CSCI-4968: ML and Optimization Spring 2023

Differentially Private Stochastic Gradient Descent

Julian Matthews, John Foster, Jody Sunray *April 7th*

1 Overview

1.1 Stochastic gradient descent

Gradient descent is a popular optimization algorithm at the core of most neural networks. It is used to find the minimum cost of an objective function $J(\theta)$ which is usually defined as the difference between the predicted and actual output. The algorithm works by taking a step in the opposite direction of the gradient and updating the model parameters accordingly.

Stochastic gradient descent (SGD) is a variation of gradient descent where a parameter update is performed for each training point x_i.

Definition 1.1 (Stochastic gradient descent)**.** *The update step for stochastic gradient descent is defined as follows:*

$$\theta_{i+1} = \theta_i - \eta \nabla_\theta J(\theta_i; x^{(i)}, y^{(i)}),$$

where J is the model objective, $x^{(i)}$ is a training data point, $y^{(i)}$ is a training label, and η is the step size we will take in each update.

The main contrast between standard gradient descent and SGD is when updating the parameters, SGD uses a small subset (or minibatch) of the data instead of using the whole dataset. This makes SGD much faster and more efficient than gradient descent, especially for large data sets.

1.2 Differential privacy

Differential privacy ensures that no single data point significantly impacts a learned model. In other words, the output of a model trained with and without a particular data point should be the same. The intuition behind differential privacy is that if we added information about an individual to a model, we would not want that individual's personal data to be compromised. Thus, we add just enough noise such that the privacy of the data is preserved.

Definition 1.2 (Pure differential privacy)**.** *Let D and D' be two datasets that differ by one element. We say that a mechanism $\mathcal{M}$ is ε-differentially private if the probability that a particular set of outcomes S is observed never differs by more than $\exp(\varepsilon)$ between D and D'. In other words,*

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(\varepsilon) \Pr[\mathcal{M}(D') \in S].$$

Averaging elements in a dataset is a simple yet illustrative example of a mechanism. If we remove one of the elements from the data set and the average changes significantly, we would say that this mechanism is not differentially private.

We often relax this definition of differential privacy to include a failure probability term, which is significantly smaller than the probability of any given outcome.

Definition 1.3 (Approximate differential privacy)**.** *Let δ be a failure probability for the differential privacy definition, such that with probability δ, we get no privacy guarantee. Extending the original definition we have*

$$\Pr[\mathcal{M}(D) \in S] \leq \exp(\varepsilon) \Pr[\mathcal{M}(D') \in S] + \delta,$$

where δ is very small. We say that $\mathcal{M}$ is (ε, δ)-differentially private.

1.3 Differentially private stochastic gradient descent

Differentially private stochastic gradient descent (DP-SGD) is an extension of SGD that incorporates differential privacy. The intuition behind DP-SGD is it ensures that model parameters do not have too much information about the data. To achieve this, we clip the gradients and add noise to each gradient update. This places a bound on the amount of privacy lost per gradient update.

2 Gaussian Mechanism

2.1 Global and local sensitivity

Broadly speaking, sensitivity is the maximum amount by which the output of a function can change when the input data is changed by one entry, such as by removing a single data point. The sensitivity of a function determines the amount of noise that needs to be added to ensure it is ε-differentially private. There are two types: global sensitivity and local sensitivity.

Global sensitivity considers the maximum difference between any two neighboring datasets. Thus, this measure of sensitivity is independent of the particular dataset we are querying.

Definition 2.1 (Global sensitivity)**.** *Given a query function f, the global sensitivity of f is the maximum difference in output, considering all possible data sets that differ by one entry. In other words,*

$$\Delta f_{\mathrm{GS}} = \max_{D_1, D_2} ||f(D_1) - f(D_2)||,$$

where D_1 and D_2 are any two data sets that differ by one element.

On the other hand, local sensitivity refers the maximum distance between two neighboring datasets when one of them is fixed.

Definition 2.2 (Local sensitivity)**.** *Given a query function f and a known dataset D_1, the local sensitivity of f is the maximum difference in output if D_1 is changed by one entry. In other words,*

$$\Delta f_{\mathrm{LS}} = \max_{D_2} ||f(D_1) - f(D_2)||,$$

where D_1 is some known dataset and D_2 is a dataset that differs from D_1 by one element.

Once the sensitivity of f is determined, we can use it to determine how much noise to add and thus make the query differentially private. Two common mechanisms for ensuring differential privacy are the Laplace mechanism and Gaussian mechanism, which add Laplace and Gaussian noise respectively.

2.2 Laplace mechanism

The Laplace mechanism adds just enough random noise, sampled from the Laplace distribution, to satisfy pure differential privacy. It is commonly used for real-valued functions $f : D \to \mathbb{R}$.

Definition 2.3 (Laplace mechanism)**.** *Let $f(D)$ be some query function that takes D as the input data set. To make the function ε-differentially private, we add Laplace noise proportional to the sensitivity of f and the desired privacy loss ε. In other words, the following definition of $F(D)$ is ε-differentially private:*

$$F(D) = f(D) + \text{Lap}\left(\frac{s}{\varepsilon}\right),$$

where s is the sensitivity of f.

