Label Leakage and Protection in Two-party Split Learning

Oscar Li[1,*], **Jiankai Sun**[2], **Weihao Gao**[2], **Hongyi Zhang**[2]
Xin Yang[3], **Junyuan Xie**[2], **Chong Wang**[2]
[1] Carnegie Mellon University, [2] ByteDance Inc., [3] University of Washington
oscarli@cmu.edu
{jiankai.sun, weihao.gao, hongyi.zhang}@bytedance.com
yx1992@cs.washington.edu, {junyuan.xie, chong.wang}@bytedance.com

Abstract

In this paper, we make the vertical federated learning more secure, by preventing the leakage of the participants' ground-truth labels in a two-party split learning setting. Particularly, we identify a simple but accurate label-uncovering method which uses the norm of the communicated gradients between the parties. Moreover, we propose a theoretically justified protection technique which strategically perturbs the gradient randomly before communication and show that the technique can protect against a large class of label-uncovering methods including our proposed norm-based method. We then experimentally demonstrate the effectiveness and competitiveness of the proposed protection method compared to other baselines.

1 Introduction

With the increasing concerns on data security and user privacy in machine learning, *federated learning* (FL) [15] becomes a promising solution to privacy and security challenges. Based on how sensitive data are distributed among various parties, FL can be classified into different categories [28], notable among which are *vertical (cross-silo) FL* and *horizontal (cross-device) FL*. In contrast to horizontal FL where the data are partitioned by examples, vertical FL assumes that the data are partitioned by different features (including labels). A typical example of vertical FL is when an online media platform A displays advertisements of an ecommerce company B to its users and charges B for each *conversion* (e.g., user clicking the ad and buying the product). In this case, both parties have different features for the same set of users: A has features on users' media viewing records and B has the users' product browsing records and conversion labels. The conversion labels are not available to A because users' buying processes happen entirely on B's website/mobile app.

If both parties want to train a deep learning model jointly to predict the conversion rate based on feature-partitioned data, they could sign legal agreements to allow them share the data with each other. However, due to privacy and business concerns, recently companies and other entities are unwilling to share their data with each other or other third parties. *Split learning* [13, 26] enables the joint training of deep learning models without sharing sensitive data by splitting the execution of a deep learning model between the parties on a layer-wise basis. In vanilla split learning, the party without labels (called the *passive party*) sends the computation results of the intermediate layer (called the *cut layer*) rather than the raw data to the party with labels (called *active party*). The active party then completes the rest of the forward step computation, computes the gradients based on the labels, and sends the gradient with respect to the cut layer back to the passive party. The passive party then completes the back propagation with the gradients of the cut layer using chain rule.

[*] work done as an intern at ByteDance Inc.

Appearing at NeurIPS 2020 Workshop on Scalability, Privacy, and Security in Federated Learning (SpicyFL), Vancouver, Canada.

At first glance, the process of split learning seems private because no raw feature or label is communicated between the two parties. However, recently Zhu et al. (2019) demonstrated that in **horizontal** FL setting, the central server could recover the raw features and labels of a device using the model parameters and the gradient shared by that device. Inspired by this, we wonder whether the raw data (specifically the labels from the active party) can also be leaked by sharing gradients of the split layer in the **split learning** (vertical FL) setting.

In this work, we identify a simple but accurate method that uses the norm of the communicated gradients from the active party to uncover the labels. Moreover, we propose a theoretically justified protection technique which strategically perturbs the gradient randomly before communication and can protect a wide class of label-uncovering methods including the proposed norm-based method. We then experimentally demonstrate the effectiveness and competitiveness of the proposed protection method compared to other baselines.

1.1 Related work

Uncovering the raw data. Even though raw data are not shared across different parties in federated learning, they are still not secure when gradients and model parameters are shared among different parties. In the horizontal FL setting, Zhu et al. (2019) showed that an honest-but-curious server can uncover the raw features and labels of a participating device by knowing the model architecture, parameters, and the communicated gradient of the loss on the device's data. Building upon the techniques in [30], Zhao et al. (2020) showed that the ground truth label of an example can be extracted by exploiting the relationship between the directions of the gradients of the weights connected to the logits of different classes. In this work, we study a different setting in FL — the two-party split learning setting for vertical FL, where no parties have access to the model architecture or model parameters on the other party's side. Our proposed method recovers the hidden label (not the raw input features [27]) and uses the magnitude (2-norm) instead of the direction of the gradients.

