DA 5001/6400 (July-Nov 2024): HW2

IIT Madras

Due Date: September 15, 2024 at 11:59 PM

Instructions

- The maximum score on the homework is 100 marks, including the bonus (extra credit) exercises.
- List the names of students you collaborated with for the homework. Also, cite any books, notes, or web resources you used for any problem. Failure to do so will be considered a violation of the honour code.
- If you used an LLM for help in one of the exercises, you must specify the name of the LLM and the exact prompt used.
- Corrections to the homework are shown in blue. Please note them.
- **Last updated**: Wednesday, September 11th at 5 pm.
- When submitting your solution on gradescope, please mark the pages corresponding to each solution along with the
- Please also submit you r code in a single zip/tar archive under the assignment "HW2-code" on gradescope.

0 Honour Code (0 marks but mandatory)

Please read the full honour code on the course webpage and write "I ACCEPT THE HONOUR CODE" in your submission. **Your submission will not be graded if you do not accept the honour code.**

1 Fill in the Course Feedback Form (5 marks)

Please give your feedback on the course so far in this anonymous form: `https://forms.gle/A2Fcw33CqiFbzbjd7`.

Write "I HAVE COMPLETED THE FEEDBACK FORM" in your written solutions once you have completed the form. Your feedback will really help improve the course and adapt it to your learning objectives.

2 Vector-Valued Laplace/Gaussian Mechanisms (2 + 2 + 2 + 4 = 10 marks)

In the course so far, we only considered privatizing deterministic algorithms with scalar outputs. Now, we will see how to extend that to vector-valued algorithms of the form $A : \mathcal{X}^* \to \mathbb{R}^d$.

A key generalization is to define an appropriate notion of sensitivity. Since the algorithm return a vector, there are multiple ways to reduce the per-component sensitivities to a scalar for the entire algorithm. For a real $p \geq 1$, define the ℓ_p-sensitivity as

$$\Delta_p := \sup_{D \simeq D'} \|A(D) - A(D')\|_p \,,$$

where $\|x\|_p = \left(\sum_{i=1}^d |x_i|^p\right)^{1/p}$ is the ℓ_p norm of the vector $x \in \mathbb{R}^d$. We will commonly use the ℓ_1 and ℓ_2 sensitivities.

1. **Laplace Mechanism**: Let the randomized algorithm $\mathcal{A}(D)$ return $A(D) + (\xi_1, \ldots, \xi_d)$ where $\xi_i \sim \text{Laplace}(0, \Delta_1/\varepsilon)$. Show that $\mathcal{A}$ is ε-DP. In other words, the Laplace mechanism scales with the ℓ_1 sensitivity of the algorithm.

2. **Gaussian Mechanism**: Consider the randomized algorithm $\mathcal{A}(D) \sim \mathcal{N}\big(A(D), \sigma^2 I_d\big)$ be Gaussian distributed with covariance matrix $\sigma^2 I_d$, where I_d is the $d \times d$ identity matrix. Then, $\mathcal{A}$ satisfies ρ-zCDP with $\rho = \frac{\Delta_2^2}{2\sigma^2}$.

3. **Comparison of sensitivities**: Prove that we have

   $$\Delta_2 \leq \Delta_1 \leq \sqrt{d}\ \Delta_2 \,.$$

   In other words, the multivariate Gaussian mechanism always has a smaller sensitivity than a multivariate Laplace mechanism. It can be up to $\sqrt{d}$ smaller — think of what this number is for AI model training. **Hint**.[1]

4. **Gaussian Mechanism with a Linear Map**: Consider an input dataset $x \in [0,1]^n$ over n numbers between 0 and 1. (Notice that we represent it with a vector x obtained by stacking all the numbers, instead of a set.) Consider the deterministic algorithm $A(x) = Mx$ for some matrix $M \in \mathbb{R}^{m \times n}$. Show that its ℓ_2 sensitivity is $\Delta_2 = \|M\|_{1\to 2} := \max_{i \in [n]} \|M[:, i]\|_2$ is the maximum norm of any column of M. Hence, conclude that $\mathcal{A}(x) = \mathcal{N}(Mx, \sigma^2 I_m)$ is ρ-zCDP with $\rho = \|M\|_{1\to 2}^2/(2\sigma^2)$. **Note**.[2] **Context**: This shows up with correlated noise DP mechanisms.

