Differential Privacy Meets Weight Decay: A New Optimization Approach for Deep Learning

Guo Songjie

Supervised by **Prof. Yuan CAO**

2 Dec 2024

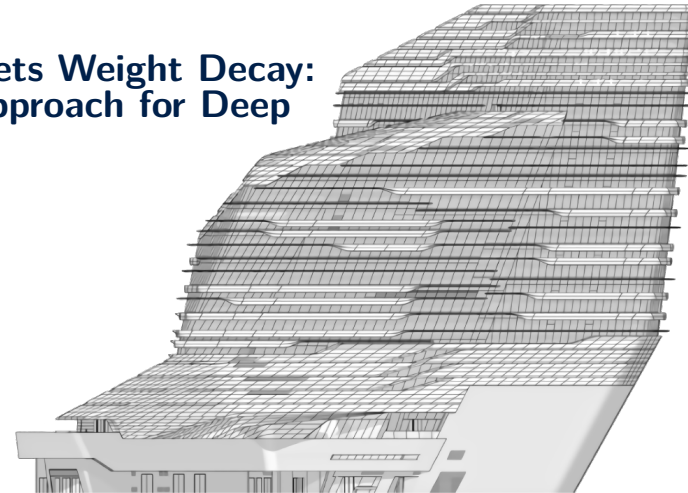


Table of Contents

Background

1 Introduction

- The advances in neural networks are enabled by the availability of large and representative datasets for training.
- These datasets are often **crowdsourced**, and may **contain sensitive information**.
- Require techniques to offer principled privacy guarantees — Differential Privacy.

Adjacent Datasets

Two datasets D and D' are two sets of image-label pairs. They are **adjacent** if one image-label pair is present in one set and absent in the other.

Sex	Blood		HIV?
F	B		Y
M	A		N
M	O		N
M	O		Y
F	A		N
M	B		Y

$\mathcal{D}$

Sex	Blood		HIV?
F	B		Y
M	O		N
M	O		Y
F	A		N
M	B		Y

$\mathcal{D}'$

Figure: Example of two adjacent datasets

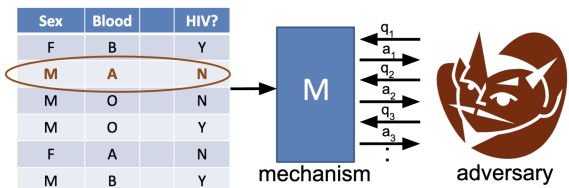
Differential Privacy

1 Introduction

A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with domain $\mathcal{D}$ and range $\mathcal{R}$ satisfies (ε, δ)-**differential privacy** [1] if for any two **adjacent** inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathcal{R}$ it holds that

$$\Pr[\mathcal{M}(d) \in S] \leq e^{\varepsilon} \Pr\left[\mathcal{M}\left(d'\right) \in S\right] + \delta,$$

which allows for the possibility that ε-differential privacy is broken with probability δ.



Source: OpenDP

- Deep neural networks define parameterized functions from inputs to outputs as a combination of multiple layers of basic building blocks. By varying the parameters of these modules we can "train" such a parameterized function with the aim of fitting any given finite set of input/output examples.
- Mathematically, we define a loss function $\mathcal{L}$ that represents the penalty for mismatching the training data. The loss $\mathcal{L}(\theta)$ on parameters θ is the average of the loss over the training examples $\{x_1, \dots x_N\}$, so

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}\left(\theta, x_i\right).$$

- Training is to consistently find θ that yields an acceptably small loss.

Stochastic Gradient Descent

- For complex networks in deep learning, the loss function $\mathcal{L}$ is usually non-convex and difficult to minimize. In practice, the minimization is often done by the mini-batch **stochastic gradient descent (SGD)** algorithm.
- **Algorithm:** at each step, one forms a batch B of random examples and computes an estimation of the gradient $\nabla_\theta \mathcal{L}(\theta)$ as

$$\mathbf{g}_B = \frac{1}{|B|} \sum_{x \in B} \nabla_\theta \mathcal{L}(\theta, x).$$

Then θ is updated following the gradient direction $-\mathbf{g}_B$ towards a local minimum.