Privacy protection methods Because sharing intermediate computation results (model parameters, representations, gradients) can leak the raw data, FL communications still need to proceed in a privacy-preserving manner. There are generally three classes of approaches to communicate in a privacy-preserving manner: **1)** cryptography-based methods such as *Secure Multi-party Computation* [2, 5, 8, 18] and *homomorphic encryption*[3, 22]; **2)** system-based methods such as *Trusted Execution Environments* [23, 25]; **3)** perturbation methods such as randomly perturbing the communicated message [1, 16], shuffling the messages [7, 11], reducing message's data-precision, compressing and sparsifying the message [30], etc. Our work belongs to the third category where we add random perturbations to the gradients before communication to protect the labels. Many of the randomness-based privacy protection methods was proposed in the domain of horizontal FL where the aim is to protect the privacy of each example / each device's dataset [1, 4, 12, 16, 17, 19, 21, 24]. In this case, *Differential privacy* (DP) [9, 10] was used to quantitatively measure the proposed random mechanisms' ability to protect the privacy of any **individual** records. In this paper, we propose, to the best of our knowledge [14], the first randomness-based technique to protect the active party's label privacy during split learning in vertical FL. Unlike the record-level privacy considered in DP, we consider protecting the **entire** training set's label information as a whole – our approach reduces the percentage of the training set whose label information is leaked. We recognize that protecting each individual example's label information is also important and we leave it as future work.

2 Label leakage in split learning

In this section we first formally describe the gradient-based two-party split learning problem for binary classification. A careful inspection of the distribution of the communicated gradients in this setting allows us to identify a very effective method to uncover the active party's labels.

2.1 Split learning on binary classification

Consider two parties split learning a model for a binary classification problem over the domain $\mathcal{X} \times \{0, 1\}$. Here the passive and active parties want to learn a composition model $h \circ f$ jointly, where the raw features X and $f : \mathcal{X} \to \mathbb{R}^d$ are stored the passive party side while the labels y and $h : \mathbb{R}^d \to \mathbb{R}$ is on the active party side. Let $\ell = h(f(X))$ be the logit of the positive class where the positive

class's predicted probability is given by the sigmoid function as follows: $\widetilde{p}_1 = 1/(1+\exp(-\ell))$. The loss of the model is given by the cross entropy $L = \log(1+\exp(-\ell)) + (1-y)\ell$. During the model inference, the passive party computes $f(X)$ and sends it to the active party who will then compute the rest of computation (Forward in Table 1). [2]

To train the model using gradient descent, the active party starts the gradient computation process by first computing the gradient of the loss with respect to the logit $\frac{dL}{d\ell} = (\widetilde{p}_1 - y)$. Using chain rule, the active party can then compute the gradient of L with respect to h's parameters through ℓ. In order to also allow the passive party to learn f, the active party also computes the gradient of L with respect to the input of the function h. We denote this gradient by $g \coloneqq \nabla_{f(X)} L = (\widetilde{p}_1 - y)\nabla_a h(a)|_{a=f(X)} \in \mathbb{R}^d$ (last equality by chain rule). After receiving g sent from the active party, the passive party can compute the gradient of L with respect to f's parameters (Backward in Table 1). Since the gradient g sent to the passive party is a function of both $f(X)$ and y, it is possible that g could contain information about the label y.

Forward:	X	$\xrightarrow{\boldsymbol{f}}$	$\boldsymbol{f}(X)$	$\overset{\text{comm.}}{\Longrightarrow}$	$\boldsymbol{f}(X)$	$\xrightarrow{\boldsymbol{h}}$	$\ell = h(f(X))$	$\rightarrow$	$\overset{y}{\downarrow}$ L
Backward:			g	$\overset{\text{comm.}}{\Longleftarrow}$	g	$\leftarrow$	$\frac{dL}{d\ell} = (\widetilde{p}_1 - y)$	$\leftarrow$	1
			$\downarrow$ f's param				$\downarrow$ h's param		

Table 1: Computation and communication diagram of model training ($\leftarrow$ and $\downarrow$ represent gradient computation using chain rule)

When B examples are forwarded as a batch, the communicated features $f(X)$ and gradients g will both be matrices of shape $\mathbb{R}^{B\times d}$ with each row belonging to a specific example in the batch. It is important to note that here the gradients as rows of the matrix are gradients of the loss with respect to different examples' intermediate computation results but not the model parameters; therefore, no averaging over or shuffling of the rows of the matrix can be done prior to communication for the sake of correct gradient computation of f's parameters on the passive party side.

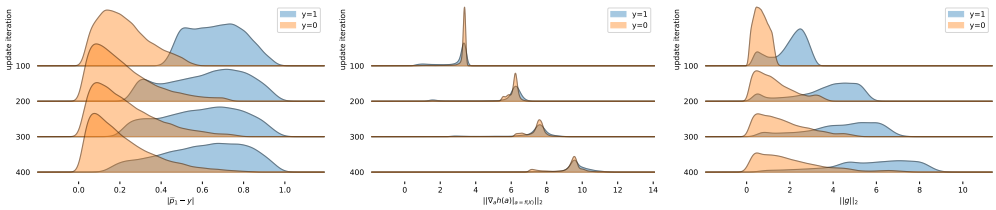
2.2 Norm-based label uncovering method

From the expression of g, we see that the norm of the communicated gradient is $\|g\|_2 = |\widetilde{p}_1 - y| \cdot \left\|\nabla_a h(a)|_{a=\boldsymbol{f}(X)}\right\|_2$. We identify two **observations** for $|\widetilde{p}_1 - y|$ and $\left\|\nabla_a h(a)|_{a=\boldsymbol{f}(X)}\right\|_2$ which hold true for a wide range of real world learning problems. Motivated by the observations together, we propose a method which can uncover the label y by using the gradient norm $\|g\|_2$.

