϵ_0, δ_0).

Furthermore, under this assumption, how many times each client submit randomized model difference does not affect the privacy consumptions. However, we introduce a parameter that limits the number of submissions per client to keep balance (i.e. fair) among observations from the clients.

Client-side Local Randomizer

To ensure the LDP guarantee discussed above, we employ a local randomizer $\mathcal{R}$ that injects the Gaussian noise into a raw model update information.

We here introduce how to design the Gaussian noise to satisfy (ϵ_0, δ_0)-LDP. We also utlize secure random sampling to robustly generate the Gaussian noise.

Overview

We employ the Gaussian analytical mechanism[3], which is an exntension of the Gaussian mechanism, as the noise injecting method. The step of the local randomizer $\mathcal{R}$ is described as follows:

1. Sample a Gaussian noise z as $z \sim \mathcal{N}(0, (2C\sigma)^2\mathbb{I}_d)$.
2. Add the sampled Gaussian noise z to the clipped model difference as $\Delta_i^* \leftarrow \pi_C(\Delta_i) + z$.

where σ is the noise scaler, C is the clipping threshold, $\pi_C(\cdot)$ is the clipping operator, and d is the dimension of the model difference Δ_i. The noise scale $2C\sigma$ is required to satisfy (ϵ_0, δ_0)-LDP. How to set the noise scaler σ also follows the Gaussian analytical mechanism[3]. The detailed designs of the noise scale and the clipping operator are described in later part.

Norm Clipping

The norm clipping[1] bounds the ℓ_2-norm of the gradient at a pre-defined constant value to bound the sensitivity at the constant. The norm clipping is defined as follows:

$$\pi_C(\Delta_i) = \Delta_i \cdot \min(1, \frac{C}{||\Delta_i||_2}) \tag{5.1}$$

where C is the clipping threshold.

Gaussian Mechanism for LDP

To apply a mechanism designed for central DP (e.g. Gaussian mechanism) to satisfy local DP, we need to modify the noise scale of the mechanism. Local DP is known as a bounded DP, while central DP is a unbounded DP[20]. Between bounded DP and unbounded DP, the sensitivity under bounded DP is at most twice the sensitivity under unbounded DP. Therefore, to satisfy (ϵ_0, δ_0)-LDP, we have to double the noise scale when we use the mechanism designed for central DP. Thus, in the Gaussian mechanism, the required noise scale is $2C\sigma$, where C is the clipping threhold and σ is the noise scaler defined by (ϵ_0, δ_0).

Secure Random Sampling

A naive implementation of sampling the Gaussian noise is known to be vulnerable against statistical attacks. The vulnerability is caused by the approximation of real values to floating point numbers. To avoid the issue, we employ the secure random sampling[16]. In this solution, we need to run the sufficient number of samples to compute the average for the sampled Gaussian noise. We sample the Gaussian noise six times for this purpose.

Further Extension

Note the description here is not included in the first release of LINE's federated learning platform. We are now studying the feasibility of the following extensions to reinforce the usability of federated learning under rigorous privacy guarantees.

One of the most important extensions will be introducing a trusted shuffler[4,6,12,14,21] to amplify local privacy through anonymizing the identity of clients. To introduce the shuffle model, we also need to consider how to securely implement the shuffler. Our current idea is to use TEEs (Trusted Execution Environments). This will improve the efficacy of federated learning while securely preserving the privacy of users under differential privacy with such trusted entities.

Secure aggregation[5,17] with secure computation is also another option for us. The secure aggregation will increase the efficacy under the securely implemented private federated learning with a combination of secure multi-party computation, homomorphic enryption, TEE, and differential privacy.

Conclusion

This white paper describes how federated learning on LINE Apps preserves the privacy of users with the differential privacy mechanisms. We always strive to seek better implementation and hyper-parameters that achieve higher efficacy as well as preserving sufficient privacy.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] J. M. Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2867–2867, 2018.

[3] B. Balle and Y.-X. Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

[4] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proceedings of the 26th symposium on operating systems principles*, pages 441–459, 2017.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[6] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev. Distributed differential privacy via shuffling. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*, pages 375–403. Springer, 2019.

[7] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Minimax optimal procedures for locally private estimation. *Journal of the American Statistical Association*, 113(521):182–201, 2018.

[8] C. Dwork. Differential privacy and the us census. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems*, pages 1–1, 2019.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

[10] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 715–724, 2010.

[11] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[12] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, S. Song, K. Talwar, and A. Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *arXiv preprint arXiv:2001.03618*, 2020.

[13] Ú. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[14] V. Feldman, A. McMillan, and K. Talwar. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 954–964. IEEE, 2022.

[15] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[16] N. Holohan and S. Braghin. Secure random sampling in differential privacy. In *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part II 26*, pages 523–542. Springer, 2021.

[17] D. Huba, J. Nguyen, K. Malik, R. Zhu, M. Rabbat, A. Yousefpour, C.-J. Wu, H. Zhan, P. Ustinov, H. Srinivas, et al. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems*, 4:814–832, 2022.

[18] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[19] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[20] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204, 2011.

[21] S. P. Liew, S. Hasegawa, and T. Takahashi. Shuffled check-in: privacy amplification towards practical distributed learning. *arXiv preprint arXiv:2206.03151*, 2022.

[22] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[23] F. 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*, pages 19–30, 2009.

[24] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and P. Molchanov. See through gradients: Image batch recovery via gradinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16337–16346, 2021.