2.3 Gaussian mechanism

The Gaussian mechanism is similar to the Laplace mechanism in that it adds noise to the output of a function. However, the Gaussian mechanism only satisfies approximate differential privacy, not pure differential privacy. Thus, for real-valued functions, the Laplace mechanism is preferred.

For vector-valued functions $f : D \to \mathbb{R}^k$, on the other hand, the Gaussian mechanism is preferred. This is because it allows for the use of the ℓ_2 norm to calculate sensitivity, which results in a lower sensitivity than the ℓ_1 norm. A lower sensitivity means less noise needs to be added.

Definition 2.4 (Gaussian mechanism)**.** *Let $f(D)$ be some query function that takes D as the input data set. To make the function (ε, δ)-differentially private, we add Gaussian noise with variance $\sigma^2 = \frac{2s^2 \log(1.25/\delta)}{\varepsilon^2}$. In other words, the following definition of $F(x)$ is (ε, δ)-differentially private:*

$$F(D) = f(D) + \mathcal{N}(\sigma^2),$$

where s is the sensitivity of f.

The next section goes into more detail about the steps of the DP-SGD algorithm.

3 The Algorithm

The goal of SGD is to minimize the loss function $\mathcal{L}(\theta)$ with model parameters θ. On each step of the algorithm, we take a random sampling of data points and compute their gradients. Algorithm 1 modifies regular SGD by post-processing the gradient updates to guarantee that each step is differentially private. The main two modifications to the algorithm are gradient clipping and the addition of noise, which is calibrated to hide their presence.

During gradient clipping, we bound each gradient by some norm denoted by C. This bound controls the sensitivity, or influence, of the individual gradients on the computed per sample average. In particular, if the ℓ_2 norm of a gradient is greater than C, then it gets scaled down to the clipping threshold.

We then add noise to each gradient using the Gaussian mechanism with noise scale $\sigma = \frac{2\log(1.25/\delta)}{\varepsilon}$. The amount of noise added is proportional to the privacy loss of each step and the sensitivity of the gradient.

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $x_1, \ldots, x_N$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.
Initialize θ_0 randomly
for $t \in [T]$ **do**
Take random sample L_t with sampling probability L/N
Compute gradient
For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$
Clip gradient
$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i)/\max\left(1, \frac{||\mathbf{g}_t(x_i)||_2}{C}\right)$
Add noise
$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$
Descent
$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$
end for
Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Algorithm 1 outputs with the final model parameters θ_T in addition to the total privacy cost. This is computed by applying the composition theorems of differential privacy, discussed in the next section.

4 Composition Theorems

As noted in the previous section, each step of DP-SGD has a particular privacy cost given by ε. On every iteration, more of the data is accessed, and as a result there is more privacy loss. This phenomenon is known as the Fundamental Law of Information Recovery.

4.1 Properties of differential privacy

Definition 4.1 (Fundamental Law of Information Recovery)**.** *If we query a data set enough times, we can discover information about specific data points. In other words, the privacy loss increases with the number of queries.*

One important property of differential privacy that follows from this notion is that we can add up privacy budgets when two mechanisms sequentially access the same input data.

Theorem 4.1 (Sequential composition)**.** *Let $\mathcal{M}_1$ be an $(\varepsilon_1, \delta_1)$-differentially private mechanism, and let $\mathcal{M}_2$ be an $(\varepsilon_2, \delta_2)$-differentially private mechanism. Then the combination of $\mathcal{M}_1$ and $\mathcal{M}_2$, given by $\mathcal{M}_{1,2} = (\mathcal{M}_1, \mathcal{M}_2)$, is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differentially private.*

We can actually reduce the privacy loss of a ε-differentially private mechanism by only accessing a fraction q of a data set. This gives rise to the privacy amplification theorem, which guarantees a privacy loss of $q\varepsilon$. Intuitively, this guarantee makes sense since we are only accessing a subset of the data.

In Algorithm 1, we take a random sample with probability $q = L/N$, where L is the group size for each step and N is the size of the input data. Thus, privacy amplification comes into play when computing a bound on the total privacy loss for this algorithm. As the size of the sample increases, so too does the privacy loss. Thus, it is especially beneficial to tune the model's batch size.

Theorem 4.2 (Privacy amplification)**.** *Let $\mathcal{M}$ be a (ε, δ)-differentially private mechanism. If we query a fraction q of a dataset D, the privacy loss reduces by a factor of q and we say that $\mathcal{M}$ is $(q\varepsilon, q\delta)$-differentially private.*

We can combine the sequential composition and privacy amplification theorems to motivate the parallel composition theorem, which is based on splitting a data set into disjoint chunks and then running a ε-differentially private mechanism on each chunk. According to the parallel composition theorem, the total privacy loss is guaranteed to be bound by ε.