- Observation 1: Almost throughout the entire training, the model tends to be **less confident** about a positive example being positive **than** a negative example being negative. In other words, the confidence gap $1 - \widetilde{p}_1$ of a positive example ($y = 1$) is typically larger than the confidence gap $1 - \widetilde{p}_0 = \widetilde{p}_1$ of a negative example ($y = 0$). Thus the value of $|\widetilde{p}_1 - y|$ is typically higher for positive examples than that of the negative examples (Figure 1.(a)). This observation is generally true for problems like advertising conversion prediction and disease prediction, where there is inherently more ambiguity for the positive class than the negative. For example, in the advertising conversion problem, uninterested users of a product will never click on its ad and convert, but those interested, even after clicking on the ad, might make the purchase only a fraction of the time depending on whether they at the moment have the time or money to execute the purchase.
- Observation 2: Almost throughout the training, the norm of the gradient vector $\|\nabla_a h(a)|_{a=\boldsymbol{f}(X)}\|_2$ is on the **same** order of magnitude for both the positive and negative examples (Figure 1.(b)). This is natural because $\nabla_a h(a)|_{a=\boldsymbol{f}(X)}$ is not a function of y.

As a consequence of these two observations, the gradient norm $\|g\|_2$ of the positive instances are generally **larger than** that of the negative ones (Figure 1.(c)). Thus $\|g\|_2$ is a very strong predictor of the unseen label y. We quantitatively measure how well $\|g\|_2$ can predict y by the AUC of the ROC curve when using $\|g\|_2$ as a predictor of y. We call this value *leak AUC*. This value can be estimated at every gradient update iteration using the minibatch of examples used in that batch. Higher leak

[2] For the simplifying of the notation and derivation, we add no additional features in the active party to compute the logit. The data leakage problem still holds true for other complicated settings (see in the experiment section).



(a) (b) (c)

Figure 1: We demonstrate these two observations and the consequence with a Wide & Deep model [6] trained on the Criteo dataset for click-through rate prediction (problem in online advertising similar to conversion prediction). Here the passive party f includes the embedding layer and the first two layers of a four layer multilayer perceptron (MLP) (the deep part); the active party includes the other two layers of the deep part together with the wide part of the model. Here the communicated gradient g is of the middle cut layer in the deep part. We show the distribution of $|\widetilde{p}_1 - y|$, $\left\|\nabla_a h(a)|_{a=\boldsymbol{f}(X)}\right\|_2$, and $\|g\|_2$ for both positive and negative examples after the first 100, 200, 300, 400 steps of stochastic gradient descent training in (a), (b), (c) respectively.

AUC means the label is leaked more consistently by the communicated gradient g. In the Criteo dataset experiment in Figure 1, the leak AUC for the four iterations presented are 0.9832, 0.8862, 0.9098, 0.9229, signaling generally a high level of label leakage throughout the model training.

3 Protecting against a general class of uncovering methods

Section 2.2 gives an existence proof that only communicating gradients but not the raw data can still leak the private label information. Beyond the norm attack, there exists a general class of uncovering methods based on the difference of distributions of gradients for positive and negative samples. In order to protect against label leakage, the active party should communicate the essential information of the gradient **without** communicating the actual gradient. Random perturbation is a class of powerful such methods. Among them a simple way is to add an isotropic Gaussian noise on the gradient to mix up the distributions of gradients g of positive samples and negative samples before sending them to the passive party. Larger noise will make leak AUC smaller but potentially could have a negative impact on the learned model's predictive performance.

Although adding **isotropic** Gaussian noise is an effective protection method, it may not be optimal due to the following two reasons. First, the gradients are vectors instead of scalars, so the direction of the noise matters. Isotropic noise might have neglected this direction information. Second, due to the asymmetry of the distribution, the active party could add different noise to the positive and negative gradients. By adding noise in a smarter way, one could conceivably achieve a better trade-off between label protection and model performance. In this section, we study where the optimal trade-off exists by mathematically defining the optimization problem for the trade-off and solving the optimization problem under Gaussianity assumption of the gradients.

3.1 Formulating the protection objective

Consider a general class of label recovering functions $\{\mathbb{1}_A, A \subset \mathbb{R}^d\}$. For any set $A \subset \mathbb{R}^d$, the labelling function $\mathbb{1}_A : \mathbb{R}^d \to \{0, 1\}$, maps the gradient $g \in A$ to the positive class 1 and $g \notin A$ to the negative class 0. This general class of functions encompasses the aforementioned 2-norm detection method by $A = \{g \in \mathbb{R}^d : ||g||_2 \geq t\}$ for some fixed threshold t. We denote the distribution of the additively perturbed positive and negative gradients $\tilde{g}^{(1)} = g^{(1)} + n^{(1)}$ and $\tilde{g}^{(0)} = g^{(0)} + n^{(0)}$ by $\widetilde{P}^{(1)}$ and $\widetilde{P}^{(0)}$ respectively. Here $n^{(i)}$ is a zero-mean random noise vector added independently to the random gradient $g^{(i)}$ sampled from examples of class $y = i$. Having zero-mean $n^{(i)}$ ensures the perturbed gradients with respect to f's parameters is unbiased. We define the *detection error* of a labelling function $\mathbb{1}_A$ under the perturbed gradient distribution $\widetilde{P}_1$ and $\widetilde{P}_0$ to be $\frac{1}{2}(\widetilde{P}^{(1)}(A^C) + \widetilde{P}^{(0)}(A))$, i.e. the average of the False Negative Rate and the False Positive Rate. To protect against the worst case labelling function, the active party wants to make the worst case labelling function's detection error to be as high as possible, which is equivalent to solving the following maximin optimization problem.