Ridge Regression & L2 Regularization

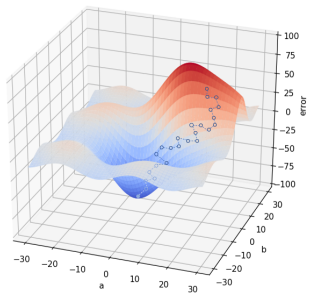
- **Ridge regression** is a method of estimating the coefficients of multiple-regression models when the independent variables are highly correlated. By adding L2 regularization $\frac{\lambda}{2}\|\theta\|_2^2$, the cost function is defined as

$$J = \mathcal{L}(\theta, x) + \frac{\lambda}{2}\|\theta\|_2^2.$$

- The parameter λ is the "weight decay" in deep learning.

Differentially-Private Stochastic Gradient Descent

- Differentially-private stochastic gradient descent (DP-SGD) is a famous algorithm proposed by Abadi et al [2] toward differentially private training of neural networks.



Source: IteractiveChaos

DP-SGD Algorithm

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, noise multiplier σ, group size L, clipping threshold R and weight decay λ.

1. Take a random sample $I^{(t)}$ with probability $q = L/N$
2. Compute gradient: for each $i \in I^{(t)}$, compute the per-sample gradient $\mathbf{g}_i^{(t)}(\mathbf{x}_i) = \nabla_{\boldsymbol{\theta}^{(t)}} \mathcal{L}\left(\boldsymbol{\theta}^{(t)}, \mathbf{x}_i, y_i\right)$
3. Clip gradient: $\overline{\mathbf{g}}_i^{(t)}(\mathbf{x}_i) = \mathbf{g}_i^{(t)}(\mathbf{x}_i) \cdot \min\left(1, \frac{R}{\left\|\mathbf{g}_i^{(t)}(\mathbf{x}_i)\right\|_2}\right)$
4. Add noise: $\tilde{\mathbf{g}}^{(t)} \leftarrow \frac{1}{\left|I^{(t)}\right|}\sum_{i \in I^{(t)}}\left[\overline{\mathbf{g}}_i^{(t)} + \sigma R \cdot N(\mathbf{0}, \mathbf{I})\right]$
5. Descent: $\boldsymbol{\theta}^{(t+1)} = (1 - \eta\lambda)\boldsymbol{\theta}^{(t)} - \eta\tilde{\mathbf{g}}^{(t)}$

Output: $\boldsymbol{\theta}^{(T)}$ and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Problem of DP-SGD with Weight Decay

For the descent equation

$$\boldsymbol{\theta}^{(t+1)} = (1 - \eta\lambda)\boldsymbol{\theta}^{(t)} - \eta\tilde{\mathbf{g}}^{(t)},$$

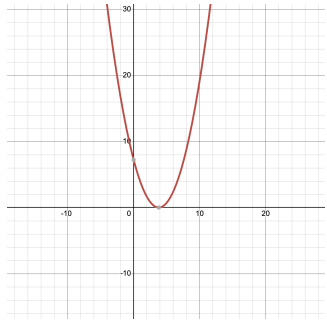
if θ is a scalar and η is small enough, θ^t can reach $-\tilde{\mathbf{g}}^{(t)}/\lambda$, a point that makes the descent equation **always hold**. This may lead to convergence to irrelevant values.

An Extreme Example

For example, consider objective functions

$$\mathcal{L}(\theta) = \frac{1}{2}\left(\theta - \theta^*\right)^2, i = 1, \ldots, n,$$

where $\theta \in R$ is the trainable parameter and θ^* is a constant.



An Extreme Example

- In **ridge regression**,

$$\mathcal{L}(\theta) = \frac{1}{2}(\theta - \theta^*)^2 + \frac{\lambda}{2}\theta^2$$

$$\theta = \arg\min \mathcal{L}(\theta) = \frac{1}{1+\lambda}\theta^*$$

- However in **DP-SGD**, for any $\theta^* > (1 + \frac{1}{\lambda})R$, with $\theta^{(0)} = 0, \sigma = 0$ and small enough η will give iterates $\theta^{(t)}$ satisfying

$$\lim_{t\to\infty} \theta^{(t)} = \frac{R}{\lambda}.$$

- The convergence of DP-SGD may not relate to θ^* at all, but only depends on hyper-parameters R and λ.

An Extreme Example

- Experimentally, θ converges to $\frac{R}{\lambda} = 2$ instead of $\theta^* = 3.8$.

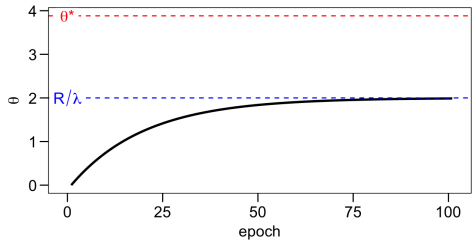


Figure: Experiment result of DP-SGD on $\mathcal{L}(\theta) = (\theta - \theta^*)^2/2$. R and λ are set to 1 and 0.5 respectively.

