Technical Report of Label Leakage and Protection in Federated Learning

Problem Setup

- In the settings of the split learning, the passive party (ByteDance) provides user embeddings, and the active party (advertisers such as JD, Alibaba etc) owns the label information and its own embeddings (if possible). The passive party sends the split layer's embedding to the active party and receives the corresponding gradients from the active party.
- The active party would like to protect their label information. However, sharing gradients definitely leaks the ground-truth labels.

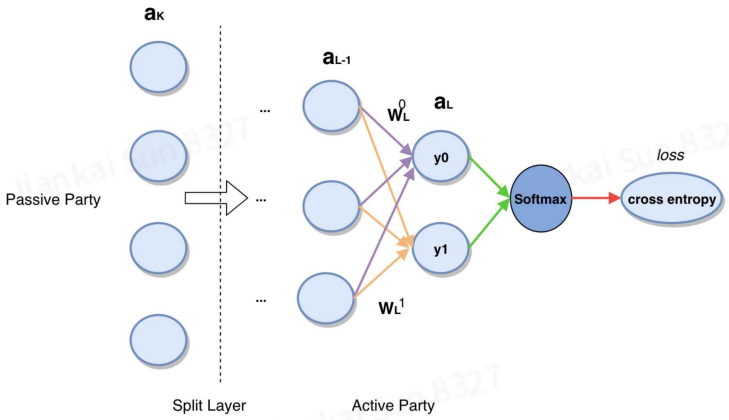


Figure 1: Illustration of the neural network.

Extracting Ground-truth Labels

We introduce a method that the passive party can use to extract the ground-truth labels from the active party via the shared gradients.

- The active party uses Softmax to calculate the loss

$$l(\mathbf{x}, c) = -log\frac{e^{y_c}}{\sum_j e^{y_j}} \qquad (1)$$

Where

- **x** is the input feature vectors
- c is the label

- **y** is the predicted logit score

$$p_i = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{4}$$

$$y_i = \mathbf{W}_L^{i\,T} \mathbf{a}_{L-1} + b_L^i \tag{5}$$

- p_i is the predicted probability of being label i

The gradient of the logit is:

$$g_i = \frac{\partial l(\mathbf{x}, c)}{\partial y_i} = -\frac{\partial log(e^{y_c}) - \partial log(\sum_j e^{y_j})}{\partial y_i}$$

$$= \begin{cases} -1 + \frac{e^{y_i}}{\sum_j e^{y_j}} & \text{if } i = c \\ \frac{e^{y_i}}{\sum_j e^{y_j}} & \text{else} \end{cases} \tag{2}$$

Then:

- given a positive instance: g_0 = p_0 = 1 - p1, g_1 = -1 + p1
- given a negative instance: g_0 = -1 + p_0 = -p1, g_1 = p1

The gradient of the activation layer:

$$\nabla \mathbf{a}_K = \sum_i \frac{\partial l(\mathbf{x}, c)}{\partial y_i} \cdot \frac{\partial y_i}{\partial \mathbf{a}_K} = \sum_i g_i \cdot \frac{\partial y_i}{\partial \mathbf{a}_K} = g_0 \cdot \frac{\partial y_0}{\partial \mathbf{a}_K} + g_1 \cdot \frac{\partial y_1}{\partial \mathbf{a}_K} \tag{12}$$

Hence, for positive instances:

$$\nabla \mathbf{a}_K^1 = (1 - p_1)(\frac{\partial y_0}{\partial \mathbf{a}_K} - \frac{\partial y_1}{\partial \mathbf{a}_K}) \tag{13}$$

For negative instances, we have:

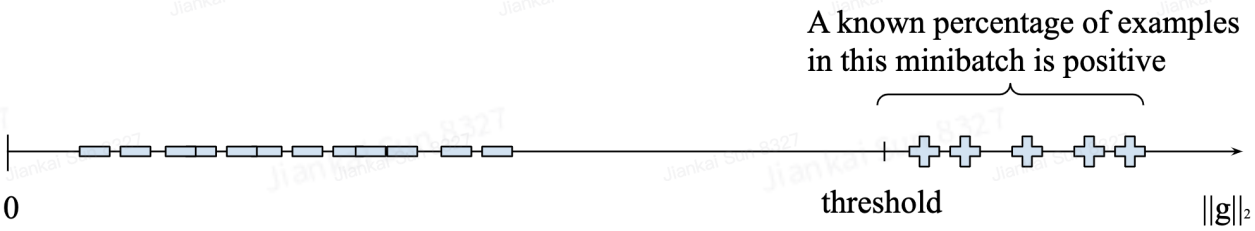
$$\nabla \mathbf{a}_K^0 = -p_1\left(\frac{\partial y_0}{\partial \mathbf{a}_K} - \frac{\partial y_1}{\partial \mathbf{a}_K}\right) \tag{14}$$

2-norm Attack

In the scenario of the online advertising, the number of positive instances are much smaller than the negative ones. For example, in the public Criteo dataset, the positive instance ratio is 25%. As a result, the gradient norm of the positive instances (||1-p1||) will be much larger than the negative ones (||p1||).

Thus we can uncover the label of examples in a minibatch by:

- ranking them according to the gradient norm
- take the top p percentage of the examples as positive



Empirically, we achieved very high leakage auc (~1.0) with the above 2-norm attacking method on different datasets.

Protection Methods

The fundamental idea of our protecting methods is we would like to add some perturbation to the gradients ($\mathbf{g}$) sent back to the passive party to make the attacker not distinguish the difference between positive and negative instances easily.

Previous training diagram (without protection):

Model Training (minibatch) (diagram)

$$\begin{array}{lccccccccc} \text{forward:} & X & \xrightarrow{\boldsymbol{f}} & \boldsymbol{f}(X) & \overset{\text{comm.}}{\Longrightarrow} & \boldsymbol{f}(X) & \xrightarrow{h} & \ell & \to & L \\ \text{Back:} & & & g & \overset{\text{comm.}}{\Longleftarrow} & g & \leftarrow & \frac{dL}{d\ell} = (\tilde{p_1} - y) & \leftarrow & 1 \\ & & & \downarrow & & & & \downarrow & & \\ & & & f\text{'s param} & & & & h\text{'s param} & & \end{array}$$

$$g = \underbrace{(\tilde{p_1} - y)}_{\in \mathbb{R}} \nabla_a h(a)\Big|_{a=\boldsymbol{f}(X)}, \text{ for every } (X, y) \text{ in the minibatch.}$$

The **g hat** is the perturbated **g** and **g hat** will be sent back to the passive party**:**

Model Training (minibatch) (diagram)

$$\begin{array}{lccccccccc} \text{forw.:} & X & \xrightarrow{\boldsymbol{f}} & \boldsymbol{f}(X) & \overset{\text{comm.}}{\Longrightarrow} & & & \boldsymbol{f}(X) & \xrightarrow{h} & \ell & \to & L \\ \text{back.:} & & & \tilde{g} & \overset{\text{comm.}}{\Longleftarrow} & \tilde{g} & \leftarrow & g & \leftarrow & \frac{dL}{d\ell} = (\tilde{p_1} - y) & \leftarrow & 1 \\ & & & \downarrow & & & & & & \downarrow & & \\ & & & f\text{'s param} & & & & & & h\text{'s param} & & \end{array}$$

Here, we provide two different ways as following to generate the perturbated **g hat**.

Max Norm Alignment

sumKL Minimization

The max norm alignment methods can prevent the passive party extracting the groud-truth labels via 2-norm attack. However, it may not be effective for some other attacks (unknown so far) which do not leverage gradient norms. Another drawback is it does not have a theoretical guarantee for the protection ability.

Here, we introduce a way to make the gradients of positive and negative instances undistinguishable from the gradient distribution perspective. We also give a theoretical guarantee for this protection method.