$$\max_{\widetilde{P}^{(1)}, \widetilde{P}^{(0)}} \min_{A} \frac{1}{2}(\widetilde{P}^{(1)}(A^C) + \widetilde{P}^{(0)}(A)) = \max_{\widetilde{P}^{(1)}, \widetilde{P}^{(0)}} \frac{1}{2}(1 - \text{TV}(\widetilde{P}^{(1)}, \widetilde{P}^{(0)})),$$

where $\mathrm{TV}(\cdot,\cdot)$ is the total variation distance and the equality follows directly from TV's definition. Thus this is equivalent to minimizing the total variation distance between $\tilde{P}^{(1)}$ and $\tilde{P}^{(0)}$. Because the TV distance between high dimensional distributions are not tractable, we seek to minimize a tractable upper bound of the total variation distance. By applying Pinsker's inequality and Jensen's inequality, we obtain an upper bound of total variation distance by the sum of KL divergence (sumKL)[3]: $\mathrm{TV}(P,Q) \leq \frac{1}{2}(\sqrt{\frac{\mathrm{KL}(P \parallel Q)}{2}} + \sqrt{\frac{\mathrm{KL}(Q \parallel P)}{2}}) \leq \frac{1}{2}\sqrt{\mathrm{KL}(P \parallel Q) + \mathrm{KL}(Q \parallel P)}$. Thus our optimization objective can be written as follows.

$$\min_{\widetilde{P}^{(1)},\widetilde{P}^{(0)}} \mathrm{KL}(\widetilde{P}^{(1)} \parallel \widetilde{P}^{(0)}) + \mathrm{KL}(\widetilde{P}^{(0)} \parallel \widetilde{P}^{(1)}).$$

In an extreme case, we can add infinitely large noise for both negative and positive gradients, then the sumKL will be 0 and the error probability will be 0.5 which is equivalent to a random guess. However, stochastic gradient descent cannot converge under infinitely large gradient noise, so it is necessary to control the variance of the additive noise. We thus introduce the noise power constraint: $p \cdot \mathrm{tr}(\mathrm{Cov}[n^{(1)}]) + (1-p) \cdot \mathrm{tr}(\mathrm{Cov}[n^{(0)}]) \leq P$, where p is the fraction of positive examples (known to the active party), $\mathrm{tr}(\mathrm{Cov}[n^{(i)}])$ denotes the trace of the covariance matrix of the random noise vector $n^{(i)}$, and the power constraint P is a tunable hyper-parameter to control the level of noise. Now the constrained optimization problem can be written as follows.

$$\begin{aligned} \min_{\widetilde{P}^{(1)},\widetilde{P}^{(0)}} \ & \mathrm{KL}(\widetilde{P}^{(1)} \parallel \widetilde{P}^{(0)}) + \mathrm{KL}(\widetilde{P}^{(0)} \parallel \widetilde{P}^{(1)}) \\ \text{s.t.} \quad & p \cdot \mathrm{tr}(\mathrm{Cov}[n^{(1)}]) + (1-p) \cdot \mathrm{tr}(\mathrm{Cov}[n^{(0)}]) \leq P. \end{aligned} \tag{1}$$

3.2 Optimizing the objective

We introduce some assumptions in order to solve the optimization problem. We assume that the distribution of gradients (across samples) are Gaussian: $g^{(1)} \sim \mathcal{N}(\bar{g}^{(1)}, uI_{d\times d})$ and $g^{(0)} \sim \mathcal{N}(\bar{g}^{(0)}, vI_{d\times d})$. Let $\Delta g = \bar{g}^{(1)} - \bar{g}^{(0)}$ denote the difference of the two mean vectors. Additionally, we consider Gaussian additive noise: $n^{(1)} \sim \mathcal{N}(0, \Sigma_1)$ and $n^{(0)} \sim \mathcal{N}(0, \Sigma_0)$ where the covariance matrices commute: $\Sigma_1\Sigma_0 = \Sigma_0\Sigma_1$. Then we have the following theorem.