Theorem 4.3 (Parallel composition)**.** *Let $\mathcal{M}$ be an (ε, δ)-differentially private mechanism, and let X be a data set that has been split into k disjoint chunks such that $X = x_1 \cup x_2 \cup \cdots \cup x_k$. Then the combination of $\mathcal{M}(x_1), \mathcal{M}(x_2), \cdots, \mathcal{M}(x_k)$ is (ε, δ)-differentially private.*

4.2 Privacy accounting

While these privacy bounds are guaranteed to hold true, they are very loose. There has been a substantial amount of research dedicated to privacy loss composition, and this research has led to the strong composition theorem, which places a much tighter bound on privacy.

Theorem 4.4 (Strong composition)**.** *Let $\mathcal{M}$ be a (ε, δ)-differentially private mechanism that takes a dataset D as input. If we query a fraction q of D over T iterations, we get a $(q\varepsilon\sqrt{T\log(\frac{1}{\delta})}, qT\delta)$ differential privacy guarantee.*

We can actually get an even tighter bound given by the moments accountant privacy guarantee. The mathematical insights behind the moments accountant technique are beyond the scope of our project. However, its guarantee can be used in Algorithm 1 to compute an appropriate overall privacy cost.

Theorem 4.5 (Moments accountant)**.** *Let $\mathcal{M}$ be a (ε, δ)-differentially private mechanism that takes a dataset D as input. If we query a fraction q of D over T iterations, we get a $(q\varepsilon\sqrt{T}), \delta)$ differential privacy guarantee.*

5 Convergence Guarantees

5.1 Neural tangent kernel (NTK) matrix

Definition 5.1 (Loss dynamics of a neural network and NTK matrix definition)**.**

$$\dot{L} = \frac{\partial L}{\partial \mathbf{w}}\dot{\mathbf{w}} = -\frac{\partial L}{\partial \mathbf{w}}\frac{\partial L}{\partial \mathbf{w}}^T = -\frac{\partial L}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \mathbf{w}}\frac{\partial \mathbf{f}}{\partial \mathbf{w}}^T\frac{\partial \mathbf{L}}{\partial \mathbf{f}}^T = -\frac{\partial L}{\partial \mathbf{f}}\mathbf{H}(t)\frac{\partial L}{\partial \mathbf{f}}^T,$$

where the NTK matrix is $\mathbf{H}(t) \coloneqq \frac{\partial \mathbf{f}}{\partial \mathbf{w}}\frac{\partial \mathbf{f}}{\partial \mathbf{w}}^T \in \mathbb{R}^{n\times n}$ and $\mathbf{f}$ is defined as the gradient flow, or the direction of steepest descent of a function.

For any arbitrary neural network given the loss dynamics, we can construct the neural tangent kernel (NTK) matrix. The NTK matrix is a symmetric, positive semi-definite matrix that captures information about the loss function of a neural network. It is defined as the Jacobian of predictions with respect to the weights times itself transpose. This matrix is crucial to analyzing the convergence of DP-SGD.

An important property of the NTK matrix is being positive semi-definite (PSD). This ensures that all eigenvalues are non-negative and thus, the gradient descent updates will always move in a direction that minimizes the objective function.

5.2 Different methods of gradient clipping

The choice of clipping method used in DP-SGD will drastically affect convergence. Some different clipping methods include local, global, layerwise and flat clipping. Each of these clipping methods affect the properties

of the NTK matrix differently, and thus will all modify convergence. In local clipping, whether or not the ith gradient is clipped depends only on that gradient's norm compared to some clipping threshold. This differs from global clipping, which considers all the other gradient norms in the dataset when deciding whether or not to clip the ith gradient. Flat and layerwise clipping involve applying an upper bound limit to the gradients. In flat clipping, the upper bound is applied to the entire gradient vector, while in layerwise clipping, only an individual layer's gradient norm is affected by the upper bound.

5.3 Effect of gradient clipping on convergence and NTK matrix

Per-sample gradient clipping, or local clipping, can break the PSD property of the NTK matrix because it modifies the direction and magnitude of the gradients, which can change the sign of the NTK.

Per-sample clipping with a small clipping threshold value breaks the positive semi-definite property of the NTK matrix as it is non-symmetric and may be non-positive in quadratic form. This is because, with a small clipping threshold, the gradients are clipped more aggressively and may become misaligned with the true gradients, leading to a negative NTK.

Clipping with a large clipping threshold has shown to lead to better convergence. Because clipping with a large threshold is less aggressive, the clipped gradients tend to be more aligned with the true gradients. Most importantly, the positive semi-definite property of the NTK matrix is preserved. Because of this better convergence, a large clipping threshold can also mitigate some of the noise needed to achieve differential privacy.

Researchers have also been looking into adaptive clipping techniques that adjust the clipping threshold at each layer. These techniques preserve NTK positivity and aim to lessen the trade-off of accuracy for privacy from DP-SGD.

NTK matrix properties and convergence. The following table shows the convergence properties given different properties of the NTK matrix. Flat and layerwise clipping use a small clipping threshold. As shown in the table, the clipping methods that preserve NTK as a positive semi-definite matrix lead to better convergence.