Optimization Setup

Minimizing the sumKL

Suppose u and v are standard deviations of the noise added to the negative and positive instance respectively:

Approximate $\Sigma_- \approx uI$ and $\Sigma_+ \approx vI$, we now have

$$\begin{aligned}\min_{\Sigma_1,\Sigma_0 \in \mathbb{S}_+^{d\times d}} & \mathrm{KL}(\mathcal{N}(g^{(1)}, vI + \Sigma_1) \parallel \mathcal{N}(g^{(0)}, uI + \Sigma_0)) \\ & + \mathrm{KL}(\mathcal{N}(g^{(0)}, uI + \Sigma_0) \parallel \mathcal{N}(g^{(0)}, vI + \Sigma_0)) \\ \text{subject to} \quad & p \cdot \mathrm{tr}(\Sigma_1) + (1-p) \cdot \mathrm{tr}(\Sigma_0) \leq P\end{aligned}$$

Under this assumption $\Sigma_- = uI_d$ and $\Sigma_+ = vI_d$ and $u, v > 0$, we express

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

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

Now the optimization becomes

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

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

Through simplication, we can convert this problem into a **4** variable optimization problem instead of $O(d^2)$. $(c \coloneqq \left\|g^{(1)} - g^{(0)}\right\|_2^2)$

$$\min_{\lambda_1^{(0)},\lambda_1^{(1)},\lambda_2^{(0)},\lambda_2^{(1)}} (d-1)\frac{\lambda_2^{(0)} + u}{\lambda_2^{(1)} + v} + (d-1)\frac{\lambda_2^{(1)} + v}{\lambda_2^{(0)} + u} + \frac{\lambda_1^{(0)} + u + c}{\lambda_1^{(1)} + v} + \frac{\lambda_1^{(1)} + v + c}{\lambda_1^{(0)} + u}$$

$$\begin{aligned}\text{subject to} \quad & p\lambda_1^{(1)} + p(d-1)\lambda_2^{(1)} + (1-p)\lambda_1^{(0)} + (1-p)(d-1)(\lambda_2^{(0)}) \leq P \\ & -\lambda_1^{(1)} \leq 0, \quad -\lambda_1^{(0)} \leq 0, \quad -\lambda_2^{(1)} \leq 0, \quad -\lambda_2^{(0)} \leq 0 \\ & \lambda_2^{(1)} - \lambda_1^{(1)} \leq 0 \\ & \lambda_2^{(0)} - \lambda_1^{(0)} \leq 0.\end{aligned}$$

With insights from the KKT conditions, we can optimize this very quickly through an alternating minimization algorithm like SMO used in SVM solver.

After we solve all the lambdas, the covariance matrix of noise to be added to the negative instances are:

With solution to (10) $(\lambda_1^{(0)*}, \lambda_2^{(0)*}, \lambda_1^{(1)*}, \lambda_2^{(1)*})$, we see that

$$\begin{aligned}\Sigma_0^* &= Q^T \text{diag}(\lambda_1^{(0)*}, \lambda_2^{(0)*} \ldots, \lambda_2^{(0)*}) Q \\ &= \frac{\lambda_1^{(0)*} - \lambda_2^{(0)*}}{\|g^{(1)} - g^{(0)}\|_2^2}(g^{(1)} - g^{(0)})(g^{(1)} - g^{(0)})^T + Q^T \text{diag}(\lambda_2^{(0)*}, \lambda_2^{(0)*} \ldots, \lambda_2^{(0)*}) Q \\ &= \frac{\lambda_1^{(0)*} - \lambda_2^{(0)*}}{\|g^{(1)} - g^{(0)}\|_2^2}(g^{(1)} - g^{(0)})(g^{(1)} - g^{(0)})^T + \lambda_2^{(0)*} I_d\end{aligned}$$

We know that the distribution of $X \sim \mathcal{N}(\mathbf{0}, \Sigma_0^*)Y + Z$, where

$$\begin{aligned}Y &\sim \mathcal{N}(\mathbf{0}, \frac{\lambda_1^{(0)*} - \lambda_2^{(0)*}}{\|g^{(1)} - g^{(0)}\|_2^2}(g^{(1)} - g^{(0)})(g^{(1)} - g^{(0)})^T), \\ Z &\sim \mathcal{N}(\mathbf{0}, \lambda_2^{(0)*} I_d).\end{aligned}$$