Theorem 1. *The optimal Σ_1^* and Σ_0^* to* (1) *have the form*

$$\Sigma_1^* = \frac{\lambda_1^{(1)*} - \lambda_2^{(1)*}}{\|\Delta g\|_2^2}(\Delta g)(\Delta g)^T + \lambda_2^{(1)*} I_d, \qquad \Sigma_0^* = \frac{\lambda_1^{(0)*} - \lambda_2^{(0)*}}{\|\Delta g\|_2^2}(\Delta g)(\Delta g)^T + \lambda_2^{(0)*} I_d$$

where $(\lambda_1^{(0)*}, \lambda_2^{(0)*}, \lambda_1^{(1)*}, \lambda_2^{(1)*})$ are the solution to the following 4 variable optimization problem:

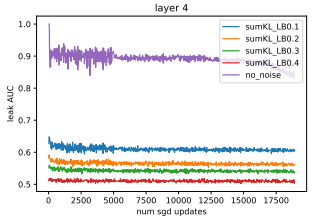
$$\begin{aligned} \min_{\lambda_1^{(0)},\lambda_1^{(1)},\lambda_2^{(0)},\lambda_2^{(1)}} \ & (d-1)\frac{\lambda_2^{(0)}+u}{\lambda_2^{(1)}+v} + (d-1)\frac{\lambda_2^{(1)}+v}{\lambda_2^{(0)}+u} + \frac{\lambda_1^{(0)}+u+\|\Delta g\|_2^2}{\lambda_1^{(1)}+v} + \frac{\lambda_1^{(1)}+v+\|\Delta g\|_2^2}{\lambda_1^{(0)}+u} \\ \textit{s.t.} \quad & p\lambda_1^{(1)} + p(d-1)\lambda_2^{(1)} + (1-p)\lambda_1^{(0)} + (1-p)(d-1)(\lambda_2^{(0)}) \leq P \\ & -\lambda_1^{(1)} \leq 0, \quad -\lambda_1^{(0)} \leq 0, \quad -\lambda_2^{(1)} \leq 0, \quad -\lambda_2^{(0)} \leq 0 \\ & \lambda_2^{(1)} - \lambda_1^{(1)} \leq 0, \qquad \lambda_2^{(0)} - \lambda_1^{(0)} \leq 0. \end{aligned}$$

The optimal noise consists of two independent components — one random component lies along the direction of the difference between the positive and negative gradient mean vectors Δg (whose covariance matrix of form $c\frac{\Delta g \Delta g^T}{\|\Delta g\|_2^2}$); the other component is sampled from an isotropic Gaussian. Σ_1^* and Σ_0^* have different mixture coefficients of these two components. The detailed derivation of the theorem can be found in the Appendix.

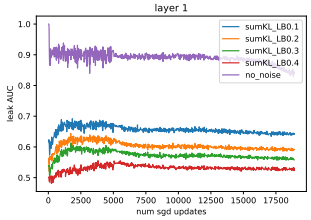
In practice, we can estimate $\bar{g}^{(1)}$, $\bar{g}^{(0)}$, u, and v using maximum likelihood estimation using the minibatch's g of the current communication round. With these constants computed, the 4-variable optimization problem can be easily solved by an alternating optimization algorithm similar to *Sequential Minimal Optimization* (SMO) used in SVM [20]. We note that by varying P we can achieve different trade-offs between label privacy and model performance. Larger P will give smaller sumKL*, and therefore smaller leak AUC, but worse model performance. To achieve a lower bound L of the worst case detection error, we simply require that the optimal sumKL value to be sumKL$^* \leq (2-4L)^2$ by the aforementioned upper bound of total variation distance using sumKL. We could achieve this by keeping on increasing the value of P used in the power constraint until the corresponding sumKL* satisfies this requirement.

[3] We use summation of KL divergences to maintain symmetry of the objective.

4 Experiments



(a)



(b)

	no noise	iso	sumKL
train loss	0.4067	0.4700	0.4672
test AUC	0.8062	0.7921	0.7949
	max leak AUC		
layer1	1.0000	0.5536	0.5554
layer2	1.0000	0.5652	0.5583
layer3	1.0000	0.6089	0.5710
layer4	1.0000	0.5129	0.5188

(c)

Figure 2: Figure (a) and (b) shows the leak AUC of no protection training and sumKL with different values of lower bound $L = [0.1, 0.2, 0.3, 0.4]$ at the cut layer (layer 4) and a previous layer (layer 1) respectively. Figure (c): Comparing sumKL with a baseline method (referred as iso in the table) where isotropic Gaussian noise is added to perturb the communicated gradients. In all the figures, no_noise is standard training with no gradient perturbation as protection).

In this section, we evaluate the proposed gradient perturbation method on the Criteo dataset [4], a real-world large-scale binary classification dataset with approximately 45 million user click records in online advertising. We split the dataset randomly into two parts: with 90% for training and the rest for tests. We train a Wide&Deep model [6] where the passive party f consists of the embedding layer for the input features and the four layers of 128-unit ReLU activated multilayer perceptron (deep part) and the active party consists of the last logit layer of the deep part and the entire wide part of the model. In every iteration, the passive party sends a minibatch of $65,536$ examples' 128-dimensional vector to the active party and the active party sends the gradients of the loss with respect to these vectors back to the passive side. We train the model for 30 epochs.