Clipping type	Symmetric NTK	PSD NTK	Uniform loss convergence	To 0 loss
No clipping	yes	yes	yes	yes
Flat clipping	no	yes	no	yes
Layerwise clipping	no	no	no	no
Large threshold	yes	yes	yes	yes

5.4 Effect of adding noise on convergence

While not as drastic as the effects of the NTK matrix, the added noise also modifies the convergence of DP-SGD. Because noise does not directly modify the NTK matrix, the only element of convergence that is affected by noise is the speed of convergence, while convergence to 0 and monotone convergence remain unaffected. Making use of techniques such as moments accountant allows us to adjust the amount of noise added to each gradient update dynamically based on a privacy budget. This, in turn, can allow us to improve the convergence of DP-SGD and lessen the effect noise has on convergence, while still providing a low privacy budget.

6 Experimental results

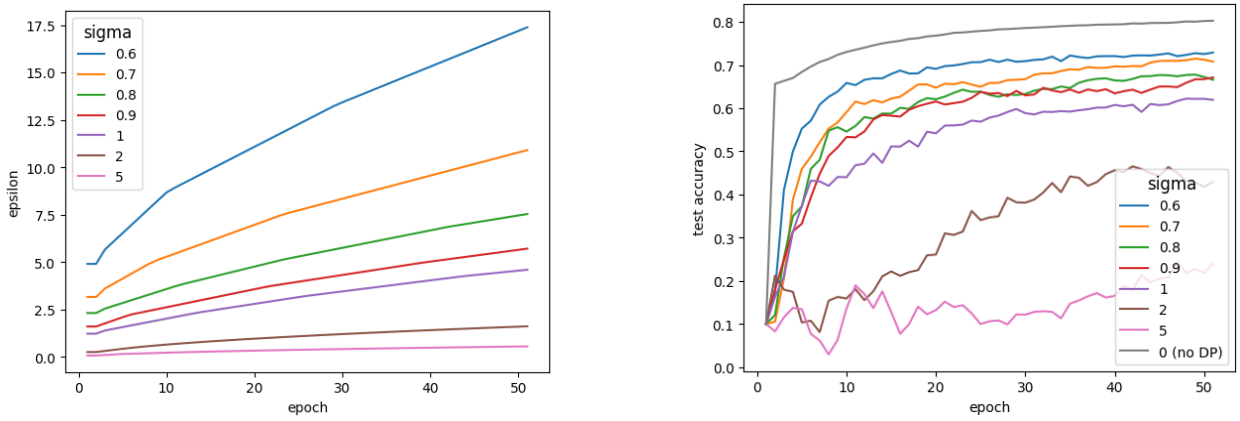
We trained a DP-SGD model using PyTorch as well as from scratch using NumPy. We plotted different values of privacy (ε) as well as test accuracy. We also plotted the trade-off between accuracy and privacy and how different noise scales (σ) affect the overall privacy of the model. In the sections that follow, we will discuss the results of the NumPy implementation which created a logistic regression model for classifying clothing items in the Fashion MNIST data set.

6.1 Effects of noise scale

We implemented DP-SGD in NumPy using a simple logistic regression model with gradient clipping and noise addition. We did privacy accounting using the TensorFlow Privacy Rényi Differential Privacy (RDP) accounting method. RDP is a privacy accounting method that is a generalization of ε-differential privacy, and is similar to the moments accountant (Theorem 4.5) but uses the Rényi moment instead of the Gaussian moment.

To calculate ε, the RDP accounting method uses the moments of the privacy loss distribution. The privacy loss distribution shows how the output of the model differs when run on neighboring input datasets, as mentioned previously as D and D'. The moments of this privacy loss distribution serve to summarize the shape of the distribution, and the RDP accountant uses this information to calculate the privacy lost.

One study achieved 90 percent accuracy for $(0.5, 10^{-5})$-differential privacy and 97 percent accuracy for $(2.0, 10^{-5})$-differential privacy on the digits MNIST dataset using a 1,000 neuron hidden layer cross entropy loss neural network. With our simple logistic regression model on the fashion MNIST dataset, the effect on accuracy for a reasonable privacy loss was substantial. We achieved 80 percent accuracy for no DP, 73 percent accuracy for $(17.0, 10^{-5})$-differential privacy, and 62 percent accuracy for $(4.6, 10^{-5})$-differential privacy. From Figures 1(a) and 1(b), it is clear that the addition of gradient noise from the start of training both reduces the rate at which the model learns about the data and the maximum possible accuracy.



(a) Privacy cost for different noise amounts

(b) Test accuracy for different noise amounts

Figure 1: Privacy cost and test accuracy for different noise amounts, $\delta = 10^{-5}$ (Fashion MNIST)

Figure 2 summarizes the accuracy versus differential privacy trade-off. As shown in this plot, accuracy increases with privacy loss. In other words, if the privacy budget is low, more noise needs to be added and thus the accuracy is lower. It is clear that for a moderate epsilon (<= 10.0) for our simple logistic regression model, accuracy is greatly affected.