3 Completing the proof of amplification by sampling (2 + 1 + 1 = 5 marks)

We wish to establish that for any (ε, δ)-DP algorithm $\mathcal{A}$, its Poisson subsampled (w.p. p) version $\mathcal{A} \circ \mathcal{S}_p$ is $(1 + p(e^\varepsilon - 1), p\delta)$-DP. For D' which obtained by replacing an element of D with a null element, we proved

[1] **Hint**: For any vector $u \in \mathbb{R}^d$, prove that

$$\|u\|_2 \leq \|u\|_1 \leq \sqrt{d}\ \|u\|_2 \,.$$

[2] The induced matrix norm $\|M\|_{1\to 2}$ is defined as the following:

$$\|M\|_{1\to 2} = \sup_{x \neq 0} \frac{\|Mx\|_2}{\|x\|_1} \,.$$

It turns out to be equal to $\max_i \|M[:, i]\|_2$. This result comes from the fact that, to maximize $\|Mx\|_2$ under the constraint $\|x\|_1 = 1$, the vector x should place all its "weight" on the column $M[:, i]$ that has the largest 2-norm.

the following inequalities in class:

$$\mathbb{P}(\mathcal{A} \circ \mathcal{S}_p(D) \in S) \leq (1 - p + pe^{\varepsilon})\mathbb{P}(\mathcal{A} \circ \mathcal{S}_p(D) \in S') + p\delta\,, \tag{1}$$
$$\mathbb{P}(\mathcal{A} \circ \mathcal{S}_p(D) \in S) \geq (1 - p + pe^{-\varepsilon})\mathbb{P}(\mathcal{A} \circ \mathcal{S}_p(D) \in S') - pe^{-\varepsilon}\delta\,. \tag{2}$$

To complete the proof, we need to show that

$$\mathbb{P}(\mathcal{A} \circ \mathcal{S}_p(D) \in S) \geq \frac{1}{1 - p + pe^{\varepsilon}}\Big(\mathbb{P}(\mathcal{A} \circ \mathcal{S}_p(D) \in S') - p\delta\Big)\,. \tag{3}$$

Establish the following:

1. $(1 - p + p/x) \geq 1/(1 - p + px)$ for all $p \in (0, 1)$ and $x > 0$. **Hint**.[3]
2. $1 - p + px \leq x$ for all $p \in (0, 1)$ and $x \geq 1$.
3. Using these two, show that (3) follows from (2).

4 Batch Sizes in Private Optimization: Signal-to-noise ratio (10 Marks)

In this exercise, we will examine the impact of varying the batch size on DP-SGD.

Recall that each step t of DP-SGD involves the private estimate of the sums of the minibatch gradients:

$$\hat{g} = \left(\sum_{i \in B_t} \mathrm{clip}_C\big(\nabla \ell_i(\theta_t)\big)\right) + w_t$$

for $w_t \sim \mathcal{N}(0, \sigma^2 C^2 I)$, where B_t is the batch of (expected) size m, σ is the noise multiplier and C is the clip norm. The noise multiplier $\sigma = \sigma(m/n, T, \varepsilon, \delta)$ is a function of the sub-sampling ratio $p = m/n$, number of iterations T, and the target privacy parameters ε, δ — we saw how to obtain the noise multiplier from these parameters using the `dp-accounting` library. The dataset size n is fixed. For the privacy parameters, $\delta = 1/n^{1.1}$ is a common default, and ε is given to us (in most experiments, we will actually vary ε). Thus, it remains to tune the batch size m and the number of iterations T.

A useful quantity to predict the performance of DP-SGD is the *noise standard deviation for the average gradient* r, which is defined as

$$r := \frac{\sigma}{m}\,.$$

Indeed, the average gradient $\bar{g}$ is given by

$$\bar{g} = \frac{\hat{g}}{m} = \left(\frac{1}{m}\sum_{i \in B_t} \mathrm{clip}_C\big(\nabla \ell_i(\theta_t)\big)\right) + \bar{w}_t\,, \quad \text{for} \quad \bar{w}_t \sim \mathcal{N}\left(0, \frac{\sigma^2 C^2}{m^2}\right)\,.$$

Thus, the variance of noise (per-dimension) added to the average gradient is $\sigma^2 C^2/m^2$. Normalizing this by the (squared) norm C^2 of each per-example gradient and taking the square root gives us the ratio r. This is a related to (the inverse of) the signal-to-noise ratio.

In general, DP-SGD needs a much larger batch size than regular (non-private) SGD to obtain reasonable performance, usually around $10\times$ to $100\times$ larger.

[3] **Hint**: By the arithmetic-geometric mean inequality, we have $x + 1/x \geq 2$ for all $x > 0$.

