Homework 4

Deadline: 30th December

Objective

In this assignment, you will implement a differentially private stochastic gradient descent (DPSGD) algorithm for a classification task on the MNIST dataset. You will use the Opacus library to handle privacy accounting, and implement clipping and noise addition operations within the DPSGD algorithm. We will assume everyone is fluent in Python.

Background

Differentially Private Stochastic Gradient Descent (DPSGD) [1] is an adaptation of SGD that provides privacy guarantees for individuals' data within a dataset. The key idea in DPSGD is to limit the influence of any single data point on the gradient updates by:

- **Clipping gradients**: Scaling down large gradients to limit their contribution.
- **Adding noise**: Introducing random noise to the (aggregated) clipped gradients to obscure the effect of individual data points.

The Opacus library is a PyTorch-based tool designed to facilitate the implementation of differential privacy in machine learning models by handling privacy accounting.

Task Description

In this homework, you will:

1. Use Opacus for privacy accounting, understanding basic "ensuring differential privacy via randomness calibration".
2. Implement the specific gradient clipping operation.
3. Evaluate the performance of your DPSGD implementation on MNIST for a simple classification task.

Tasks

We will use Google Colab to run the code so that everyone can access accelerators. I have added all necessary comments alongside the code provided. And you are going to finish the `TODO`s.

Using Opacus for Privacy Accounting (4 points): You will utilize the Opacus library to manage privacy accounting. Opacus automatically tracks the cumulative privacy loss, helping you understand the trade-off between model accuracy and privacy. Despite its convenience, we will hack a bit to appreciate some core operations under the hood.

`you should implement a function that determines the standard deviation of the Gaussian noise given a privacy budget. Specifically, use binary search.`

This is probably the most basic problem for privacy machine learning. Opacus has functions to implement such a routine, but we need to implement it ourselves.

Implementing Gradient Clipping (4 points): you need to implement the clipping operation. In Pytorch, suppose the model contains d trainable parameters, and each batch contains B sampled data examples, the gradient

is not trivially stored as a vector of shape $(d,)$, instead, it is stored as an iterator/container $[p_1, p_2, \cdots, p_k]$ such that p_i is a pytorch tensor and we have k layers and

$$sum([sum(p_1.shape), sum(p_2.shape), \cdots, sum(p_k.shape)]) = d$$

.
Accordingly, the computed per-example gradient iterator/container is $[q_1, q_2, \cdots, q_k]$ where q_i is a pytorch tensor with shape $(B, p_i.shape)$. If you index $q_i[j]$, you derive the j-th data's gradient concerning the part for i-th layer.

`Your task is to clip the gradient for each per-example.`

Get a feeling on privacy-utility trade-offs (2 points):

`Your task is to derive a curve where x-axis is` ε `and y-axis is the test accuracy at fixed` $\delta = 10^{-5}$.

Deliverables

Please submit the following:

1. The notebook sent to you with all TODO finished.
2. A brief report containing a figure showing how the privacy guarantee affects the utility. Specifically for the report, you will need to write some additional code where you set different ε values, compute the final test accuracy, and derive the curve. You do not have to submit code for this task.

Additional Notes

You may refer to the Opacus documentation for additional details on privacy accounting and implementation:

`https://opacus.ai`

For any questions, please contact me by zihang.xiang@kaust.edu.sa

References

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