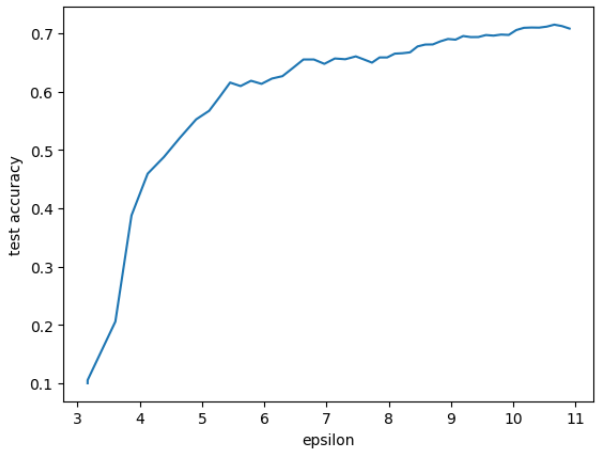
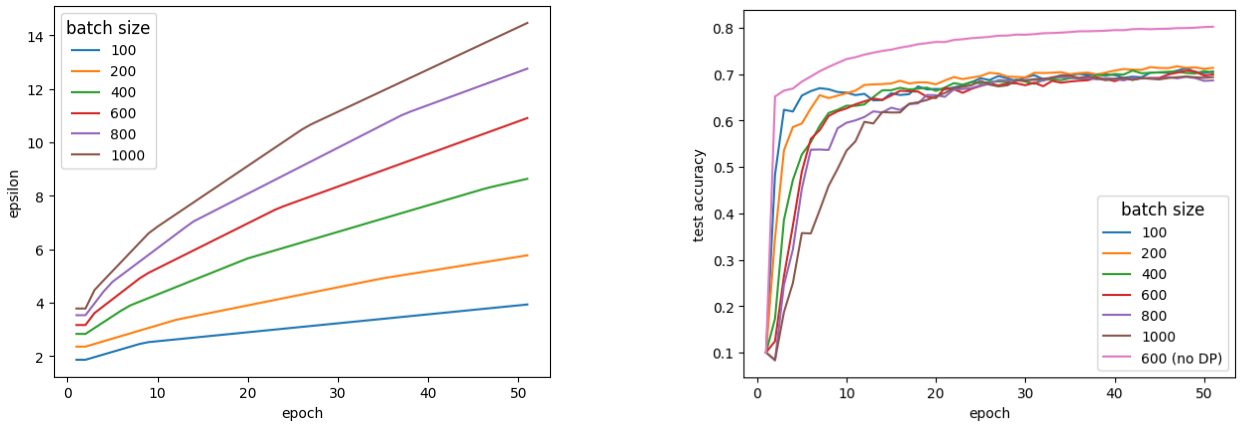


Figure 2: Test accuracy vs. privacy throughout training, $\delta = 10^{-5}$, $\sigma = 0.7$ (Fashion MNIST)

6.2 Effects of batch size



(a) Batch size vs. privacy

(b) Batch size vs. test accuracy

Figure 3: Privacy cost and test accuracy for different batch sizes, $\delta = 10^{-5}$, $\sigma = 0.7$ (Fashion MNIST)

As can be seen in Figure 3(a) where σ was fixed, batch size is an especially important hyperparameter to tune in DP-SGD. The more data the model sees, the higher the privacy loss. However, higher batch size is generally expected to increase accuracy, but Figure 3(b) shows otherwise. This could be due to a higher batch size adding more noise to the gradients, and therefore slowing the initial learning of the model. After around 50 epochs, however, different batch sizes give roughly the same accuracy but vastly different privacy losses (largest $\varepsilon = 14.5$, smallest $\varepsilon = 3.9$).

7 Drawbacks and Further Research

DP-SGD tends to be very slow in comparison to non-DP optimizers. This stems from the computation of per-sample gradient norms, which in practice are very slow and computationally expensive to compute. Efficient implementations of backpropagation such as in TensorFlow and PyTorch only maintain the average of per- sample gradients across a batch by default. Therefore, getting the norm of each per-sample gradient requires significantly more time and memory.

Due to the noise addition of DP-SGD, there is also an accuracy trade-off with most studies finding DP optimizers anywhere from 10 to 15 percent less accurate than non-DP models. New techniques such as micro-batching, adaptive clipping, and even new DP-SGD algorithms such as DP-SGD-JL and DP-Adam-JL are at the forefront of research to speed up DP optimizers.

7.1 Alternative approaches

Micro-batching. TensorFlow's implementation of DP-SGD uses an optimization technique called micro-batching. Micro-batching involves splitting the batch into many "micro-batches" and using those to clip the gradients. The intuition behind micro-batching is the per-sample gradients are first averaged and then clipped. This results in a speedup of n over standard DP-SGD for any micro-batch of size n. However, micro-batching results in more sensitive gradients which in turn require the addition of more noise. Ultimately this could lead to a slight drop in accuracy.

DP-SGD-JL. DP-SGD-JL is a newly designed algorithm that tries to improve on the drawbacks of existing DP methods. Recall the most computationally expensive step of DP-SGD is the computation of per-sample gradient norms. DP-SGDJL uses a technique called JL projections to quickly approximate these gradient norms without having to actually compute them. This, in turn, brings the training time and memory requirements more in line with non-DP methods. This new method has shown to be significantly faster than traditional DP-SGD methods as well as achieve good privacy vs accuracy tradeoff.