Table of Contents

2 Method

Our Method

2 Method

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, noise multiplier σ, group size L, clipping threshold R, weight decay λ.

1. Take a random sample $I^{(t)}$ with probability $q = L/N$
2. Compute gradient: for each $i \in I^{(t)}$, compute the per-sample gradient $\mathbf{g}_i^{(t)}(\mathbf{x}_i) = \nabla_{\boldsymbol{\theta}^{(t)}}\{\mathcal{L}\left(\boldsymbol{\theta}^{(t)}, \mathbf{x}_i, y_i\right) + \frac{\lambda}{2}\|\boldsymbol{\theta}^{(t)}\|_2^2\}$
3. Clip gradient: $\overline{\mathbf{g}}_i^{(t)}(\mathbf{x}_i) = \mathbf{g}_i^{(t)}(\mathbf{x}_i) \cdot \min\left(1, \frac{R}{\left\|\mathbf{g}_i^{(t)}(\mathbf{x}_i)\right\|_2}\right)$
4. Add noise: $\tilde{\mathbf{g}}^{(t)} \leftarrow \frac{1}{\left|I^{(t)}\right|}\sum_{i \in I^{(t)}}\left[\overline{\mathbf{g}}_i^{(t)} + \sigma R \cdot N(\mathbf{0}, \mathbf{I})\right]$
5. Descent: $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta\tilde{\mathbf{g}}^{(t)}$

Output: $\boldsymbol{\theta}^{(T)}$ and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Our Method

By using the descent equation

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \tilde{\mathbf{g}}^{(t)},$$

$\theta^{(t)}$ cannot stick to a point as that in DP-SGD, unless it reaches a **local minimum** when

$$\tilde{\mathbf{g}}^{(t)} = 0.$$

Privacy Guarantee

2 Method

Theorem

There exist c_1 and c_2 so that given the sampling probability $q = L/N$ and the number of steps T, for any $\varepsilon < c_1 q^2 T$, our method is (ε, δ)-differentially private for any $\sigma > 0$ if we choose

$$\sigma \geq c_2 \frac{q\sqrt{T \log(1/\delta)}}{\varepsilon}.$$

Our method guarantees (ε, δ)-differentially private.

Table of Contents

3 Results

Evaluation

3 Results

To evaluate our method, we compare the performance of our method with DP-SGD on

A. Synthetic Data
B. Real Data

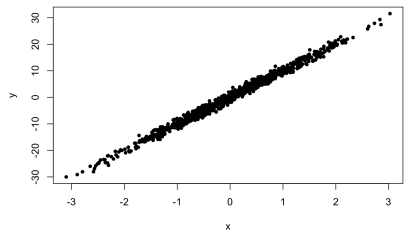
A. Evaluation on Synthetic Data

3 Results

We generate synthetic data by adding noise to linear data. We set the true $\mathbf{w}$ and true $b = 0$, and assume

$$\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$$
$$\varepsilon \sim \mathcal{N}(0, \sigma^2)$$
$$y = \mathbf{X}\mathbf{w} + b + \boldsymbol{\varepsilon}$$

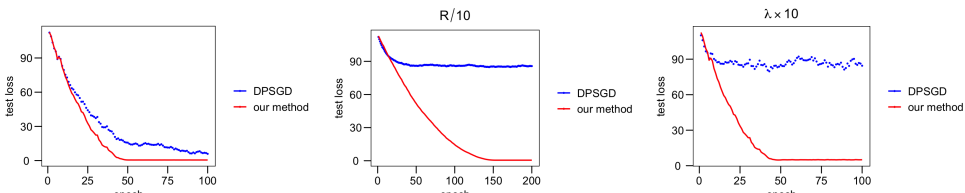
to get 1,000 examples for training and testing.



Simple Linear Regression

3 Results

- $\mathbf{w} = 10$
- When $\|\mathbf{w}\|$ and λ are large enough, and R is small enough, DP-SGD performs poorly, but **our method is almost unaffected**.