In Figure 2(a), we plot the leak AUC for every batch of communicated gradient at the cut layer (layer 4) throughout the entire training. We see that with no gradient perturbation, the leak AUC is consistently around 0.9 through the majority of training. On the other hand, when using our proposed technique (sumKL), we observe a big reduction in leak AUC, with leak AUC being lower (more privacy protection) when we impose a higher lower bound L for the worst case detection error. In addition, it is natural to ask whether the gradients of the layers before the cut layer can also leak the label information as the passive party keeps back propagating towards the first layer. In Figure 2(b), we plot the leak AUC of the first layer's activation gradients for the same set of methods shown in Figure 2(a). We notice that the leak AUC is at the same high level for the no protection method, indicating that previous layer's gradients could also leak the label information. However, our proposed method with different lower bound L can still significantly reduce the leak AUC value even though the protection was analyzed at the cut layer.

Lastly, we analyze how our technique compares to a baseline method where i.i.d. isotropic Gaussian noise is added to every gradient example g in the communicated minibatch. Because we care about a strong level of label protection, we compare our method that has the strongest privacy requirement $L = 0.4$ against a white Gaussian baseline where every g_i in the batch of size B is added with $\mathcal{N}(0, \frac{25 \cdot \max_{i=1}^{B} \|g_i\|_2^2}{d} I_{d \times d})$. We show the end-of-training loss, the best test AUC, and each deep layer's leak AUC value in Figure 2(c). As we can see, our proposed perturbation has achieved smaller training loss and higher test AUC while also having approximately the same if not better leak AUC. This shows that our optimization-based model has achieved a better privacy performance trade-off than adding isotropic Gaussian noise.

5 Conclusion

In this paper, we identify a simple norm-based method to uncover the private labels of the active party using the communicated gradients in the two-party split learning problem for binary classification. We then propose a theoretically principled method to determine the best additive random perturbation to the gradients to reduce the probability of label leakage against a general class of label uncovering methods. An open problem for future work is whether there exists other simple methods to uncover the label information from the gradients and understanding how our method could protect against those methods theoretically and empirically.

[4] https://www.kaggle.com/c/criteo-display-ad-challenge/data

Acknowledgements We would like to express our thanks to the team members at ByteDdance for providing the collaborative federated learning framework, Fedlearner[5]. In addition, we would like to thank Liangchao Wu, Di Wu, and Xiaobing Liu for their discussions and supports. The code is availale at `https://github.com/bytedance/fedlearner/tree/master/example/privacy/label_protection`.

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] N. Agrawal, A. Shahin Shamsabadi, M. J. Kusner, and A. Gascón. Quotient: two-party secure neural network training and prediction. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pages 1231–1247, 2019.

[3] Y. Aono, T. Hayashi, L. Wang, S. Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. IEEE Transactions on Information Forensics and Security, 13(5): 1333–1345, 2017.

[4] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers. Protection against reconstruction and its applications in private federated learning. arXiv preprint arXiv:1812.00984, 2018.

[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 privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1175–1191, 2017.

[6] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In Proceedings of the 1st workshop on deep learning for recommender systems, pages 7–10, 2016.

[7] A. Cheu, A. Smith, J. Ullman, D. Zeber, and M. Zhilyaev. Distributed differential privacy via shuffling. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 375–403. Springer, 2019.

[8] W. Du, Y. S. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In Proceedings of the 2004 SIAM international conference on data mining, pages 222–233. SIAM, 2004.

[9] C. Dwork. Differential privacy. In M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, editors, Automata, Languages and Programming, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35908-1.

[10] 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.

[11] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 2468–2479. SIAM, 2019.

[12] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557, 2017.

[13] O. Gupta and R. Raskar. Distributed learning of deep neural network over multiple agents. Journal of Network and Computer Applications, 116:1–8, 2018.

[14] L. Lyu, H. Yu, and Q. Yang. Threats to federated learning: A survey. arXiv preprint arXiv:2003.02133, 2020.

[5] `https://github.com/bytedance/fedlearner`

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics, pages 1273–1282. PMLR, 2017.

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

[17] H. B. McMahan, G. Andrew, U. Erlingsson, S. Chien, I. Mironov, N. Papernot, and P. Kairouz. A general approach to adding differential privacy to iterative training procedures. arXiv preprint arXiv:1812.06210, 2018.

[18] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft. Privacy-preserving ridge regression on hundreds of millions of records. In 2013 IEEE Symposium on Security and Privacy, pages 334–348. IEEE, 2013.

[19] V. Pichapati, A. T. Suresh, F. X. Yu, S. J. Reddi, and S. Kumar. Adaclip: Adaptive clipping for private sgd. arXiv preprint arXiv:1908.07643, 2019.

[20] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[21] A. Rajkumar and S. Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In N. D. Lawrence and M. Girolami, editors, Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, volume 22 of Proceedings of Machine Learning Research, pages 933–941, La Palma, Canary Islands, 21–23 Apr 2012. PMLR. URL `http://proceedings.mlr.press/v22/rajkumar12.html`.

[22] S. S. Sathya, P. Vepakomma, R. Raskar, R. Ramachandra, and S. Bhattacharya. A review of homomorphic encryption libraries for secure computation. arXiv preprint arXiv:1812.02428, 2018.