8 Next Steps

For our final presentation, we plan to explore how to derive the Laplace and Gaussian mechanisms. In addition, we aim to look more into the different clipping techniques, such as adaptive clipping and automatic clipping, as well as local versus global clipping, to see how these affect privacy and convergence.

We also plan to further look into the impact gaussian noise has on the convergence of DP-optimizers. We plan to test DP-SGD with different values of noise (σ) without any gradient clipping to see what impact if at all noise has on convergence.

To refine our experimental results we also plan to extend the benefits of transfer learning to our DP-SGD model. Transfer learning is the process of training a network on an existing dataset and tuning it on a new dataset. For our case, this means tuning a model on a dataset we wish to be differentially private. In practice, this involves removing the final layer of a neural network that has already been trained and replacing it with a fresh new layer to be trained on the new dataset. In our case, extending transfer learning will allow us to preserve accuracy from a pre-trained non-DP model, while still achieving DP, thus solving the problem of DP-SGD being very inaccurate when trained from scratch.

Additionally, we would like to compare the performance of standard DP-SGD and some of the new algorithmic approaches to DP-SGD such as DP-SGD-JL and DP-Adam-JL. This will likely include an experiment comparing the speed of these new approaches, as well as their memory footprint compared to standard DP-SGD.

References

[1] Abadi M, Chu A, Goodfellow I, et al. Deep Learning with Differential Privacy. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16. Published online 2016. https://doi.org/10.1145/2976749.2978318

[2] Bagdasaryan E, Shmatikov V. Differential Privacy Has Disparate Impact on Model Accuracy. arXiv:190512101 [cs, stat]. Published online October 26, 2019. Accessed April 7, 2023. https://arxiv.org/abs/1905.12101

[3] Bu Z, Gopi S, Kulkarni J, Lee YT, Shen JH, Tantipongpipat U. Fast and Memory Efficient Differentially Private-SGD via JL Projections. arXiv:210203013 [cs]. Published online February 5, 2021. https://arxiv.org/abs/2102.03013

[4] Bu Z, Wang H, Long Q. On the Convergence and Calibration of Deep Learning with Differential Privacy. arXiv:210607830 [cs, stat]. Published online February 1, 2022. Accessed April 7, 2023. https://arxiv.org/abs/2106.07830

[5] Differential Privacy - Programming Differential Privacy. programming-dp.com. Accessed April 7, 2023. https://programming-dp.com/ch3.html

[6] Dwork C, Roth A. The Algorithmic Foundations of Differential Privacy. Foundations and Trends® in Theoretical Computer Science. 2013;9(3-4):211-407. https://doi.org/10.1561/0400000042

[7] Martin S. What Is Transfer Learning? — NVIDIA Blog. The Official NVIDIA Blog. Published February 7, 2019. https://blogs.nvidia.com/blog/2019/02/07/what-is-transfer-learning/

[8] Lundmark M, Dahlman CJ. Differential Privacy and Machine Learning: Calculating Sensitivity with Generated Data Sets Differential Privacy Och Maskininlärning: Beräkning Av Sensitivitet Med Genererade Dataset.; 2017. https://kth.diva-portal.org/smash/get/diva2:1112478/FULLTEXT01.pdf

[9] Rathi M. Deep Learning with Differential Privacy (DP-SGD Explained). mukulrathi.com. https://mukulrathi.com/privacy-preserving-machine-learning/deep-learning-differential-privacy/

[10] Ridout D, Judd K. Convergence properties of gradient descent noise reduction. Physica D: Nonlinear Phenomena. 2002;165(1-2):26-47. https://doi.org/10.1016/s0167-2789(02)00376-7

[11] Ruder S. An Overview of Gradient Descent Optimization Algorithms. https://arxiv.org/pdf/1609.04747.pdf

[12] The Theory of Reconstruction Attacks. differentialprivacy.org. Accessed April 7, 2023. https://differentialprivacy.org/reconstruction-theory/

[13] Wang YX, Balle B, Kasiviswanathan S. Subsampled Rényi Differential Privacy and Analytical Moments Accountant. Journal of Privacy and Confidentiality. 2020;10(2). https://doi.org/10.29012/jpc.723

9 Appendix

9.1 Code repository

Linked here is our project code repository: https://github.com/JohnF1103/ML-OPT-project.

9.2 Problem set

The following problem set has a combination of experimental and non-experimental questions. The first two problems apply knowledge of the composition theorems of differential privacy; the goal is to give students a better understanding of these theorems by deriving their guarantees.

There are two coding problems in this set. The first asks students to use the Laplace mechanism to make a differentially private averaging query. This exercise is rather straightforward but takes students through the steps required to add noise to an existing mechanism, including the calculation of sensitivity.

The second coding example uses a much larger data set, asking students to create a differentially private logistic regression model for the handwritten digits MNIST data set. The goal of this exercise is to understand the impact of the clipping bound on a model.