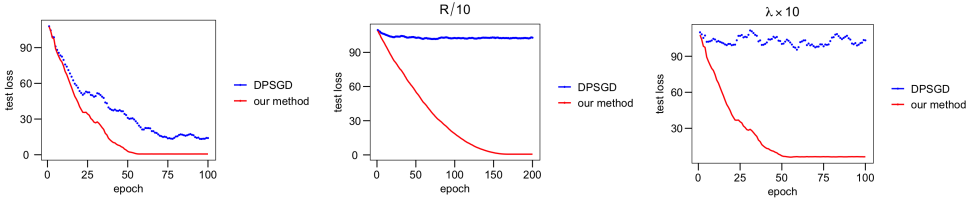
(a) $R = 0.1, \lambda = 0.01$ (b) $R = 0.01, \lambda = 0.01$ (c) $R = 0.1, \lambda = 0.1$

Figure: Results on test loss for different clipping thresholds R and weight decay λ on simulated data of linear regression with $\mathbf{w} = 10, b = 0$. The (a) and (c) graph use a learning rate of 0.03, (b) uses a learning rate of 0.1. In all cases, σ is set to 0.1.

Multiple Linear Regression

3 Results

- $\mathbf{w} = [10, 5]$
- Multiple linear regression follows the same conclusion.



(a) $R = 0.1, \lambda = 0.01$ (b) $R = 0.01, \lambda = 0.01$ (c) $R = 0.1, \lambda = 0.1$

Figure: Results on test loss for different clipping thresholds R and weight decay λ on simulated data of linear regression with $\mathbf{w} = [10, 5], b = 0$. The (a) and (c) graph use a learning rate of 0.03, (b) uses a learning rate of 0.1. In all cases, σ is set to 0.1.

B. Evaluation on Real Data

3 Results

- Opacus is a library that enables training PyTorch models with differential privacy. It is one of the hottest libraries in this area and is based on DP-SGD.
- We use **the same model** as that used in the examples in Opacus and implement our method.
- We compare the performance of our method with the examples in Opacus, including MNIST and CIFAR-10.

MNIST Database

3 Results

- The MNIST database is a large database of handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples.
- **Task:** identify numbers 0-9 from handwriting digits

Figure: Sample images from MNIST test dataset

MNIST: Model

3 Results

- Two 2D convolutional layers, followed by ReLU and max pooling
- Flatten
- Two fully connected layers

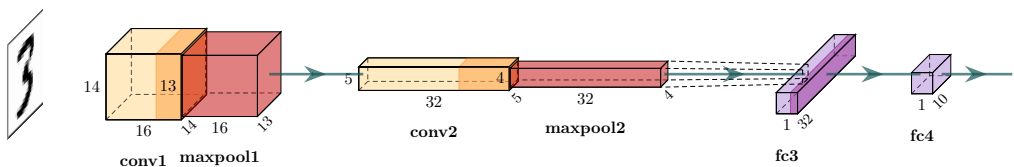


Figure: Model of MNIST

MNIST: Baseline

3 Results

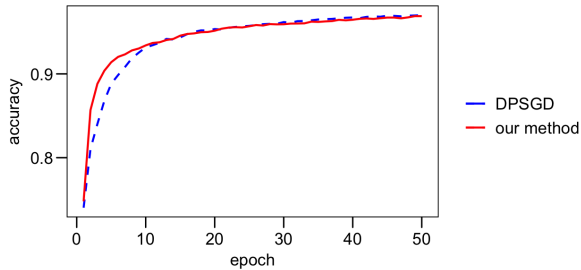


Figure: Result on accuracy for running a baseline on MNIST. With δ set to 10^{-5}, we achieve accuracy **97.01%** and **96.89%** with DP-SGD and our method respectively. Clipping threshold R is set to 3, weight decay λ is set to 10^{-4}, and noise multiplier σ is set to 1.3.

MNIST: Clipping Threshold R

- The performance of both methods deteriorates as clipping threshold R decreases. Among all R values, our method performs better than DP-SGD.

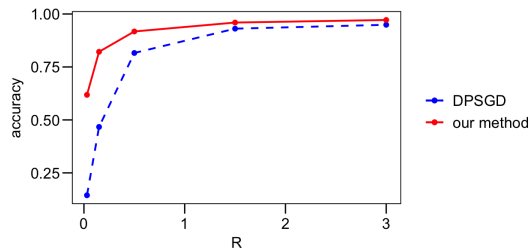
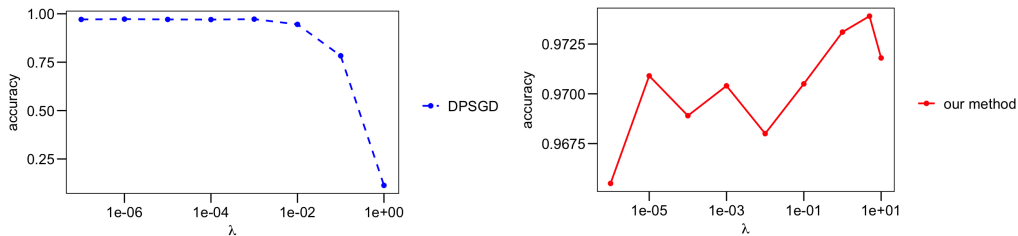


Figure: Result on accuracy for different clipping thresholds R on MNIST. $R \in \{0.03, 0.15, 0.5, 1.5, 3\}$, and others are fixed at baseline values.