[23] P. Subramanyan, R. Sinha, I. Lebedev, S. Devadas, and S. A. Seshia. A formal foundation for secure remote execution of enclaves. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 2435–2450, 2017.

[24] O. Thakkar, G. Andrew, and H. B. McMahan. Differentially private learning with adaptive clipping. arXiv preprint arXiv:1905.03871, 2019.

[25] F. Tramer and D. Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. arXiv preprint arXiv:1806.03287, 2018.

[26] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. arXiv preprint arXiv:1812.00564, 2018.

[27] P. Vepakomma, O. Gupta, A. Dubey, and R. Raskar. Reducing leakage in distributed deep learning for sensitive health data. arXiv preprint arXiv:1812.00564, 2019.

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

[29] B. Zhao, K. R. Mopuri, and H. Bilen. idlg: Improved deep leakage from gradients. arXiv preprint arXiv:2001.02610, 2020.

[30] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In Advances in Neural Information Processing Systems, pages 14774–14784, 2019.

Appendix

A.1 Proof of theorem 1

Based on the assumption made in Section 3.2, the optimization problem we want to solve is:

$$\min_{\Sigma_0,\Sigma_1\in\mathbb{S}} \mathrm{KL}(\mathcal{N}(\bar{g}^{(1)}, uI+\Sigma_1) \parallel \mathcal{N}(\bar{g}^{(0)}, vI+\Sigma_0)) + \mathrm{KL}(\mathcal{N}(\bar{g}^{(0)}, vI+\Sigma_0) \parallel \mathcal{N}(\bar{g}^{(1)}, uI+\Sigma_1))$$

subject to

$$\begin{aligned} \Sigma_0\Sigma_1 &= \Sigma_1\Sigma_0 \\ p\cdot\mathrm{tr}(\Sigma_1) + (1-p)\cdot\mathrm{tr}(\Sigma_0) &\le P \\ \Sigma_1 &\succeq \mathbf{0} \\ \Sigma_0 &\succeq \mathbf{0}. \end{aligned}$$

By writing out the analytical close-form of the KL divergence between two Gaussian distributions, the optimization can be written as:

$$\begin{aligned} \min_{\Sigma_0,\Sigma_1\in\mathbb{S}} \ & \mathrm{tr}((\Sigma_1+vI)^{-1}(\Sigma_0+uI)) \\ & + \mathrm{tr}((\Sigma_0+uI)^{-1}(\Sigma_1+vI)) \\ & + (\bar{g}^{(1)}-\bar{g}^{(0)})^T\left((\Sigma_1+vI)^{-1}+(\Sigma_0+uI)^{-1}\right)(\bar{g}^{(1)}-\bar{g}^{(0)}) \end{aligned}$$

subject to

$$\begin{aligned} \Sigma_0\Sigma_1 &= \Sigma_1\Sigma_0 \\ p\cdot\mathrm{tr}(\Sigma_1) + (1-p)\cdot\mathrm{tr}(\Sigma_0) &\le P \\ \Sigma_1 &\succeq \mathbf{0} \\ \Sigma_0 &\succeq \mathbf{0}. \end{aligned}$$

By the commutative constraint on the two positive semidefinite matrices Σ_1 and Σ_0, we know that we can factor these two matrices using the same set of eigenvectors. We thus write:

$$\begin{aligned} \Sigma_0 =& Q^T\mathrm{diag}(\lambda_1^{(0)},\dots,\lambda_d^{(0)})Q \\ \Sigma_1 =& Q^T\mathrm{diag}(\lambda_1^{(1)},\dots,\lambda_d^{(1)})Q, \end{aligned}$$

where $Q\in\mathbb{R}^{d\times d}$ is an orthogonal matrix and the eigenvalues $\lambda_i^{(0)}, \lambda_i^{(1)}$ are nonnegative and decreasing in value.

Using this alternative expression of Σ_1 and Σ_0, we can express the optimization in terms of $\{\lambda_i^{(1)}\}, \{\lambda_i^{(0)}\}, Q$:

$$\begin{aligned} \min_{\{\lambda_i^{(1)}\},\{\lambda_i^{(0)}\},Q} & \sum_{i=1}^{d}\frac{\lambda_i^{(0)}+u}{\lambda_i^{(1)}+v} + \sum_{i=1}^{d}\frac{\lambda_i^{(1)}+v}{\lambda_i^{(0)}+u} \\ & + \left[Q(\bar{g}^{(1)}-\bar{g}^{(0)})\right]^T \mathrm{diag}\left(\dots,\frac{1}{\lambda_i^{(0)}+u}+\frac{1}{\lambda_i^{(1)}+v},\dots\right) Q(\bar{g}^{(1)}-\bar{g}^{(0)}) \end{aligned}$$

$$\begin{aligned} \text{subject to} \quad & p(\sum_{i=1}^{d}\lambda_i^{(1)}) + (1-p)(\sum_{i=1}^{d}\lambda_i^{(0)}) \le P \\ & -\lambda_i^{(1)} \le 0,\ \forall i\in[d] \\ & -\lambda_i^{(0)} \le 0,\ \forall i\in[d]. \\ & \lambda_i^{(1)} \ge \lambda_j^{(1)}, \forall i<j. \\ & \lambda_i^{(0)} \ge \lambda_j^{(0)}, \forall i<j. \\ & Q \ \text{ orthogonal.} \end{aligned}$$