Lastly, we have a problem that asks students to calculate the privacy loss and failure probability bounds using various composition theorems, given a particular per-step ε and δ. Through this exercise, students will be able to mathematically compare the bounds guaranteed by each theorem.

1. Let $\mathcal{M}_1$ be an ε_1-differentially private mechanism, and let $\mathcal{M}_2$ be an ε_2-differentially private mechanism. Argue that the sequential composition theorem, which states that the combination of $\mathcal{M}_1$ and $\mathcal{M}_2$, given by $\mathcal{M}_{1,2} = (\mathcal{M}_1, \mathcal{M}_2)$, is $\varepsilon_1 + \varepsilon_2$-differentially private. Describe the intuition behind this privacy guarantee.

   Solution. Recall the definition of an ε-differentially private mechanism:

   $$\Pr[\mathcal{M}(x) \in S] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in S].$$

   Let r_1 be some output of $\mathcal{M}_1$ and r_2 be some output of $\mathcal{M}_2$. By definition,

   $$\Pr[\mathcal{M}_1(x) = r_1] \leq \exp(\varepsilon_1) \Pr[\mathcal{M}_1(y) = r_1]$$

   and

   $$\Pr[\mathcal{M}_2(x) = r_2] \leq \exp(\varepsilon_2) \Pr[\mathcal{M}_2(y) = r_2].$$

   Then we have

   $$\begin{aligned}
   \frac{\Pr[\mathcal{M}_{1,2}(x) = (r_1, r_2)]}{\Pr[\mathcal{M}_{1,2}(y) = (r_1, r_2)]} &= \frac{\Pr[\mathcal{M}_1(x) = r_1] \Pr[\mathcal{M}_2(x) = r_2]}{\Pr[\mathcal{M}_1(y) = r_1] \Pr[\mathcal{M}_2(y) = r_2]} \\
   &= \left(\frac{\Pr[\mathcal{M}_1(x) = r_1]}{\Pr[\mathcal{M}_1(y) = r_1]} \right) \left(\frac{\Pr[\mathcal{M}_2(x) = r_2]}{\Pr[\mathcal{M}_2(y) = r_2]} \right) \\
   &= \exp(\varepsilon_1) \exp(\varepsilon_2) \\
   &= \exp(\varepsilon_1 + \varepsilon_2).
   \end{aligned}$$

   This gives us

   $$\Pr[\mathcal{M}_{1,2}(x) = (r_1, r_2)] = \exp(\varepsilon_1 + \varepsilon_2) \Pr[\mathcal{M}_{1,2}(y) = (r_1, r_2)].$$

   Thus, $\mathcal{M}_{1,2}$ is $\varepsilon_1 + \varepsilon_2$-differentially private.

   The intuition behind this conclusion is that each time a data set is queried via particular mechanism, more is learned about that data and thus the privacy loss increases. The sequential composition theorem allows us to bound the privacy loss when more than one differentially private mechanism is applied on the data. This is an upper bound, i.e., the actual total privacy loss could be less.

2. Let $\mathcal{M}$ be an ε-differentially private mechanism, and let X be a data set that has been split into k disjoint chunks such that $X = x_1 \cup x_2 \cup \cdots \cup x_k$. Argue that the parallel composition theorem, which states that the combination of $\mathcal{M}(x_1), \mathcal{M}(x_2), \cdots, \mathcal{M}(x_k)$ is ε-differentially private. Describe the intuition behind this privacy guarantee.

Solution. Let r_i be some output of $\mathcal{M}(x_i)$. By the definition of differential privacy we have

$$\Pr[\mathcal{M}(X) = r] = \exp(\varepsilon) \Pr[\mathcal{M}(Y) = r]$$
$$\Pr[(\mathcal{M}(x_1), \cdots, \mathcal{M}(x_k)) = (r_1, \cdots, r_k)] = \exp(\varepsilon) \Pr[(\mathcal{M}(y_1), \cdots, \mathcal{M}(y_k)) = (r_1, \cdots, r_k)]$$

Using the definition of joint probability we get

$$\prod_i \Pr[\mathcal{M}(x_i) = r_i] = \exp(\varepsilon) \prod_i \Pr[\mathcal{M}(y_i) = r_i]$$

Thus $\prod_i \Pr[\mathcal{M}(x_i) = r_i]$ is ε-differentially private. This bound for the privacy cost is better than that given by the sequential privacy theorem, which would be $k\varepsilon$.

The intuition behind this conclusion is that $\mathcal{M}$ queries each chunk in X one time each, so the total privacy cost will not exceed ε.

3. Suppose we want to compute the average starting salary of a group of Rensselaer Polytechnic Institute computer science graduates. The salaries are given in the data set `D1`. Now suppose we remove one salary from the data set, given by the data sets `D2` through `D6`.

```
import numpy as np

D1 = np.array([130530, 104141, 90385, 91919, 81334])
D2 = np.array([104141, 90385, 91919, 81334])
D3 = np.array([130530, 90385, 91919, 81334])
D4 = np.array([130530, 104141, 91919, 81334])
D5 = np.array([130530, 104141, 90385, 81334])
D6 = np.array([130530, 104141, 90385, 91919])
```

   (a) Write a query `f(D)` that computes the average salary of a given data set.