Context The connection to the signal-to-noise ratio is obtained as follows. The variance of the private mean $\bar{g}$ is $N = \sigma^2 C^2/m^2$ as calculated above; this the contribution of the noise. The amount of "signal" is the norm of the gradients, this is bounded as

$$\left\| \frac{1}{m} \sum_{i \in B_t} \text{clip}_C\big(\nabla \ell_i(\theta_t)\big) \right\|_2^2 \overset{\text{(a)}}{\leq} \frac{1}{m} \sum_{i \in B_t} \big\| \text{clip}_C\big(\nabla \ell_i(\theta_t)\big) \big\|_2^2 \overset{\text{(b)}}{\leq} C^2 \,,$$

where we used (a) Jensen's inequality assuming that $|B_t| = m$, and, (b) that $\text{clip}_C(\cdot)$ returns a vector with ℓ_2-norm at most C. Taking a square root of the ratio of these two quantities gives r.

Exercise Task Your task is to make the some plots using the "RDPAccountant" provided by the `dp-accounting` library. You may refer to code from our lab on this topic.

- Fix the default parameters: dataset size $n = 60000$ (corresponding to MNIST; earlier version had a typo saying $n = 6000$, instead of 60000) and $\delta = 1/n^{1.1}$.
- Set $T = 2400$, which corresponds to 4 epochs at a batch size of 100 — this was the number of updates required for convergence in the non-private training (recall Lab 1). This provides a reasonably accurate initial guess of the number of updates T. Once we fix the batch size, we can go back and change T in order to get the best performance. In practice, once we fix the batch size m, the choice of T is usually dictated by the computational budget available.
- Set $\varepsilon = 2$. Calculate the ratio r as you vary the batch size $m \in [64, 128, \ldots, 4096, 8192]$. Make a plot of r vs. m in log-log scale, keeping all other quantities fixed. You should obtain a hinge-shaped curve, as in Figure 1 of Ponomareva et al. 2023 or Figure 1 below. That is, there is a linear decrease and it saturates after a point.

  Q1 What is the slope of line (in log-log scale) where we observe linear decrease?

  Q2 What is the approximate value of m for which the ratio r saturates?

- Repeat the above for $\varepsilon = 8, 32, 128$.

Practical Guidance for DP-SGD Non-private training typically uses a batch size around 32 or 64 or 128 for image classification models. These plots show that increasing the batch size beyond this point gives significant improvements (linear part of the curve) at a higher computational cost. (Larger batch sizes imply more computational and memory cost per update.)

It levels off beyond a certain point (flat part of the curve), meaning that there is no benefit to increasing the batch size further. To obtain the best performance while avoiding unnecessary computation cost, we choose the batch size around the saturation point.

5 DP-SGD Implementation: Part 1 (45 marks)

(The breakdown is 25 marks for the implementation + 20 marks for running the code.)

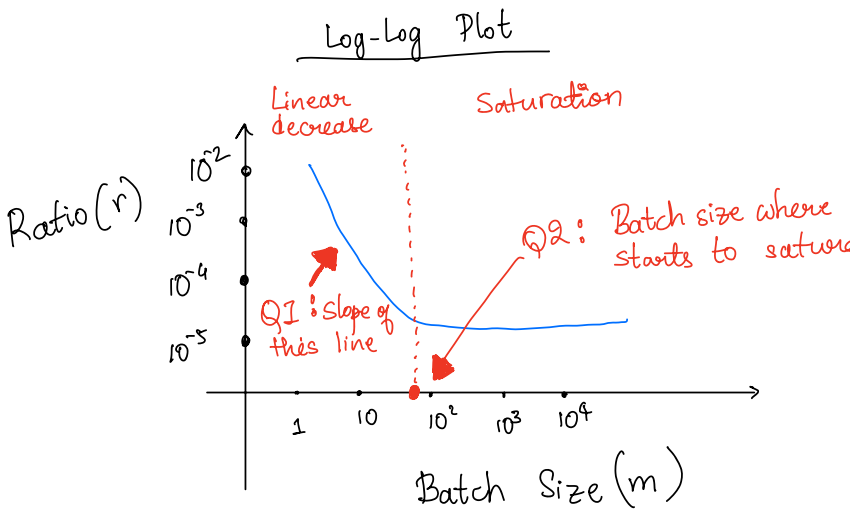

Figure 1: Figure illustrating the expected behavior.

Your task is to implement DP-SGD on the MNIST dataset now. You may wish to use a GPU for this: cloud providers like Google Colab or Kaggle give you free GPUs you could experiment with.[4] Here are some sub-tasks:

1. Write a function using PyTorch's `vmap` functionality to compute per-example gradients, given a batch of data as input. You may use code from the PyTorch tutorial on this topic. Note that `vmap` returns a dictionary of the form {`param_name : param_grad`}, where `param_grad` is of shape $(b, d_1, d_2, \ldots)$ where b is the batch size and $(d_1, d_2, \ldots)$ is the shape of the corresponding parameter tensor.