For $W = \sigma \epsilon g$, where $\epsilon \sim \mathcal{N}(0, 1)$. Then

$$\begin{aligned}\text{Cov}(W) &= \mathbb{E}[(W - \mathbb{E}[W])(W - \mathbb{E}[W])^T] \\ &= \mathbb{E}[\sigma^2 \epsilon^2 g g^T] \\ &= \sigma^2 \, \mathbb{E}[\epsilon^2] g g^T \\ &= \sigma^2 g g^T\end{aligned}$$

As a result, we can construct Y as

$$Y = \epsilon \left(\sqrt{\lambda_1^{(0)*} - \lambda_2^{(0)*}} \right) \frac{g^{(1)} - g^{(0)}}{\|g^{(1)} - g^{(0)}\|_2}, \ \epsilon \sim \mathcal{N}(0, 1)$$

And we can construct Z as

$$Z = \sqrt{\lambda_2^{(0)*}} \boldsymbol{\epsilon}, \ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I_d)$$

Y, Z are the corresponding Gaussian noise we should add to perturbate the gradients of the negative instances.

Regarding positive instances,

Similarly, we have

$$\begin{aligned}\Sigma_1^* &= Q^T \text{diag}(\lambda_1^{(1)*}, \lambda_2^{(1)*}, \ldots, \lambda_2^{(1)*}) Q \\ &= \frac{\lambda_1^{(1)*} - \lambda_2^{(1)*}}{\|g^{(1)} - g^{(0)}\|_2^2} (g^{(1)} - g^{(0)})(g^{(1)} - g^{(0)})^T + \lambda_2^{(1)*} I_d\end{aligned}$$

Approximate Theoretical Guarantees

As a result of this observation, it is natural to require a lower bound L of $\text{Error}_{0.5}$ for our final perturbed gradient distribution $(\tilde{P}^{(1)}, \tilde{P}^{(0)})$. To achieve this, we require that

$$\text{Error}_{0.5}(\tilde{P}^{(1)}, \tilde{P}^{(0)}) \geq \frac{1}{2} - \frac{\sqrt{\text{sumKL}^*}}{4} \geq L$$

Given the lower bound L of the error probability, we introduce a way to find the best P value dynamically which can satisfy our optimization requirements.

tion of the learning problem. Hence the "best" P is the smallest value of P such that the covariance solution $\widetilde{\Sigma}_1$ and $\widetilde{\Sigma}_0$ satisfies exactly $\text{sumKL}^* = (2 - 4L)^2$. To approximately achieve this, we find P use a search algorithm:

$P \leftarrow g$.

If $\text{sumKL}^* > (2 - 4L)^2$:

$P \leftarrow 1.5 * P$.

Else:

Stop and use this value of P.

We can control our solution's sumKL^* by changing the power constraint hyperparameter P that governs the amount of variance of the noise.

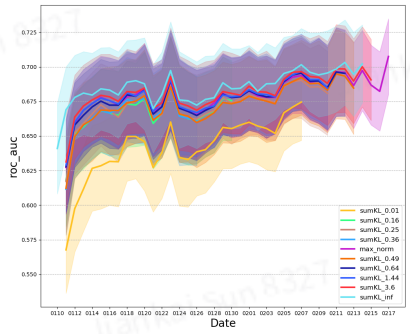
Thus we can very achieve different lower bounds of the adversary's error probability such as $40\%, 30\%, 20\%, 10\%$ etc by changing P.

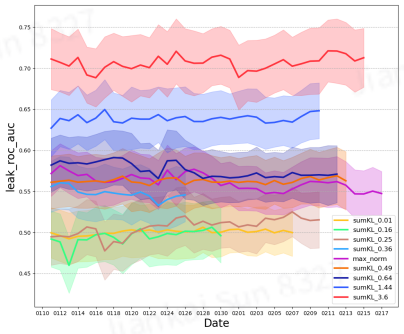
- Error lower bound by 40% requires $\text{sumKL}^* \leq 0.16$.
- Error lower bound by 30% requires $\text{sumKL}^* \leq 0.64$.