For any fixed feasible $\{\lambda_i^{(1)}\}, \{\lambda_i^{(0)}\}$, we see that the corresponding minimizing Q will set its first row to be the unit vector in the direction of Δg. Thus by first minimizing Q, the optimization objective reduces to:

$$\begin{aligned}
&\sum_{i=1}^{d} \frac{\lambda_i^{(0)} + u}{\lambda_i^{(1)} + v} + \sum_{i=1}^{d} \frac{\lambda_i^{(1)} + v}{\lambda_1^{(0)} + u} + \frac{g}{\lambda_1^{(0)} + u} + \frac{g}{\lambda_1^{(1)} + v} \\
=&\sum_{i=2}^{d} \frac{\lambda_i^{(0)} + u}{\lambda_i^{(1)} + v} + \sum_{i=2}^{d} \frac{\lambda_i^{(1)} + v}{\lambda_i^{(0)} + u} + \frac{\lambda_1^{(1)} + v + \|\Delta g\|_2^2}{\lambda_1^{(0)} + u} + \frac{\lambda_1^{(0)} + u + \|\Delta g\|_2^2}{\lambda_1^{(1)} + v}.
\end{aligned}$$

We see that for the pair of variable $(\lambda_i^{(1)}, \lambda_i^{(0)})$ the function $\frac{\lambda_i^{(0)}+u}{\lambda_i^{(1)}+v} + \frac{\lambda_i^{(1)}+v}{\lambda_1^{(0)}+u}$ is strictly convex over the line segment $p\lambda_i^{(1)} + (1-p)\lambda_i^{(0)} = c$ for any positive c and attains the the minimum value at $\lambda_i^{(1)} = 0$ when $v < u$ and $\lambda_i^{(0)} = 0$ when $v \geq u$. Suppose without loss of generality $u < v$, then for the optimal solution we must have $\lambda_i^{(0)} = 0$ for all i. Under this condition, we notice that the function $x \mapsto \frac{u}{x+v} + \frac{x+v}{u}$ is strictly convex on the positive reals. Thus by Jensen inequality, the optimal solution's variables $\lambda_i^{(1)}$ must take on the same value for all i. As a result, we have proved that at the optimal solution, we must have:

$$\lambda_i^{(0)} = \lambda_j^{(0)} \text{ and } \lambda_i^{(1)} = \lambda_j^{(1)}, \text{ for } i, j \geq 2.$$

Hence, we have the equivalent optimization problem:

$$\begin{aligned}
\min_{\lambda_1^{(0)}, \lambda_1^{(1)}, \lambda_2^{(0)}, \lambda_2^{(1)}} \quad & (d-1)\frac{\lambda_2^{(0)} + u}{\lambda_2^{(1)} + v} + (d-1)\frac{\lambda_2^{(1)} + v}{\lambda_2^{(0)} + u} + \frac{\lambda_1^{(0)} + u + \|\Delta g\|_2^2}{\lambda_1^{(1)} + v} + \frac{\lambda_1^{(1)} + v + \|\Delta g\|_2^2}{\lambda_1^{(0)} + u} \\
\text{subject to} \quad & \\
& p\lambda_1^{(1)} + p(d-1)\lambda_2^{(1)} + (1-p)\lambda_1^{(0)} + (1-p)(d-1)(\lambda_2^{(0)}) \leq P \\
& -\lambda_1^{(1)} \leq 0 \\
& -\lambda_1^{(0)} \leq 0 \\
& -\lambda_2^{(1)} \leq 0 \\
& -\lambda_2^{(0)} \leq 0 \\
& \lambda_2^{(1)} - \lambda_1^{(1)} \leq 0 \\
& \lambda_2^{(0)} - \lambda_1^{(0)} \leq 0.
\end{aligned}$$

Given the optimal solution to the above 4-variable problem $(\lambda_1^{(0)*}, \lambda_2^{(0)*}, \lambda_1^{(1)*}, \lambda_2^{(1)*})$, we can set Q to be any orthogonal matrix whose first row is the vector $\frac{\Delta g}{\|\Delta g\|_2}$. Plugging this back into the expression of Σ_1 and Σ_0 gives us the final result.

Thus the proof is complete.

Remark. By analyzing the KKT condition of this four variable problem we can find that the optimal solution must occur on the hyperplane $p\lambda_1^{(1)} + p(d-1)\lambda_2^{(1)} + (1-p)\lambda_1^{(0)} + (1-p)(d-1)(\lambda_2^{(0)}) = P$. Knowing from the proof above that in addition $\lambda_2^{(1)} = 0$ if $u \geq v$ and $\lambda_2^{(0)} = 0$ if $u < v$ further reduces this problem to a 3-variable problem. Keeping one of them fixed and optimizing the other two reduces the problem to a line search over a convex objective. Alternatingly fixing one of these three variables while optimizing the other two would eventually converge and give us the optimal solution.