2. Write a function that takes as input of the output of the previous function and computes the ℓ_2 norm of each per-example gradient. Recall that we compute the ℓ_2 norm of each gradient $g = (g_1, g_2, \ldots, g_k)$ represented as sequence of tensors as

$$\|g\|_2^2 = \sum_{i=1}^{k} \|\mathrm{vec}(g_1)\|_2^2 \,,$$

   where $\mathrm{vec}(g_i)$ flattens a multi-dimensional array into a vector with one dimension. This is the same as stacking all the tensors into one big vector. (Note that the exact order of stacking the tensors does not matter because we compute the ℓ_2 norm, which is the sum of squares of all of its entries.)

3. Write a function for $\mathrm{clip}_C(\cdot)$ that clips each per-example gradient to a given norm C. It is defined mathematically as

$$\mathrm{clip}_C(z) = z \, \min\left\{1, \frac{C}{\|z\|_2 + \nu}\right\} \,,$$

[4]The best way to go about this is to implement everything on your laptop with a small subset of the data (to make sure the code works as intended). Then, move everything to your cloud GPU.

where we add $\nu = 10^{-6}$ in the denominator above for numerical stability. You may use the previous function to compute the ℓ_2 norm of per-example gradients as a subroutine.

4. Write a function for each DP-SGD update which takes in the model parameters, hyperparameters (clip norm C, noise multiplier σ, learning rate η), and returns the updated parameters. It performs the following:

   - Compute the per-example gradients.
   - Clip the per-example gradients to a given norm C.
   - Sum the clipped per-example gradients over the batch of examples.
   - Privatize the gradients with the Gaussian mechanism by adding zero-mean Gaussian noise with standard deviation σC to the summed gradients from the previous step.
   - Divide the above by the batch size to get privatized averaged gradients.
   - Perform a gradient descent step with these noisy graidents using learning rate η.

   You may refer to the pseudocode from Algorithm 1 in Ponomareva et al. for details.

Running the Code Now that you have implemented DP-SGD, let us run it. Here are some common details:

- Use the MNIST dataset. The original dataset has $n = 60000$ examples. We had used a smaller subsample of 6000 for Lab 1 but we will **use the full dataset** for this problem.
- Use a batch size $m = 100$. (Note that Problem 4 suggests that we should use larger batch sizes for good performance; we will come back to it later in Problem 6.)
- Take the number of iterations to be $T = 2400$.
- Find the noise multiplier using the RDPAccountant of the `dp-accounting` library to achieve $(\varepsilon = 8, \delta = n^{-1.1})$-DP.
- Fix the clip norm to be $C = 1$. (It is usually sufficient to fix the clip norm and tune the learning rate: the performance does not vary too much as long as the product ηC is a constant.)
- You will have to tune the learning rate η next. We will use grid search for this. Find a set of 4 different values of the learning rate on a logarithmic grid (e.g. $0.01, 0.1, 1, 10$) such that the best learning rate (in terms of test accuracy) is not at the end points of the range. For instance, if you run grid search over $0.01, 0.1, 1, 10$ and the best value you obtained is 10, then you must expand the range to include 100. **Note**: In practice, we will tune the algorithm on a validation set and report the final performance on the test set. We are taking a few liberties for this assignment to convey the key ideas without extra work.

5. Report the test accuracy for each learning rate you tried.

6 DP-SGD: Part 2 (25 marks)

Repeat Problem 5 for a larger batch size which you find using the heuristic of Problem 4.

Note Note that per-examples gradients have a high memory cost. Therefore, your machine (or your GPU, if you are using one) might run out of memory. In that case, you need to implement *gradient accumulation*. Note that we only need to maintain the sum of the clipped gradients. So, we can break up a batch of 4096 into 8 batches of 512 each. We can then run vmap with a batch size of 512 (or whatever value works), and compute the sum of the clipped gradients. We repeat that for each of the 8 batches *without* running the SGD update. In all, this gives us the sum of the clipped gradients for 4096 items while only ever running vmap with 512 items.

7 Extra Credit: Privacy-Utility Tradeoffs for DP-SGD (25 marks)

Repeat Problems 5 and 6 for $\varepsilon \in [2, 8, 32]$. Plot the *privacy-utility tradeoff* — that is, plot ε on the x-axis and the best test accuracy over all the learning rates you tried. Which batch size gives better performance: a fixed value of 100 or is it our heuristic?