Exercise 7 - Solution

Federated Learning

Reliable and Trustworthy Artificial Intelligence
ETH Zurich

Problem 1 (Analytical Gradient Inversion)**.** In this question, we are considering a simple 2-layer neural network with a hidden layer $o_2 \in \mathbb{R}^m$ that takes as input $x \in \mathbb{R}^n$ and produces a single binary classification output $p \in \mathbb{R}$:

$$\begin{aligned} z_1 &= W_1 \cdot x + b_1 \\ o_2 &= \text{ReLU}(z_1) \\ z_2 &= w_2 \cdot o_2 + b_2 \\ p &= \frac{1}{1 + e^{-z_2}}. \end{aligned} \tag{1}$$

We train it using federated learning, where each client k is using a binary cross entropy error function applied on its private data $(x_i, y_i) \sim \mathcal{D}_k$:

$$\mathcal{L}(x_i, y_i) = y_i \cdot \log(p(x_i)) + (1 - y_i) \cdot \log(1 - p(x_i)). \tag{2}$$

1. Use the chain rule to calculate the gradients sent to the server by client k on the data (x_i, y_i).
   Hint: The gradients are $\nabla_{w_2}\mathcal{L}(x_i, y_i)$, $\nabla_{b_2}\mathcal{L}(x_i, y_i)$, $\nabla_{W_1}\mathcal{L}(x_i, y_i)$ and $\nabla_{b_1}\mathcal{L}(x_i, y_i)$.

2. Use the calculated gradients to derive analytical formula for x_i in terms of the gradients calculated in the previous question. When is this formula valid?
   Hint: Look at the chain rule formulas for $\nabla_{W_1}\mathcal{L}(x_i, y_i)$ and $\nabla_{b_1}\mathcal{L}(x_i, y_i)$.

For further insight on analytical gradient inversion see [1, 2, 3].

Solution 1. 1. The derivatives are given below. We use $\mathbb{I}$ to denote the indicator function. We have boxed the gradients updates that are sent by the client to the server.

$$\nabla_p \mathcal{L}(x_i, y_i) = \frac{y_i}{p} - \frac{1-y_i}{1-p}$$

$$\nabla_{z_2} \mathcal{L}(x_i, y_i) = \frac{\partial \mathcal{L}}{\partial p}\frac{\partial p}{\partial z_2} = \nabla_p \mathcal{L}(x_i, y_i)(\frac{1}{1+e^{-z_2}})(1 - \frac{1}{1+e^{-z_2}})$$

$$\nabla_{w_2} \mathcal{L}(x_i, y_i) = \frac{\partial \mathcal{L}}{\partial z_2}\frac{\partial z_2}{\partial w_2} = \nabla_{z_2} \mathcal{L}(x_i, y_i) o_2$$

$$\nabla_{b_2} \mathcal{L}(x_i, y_i) = \frac{\partial \mathcal{L}}{\partial z_2}\frac{\partial z_2}{\partial b_2} = \nabla_{z_2} \mathcal{L}(x_i, y_i)$$

$$\nabla_{o_2} \mathcal{L}(x_i, y_i) = \frac{\partial \mathcal{L}}{\partial z_2}\frac{\partial z_2}{\partial o_2} = \nabla_{z_2} \mathcal{L}(x_i, y_i) w_2$$

$$\nabla_{z_1} \mathcal{L}(x_i, y_i) = \frac{\partial \mathcal{L}}{\partial o_2}\frac{\partial o_2}{\partial z_1} = \nabla_{o_2} \mathcal{L}(x_i, y_i) \mathbb{I}(z_1 > 0)$$

$$\nabla_{W_1} \mathcal{L}(x_i, y_i) = \frac{\partial \mathcal{L}}{\partial z_1}\frac{\partial z_1}{\partial W_1} = \nabla_{z_1} \mathcal{L}(x_i, y_i) M,$$

where M is a 3D tensor with entries: $M_{m,n,l} = \mathbb{I}(m = n)[x_i]_l$

$$\nabla_{b_1} \mathcal{L}(x_i, y_i) = \frac{\partial \mathcal{L}}{\partial z_1}\frac{\partial z_1}{\partial b_1} = \nabla_{z_1} \mathcal{L}(x_i, y_i)$$

2. We see from the fist part of the question that the gradient updates $\nabla_{W_1} \mathcal{L}(x_i, y_i)$ and $\nabla_{b_1} \mathcal{L}(x_i, y_i)$ share the non-disclosed gradient $\nabla_{z_1} \mathcal{L}(x_i, y_i)$. We can use this fact to find x_i. In particular, for the j^{th} entry in z_1, we have:

$$[z_1]_j = [W_1]_{(j,\cdot)} \cdot x_i + [b_1]_j$$

with $[W_1]_{(j,\cdot)}$ denoting the j^{th} row of W_1. Therefore, the j^{th} row of the gradient $\nabla_{W_1} \mathcal{L}(x_i, y_i)$ is

$$[\nabla_{W_1} \mathcal{L}(x_i, y_i)]_{(j,\cdot)} = [\nabla_{z_1} \mathcal{L}(x_i, y_i)]_j \, x_i$$

and the j^{th} entry of $\nabla_{b_1} \mathcal{L}(x_i, y_i)$ is

$$[\nabla_{b_1} \mathcal{L}(x_i, y_i)]_j = [\nabla_{z_1} \mathcal{L}(x_i, y_i)]_j.$$

Therefore, x_i can be found if $[\nabla_{z_1} \mathcal{L}(x_i, y_i)]_j \neq 0$ for some j by the formula

$$x_i = [\nabla_{W_1} \mathcal{L}(x_i, y_i)]_{(j,\cdot)} / [\nabla_{b_1} \mathcal{L}(x_i, y_i)]_j.$$

Note that $[\nabla_{z_1} \mathcal{L}(x_i, y_i)]_j == 0$, when $\mathbb{I}([z_1]_j \leq 0)$. Therefore, we need at least one hidden neuron that is activated in order to find x_i analytically. This is expected, since if all of the hidden neurons are not activated, $\mathcal{L}$ does not depend on x_i.

Problem 2 (Optimization-based Gradient Inversion)**.** In this question, we will implement the optimization-based gradient inversion techniques we described in the lecture. The code you are asked to complete is provided in the form of a Jupyter notebook here.

Solution 2. The solution to the notebook is given at here.

References

[1] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. "Privacy-preserving deep learning: Revisited and enhanced". In: *International Conference on Applications and Techniques in Information Security*. Springer. 2017, pp. 100–110.

[2] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. "Inverting Gradients–How easy is it to break privacy in federated learning?" In: *arXiv preprint arXiv:2003.14053* (2020).

[3] Junyi Zhu and Matthew Blaschko. "R-gap: Recursive gradient attack on privacy". In: *arXiv preprint arXiv:2010.07733* (2020).