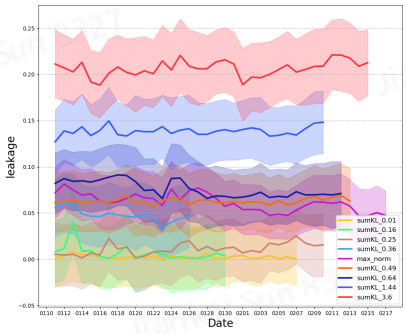
Experiments

Effectiveness

We tested above algorithms on a real CVR dataset which starts from 2020.01.11. We show the corresponding comparisons with confidence bands as follows:







- roc_auc: shows the mean performance auc per day with confidence bands
- leak_roc_auc: shows the mean leakage auc per day with confidence bands
- Leakage: shows |leak_roc_auc - 0.5|
- From the above figures, we can observe:
  - Performance: sumKL_inf > [sumKL_3.6 ~ max_norm ~ sumKL_1.44 ~ sumKL_0.64~sumKL_0.49] > sumKL_0.16 > sumKL_0.01.
    - The average diff between sumKL_inf and sumKL_0.01 is 0.04
    - The average diff between sumKL_inf and sumKL_0.16 is 0.014
    - The average diff between sumKL_inf and sumKL_0.49 is 0.012
  - Leakage (|leak_auc - 0.5|): sumKL_inf >> sumKL_3.6 >> >> sumKL_1.44 >> sumKL_0.64 > sumKL_0.49 > max_norm >> sumKL_0.16 > sumKL_0.01
- By varying the value of sumKL, we can see the trade off between training perforance and label protection effectiveness. A reasonable sumKL value can be set between 0.16 and 0.64.
  - The corresponding performance auc drop will be less than 0.02, and the leak_auc will be between [0.5, 0.6].

Efficiency

We also tested the efficiency of the sumKL minization method. Currently, we leverage tf.py_func (a little bit slow in comparison with using C++ ops) to implement the corresponding function to call the alternating minimization algorithm to compute lamdas. For example, when we set sumKL 0.04, the total time cost of computing lamdas is about 70ms.

Name	Wall Duration	Self time	Average Wall Duration	Occurrences	Event(s)	Link
PyFunc	68.124 ms	68.124 ms	68.124 ms	1	Incoming flow	input_layer/slot_1001_embedding/AsString
IteratorGetNext	61.912 ms	61.912 ms	61.912 ms	1	Incoming flow	input_layer/slot_100_embedding/AsString
RecvTensor	9.399 ms	9.118 ms	0.409 ms	23	Incoming flow	input_layer/slot_4_embedding/AsString
AsString	1.489 ms	1.489 ms	0.186 ms	8	Incoming flow	input_layer/slot_999_embedding/AsString
Const	1.467 ms	1.467 ms	0.025 ms	59	Incoming flow	input_layer/slot_900_embedding/AsString
GatherV2	1.378 ms	1.378 ms	0.020 ms	68	Incoming flow	input_layer/slot_73_embedding/slot_73_embedding_weights/embedding_lookup_sparse/Unique
Tile	1.374 ms	1.374 ms	0.060 ms	23	Incoming flow	input_layer/slot_73_embedding/slot_73_embedding_weights/embedding_lookup_sparse/Unique:1
					Incoming flow	EagerPyFunc_1

Conclusion

Summary

- We propose to use 2-norm attack method to extract the ground-truth labels of the the active party.
- We propose two different methods to prevent the label leakage
  - Max norm alignment
    - Can prevent 2-norm attack
  - sumKL minimization
    - With approximate theoretical guarantees
- We conducted several experiments to demonstrate the effectiveness of our proposed methods
- Code: https://github.com/bytedance/fedlearner/tree/master/example/privacy/label_protection

Future work

1. Design some new attack methods which avoid to use the gradient norm information
2. Speed up the alternating minimization algorithm by converting it to C++ op if necessary

Acknowledgement

- We have applied patents for our two protection methods: max norm alignment and sumKL minimization
- We would like to express our thanks to all the collaborators. Without them, we cannot finish the project.
  - Main contributors in ByteDance: Jiankai Sun, Weihao Gao, Hongyi Zhang, Junyuan Xie, Liangchao Wu, Chong Wang
  - Main intern contributors: Runliang Li, Xin Yang

- If you have any questions/suggestions, please contact: jiankai.sun@bytedance.com

References

[1] iDLG: Improved Deep Leakage from Gradients [arXiv]