   Solution. The query should look like the following:

```
def f(D):
    return sum(D) / len(D)
```

   (b) Compute the local sensitivity of `f` given `D1`.

   Solution. The sensitivity can be calculated as follows:

```
sensitivity = abs(f(D1) - f(D2))
```

   (c) Let $\varepsilon = 1.0$ be the privacy loss of `f`. Write a new query K that is ε-differentially private by adding noise to `f`. Note that K is given by the Laplace mechanism for DP functions:

$$K(D) = f(D) + \text{Lap}\left(\frac{s}{\varepsilon}\right),$$

   where s is the sensitivity of f.

   Solution. The implementation for the updated query should look like the following:

```
def K(D, sensitivity, epsilon):
    return f(D) + np.random.laplace(loc = 0.0, scale = sensitivity
   / epsilon)
```

   (d) Compute the average of `D1`, both with and without noise.

Solution. The averages with and without noise, respectively, can be calculated as follows:

```
f_D1 = f(D1)
K_D1 = K(D1, sensitivity, epsilon)
```

(e) Perform tuning of the privacy budget and compare the results. What is the relationship between the privacy loss, amount of noise added, and accuracy of the query? Discuss the differential privacy versus accuracy tradeoff.

Solution. Below is a table comparing different ε-differentially private query outputs:

ε	$f(D1)$
0.1	92788.53
1.0	111458.11
2.0	101603.66
10.0	99400.37
100.0	99682.58
1000.0	99675.37

The higher the privacy budget ε, the more accurate the outcome of a query. This is because the amount of noise that is added to an output is inversely proportional to ε. Thus, if ϵ is large, less noise will be added. Conversely, a lower acceptable privacy loss will result in more noise being added to better hide the data, and as a result the output will be less accurate.

4. Suppose we perform $1,000$ independent runs of a (ε, δ)-differentially private algorithm with $\varepsilon = 0.4$ and $\delta = 10^{-6}$. Compute the privacy budget of this algorithm using (a) the sequential composition theorem and (b) the strong composition theorem. Now suppose that each run samples a batch of 1 percent of the data. Compute new bounds for the privacy loss and failure probability using (c) the strong composition theorem with amplification and (d) the moments accountant. Compare the guarantees of each theorem.

Solution. The guarantees for privacy loss and failure probability are as follows:

(a) $\varepsilon_{\text{total}} = T\varepsilon = 1000 \times 0.4 = \boxed{400.0}$
$\delta_{\text{total}} = T\delta = 1000 \times 10^{-6} = \boxed{0.001}$

(b) $\varepsilon_{\text{total}} = \varepsilon\sqrt{T\log(\frac{1}{\delta})} = 0.4 \times \sqrt{1000 \times \log(\frac{1}{10^{-6}})} = \boxed{30.98}$
$\delta_{\text{total}} = T\delta = 1000 \times 10^{-6} = \boxed{0.001}$

(c) $\varepsilon_{\text{total}} = q\varepsilon\sqrt{T\log(\frac{1}{\delta})} = 0.01 \times 0.4 \times \sqrt{1000 \times \log(\frac{1}{10^{-6}})} = \boxed{0.3098}$
$\delta_{\text{total}} = qT\delta = 0.01 \times 1000 \times 10^{-6} = \boxed{0.00001}$

(d) $\varepsilon_{\text{total}} = q\varepsilon\sqrt{T} = 0.01 \times 0.4 \times \sqrt{1000} = \boxed{0.1265}$
$\delta_{\text{total}} = \delta = \boxed{10^{-6}}$

The sequential composition theorem has a very loose privacy loss guarantee, and this overestimation of ε will require us to add more noise than necessary to counter the privacy loss. The strong composition theorem is tighter, but we can do better if sample a subset of the data (instead of accessing the entire dataset) on each run. By applying the privacy amplification theorem, we get a privacy bound that is 100 times smaller. Still, we can do better with the moments accountant, which has the tightest privacy loss guarantee.

5. Tune hyperparameters for DP-SGD on the standard handwritten digits MNIST data set, to achieve $(< 2.0, 10^{-5})$-differential privacy using 100 epochs. Use a single layer logistic regression model with forward and backwards pass given by the following:

```
# Forward pass
scores = np.dot(X_batch, w.T) + b
probs = 1 / (1 + np.exp(-scores))
loss = -1/batch_size * \
    np.sum(y_batch * np.log(probs) + (1-y_batch) * np.log(1-probs))

# Backward pass
dscores = 1/batch_size * (probs - y_batch)
grad_w = np.dot(dscores.T, X_batch)
grad_b = np.sum(dscores, axis=0)
```

Choose learning rate, C, and σ. Use batch size 600 for $q = 0.01$. Plot test accuracy versus different values of C. How does choice of gradient-clipping parameter C affect the model?

Solution. Solution to be added.