MNIST: Weight Decay λ

3 Results

- For $\lambda \geq 10^{-2}$, the accuracy of DP-SGD drops sharply as λ increases, while the accuracy of our method fluctuates at a high level $\sim 97\%$.



(a) DP-SGD (b) our method

Figure: Results on accuracy for different weight decay λ on MNIST using DP-SGD and our method. $\lambda \in [10^{-7}, 0]$ for DP-SGD, $\lambda \in [10^{-6}, 10]$ for our method, and others are fixed at baseline values.

MNIST: Weight Decay λ

3 Results

- Our method achieves the highest accuracy 97.39% at $\lambda = 5$, and DP-SGD achieves accuracy 97.26% at $\lambda = 10^{-6}$.
- The best λ is large for our method because it is added before clipping.

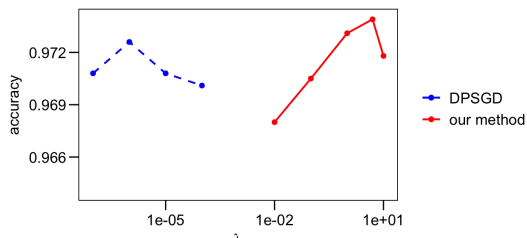


Figure: Results on accuracy for different weight decay λ on MNIST. $\lambda \in [10^{-7}, 10]$, and others are fixed at baseline values.

CIFAR-10 Dataset

3 Results

- The CIFAR-10 dataset is a collection of images commonly used to train machine learning and computer vision algorithms.
- It consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class.
- **Task:** classify the images into 10 classes, like airplane, bird,...

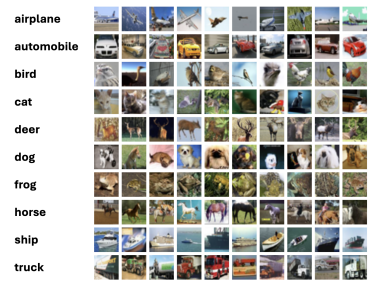


Figure: Sample images from CIFAR-10 dataset

CIFAR-10: Model

- Four 2D convolutional layers, followed by ReLU and average pooling
- Flatten
- One fully connected layer

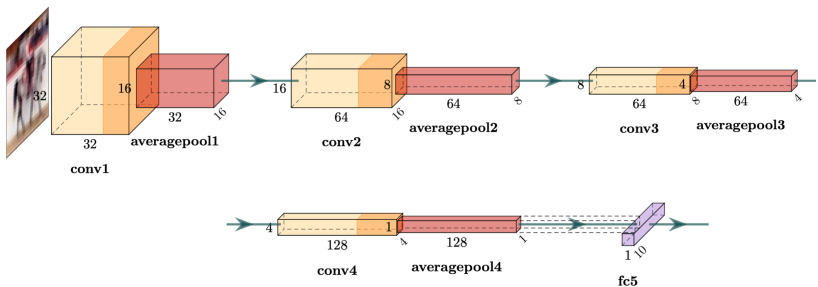


Figure: Model of CIFAR-10

CIFAR-10: Baseline

3 Results

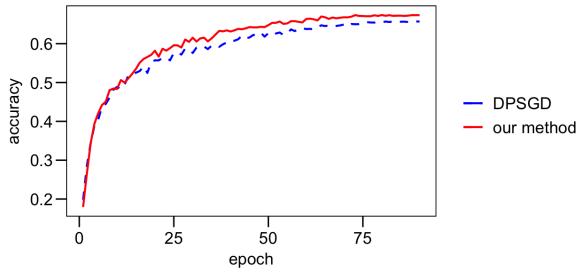


Figure: Result on accuracy for running a baseline on CIFAR-10. With δ set to 10^{-5}, we achieve accuracy **65.75%** and **67.37%** with DP-SGD and our method respectively. Clipping threshold R is set to 10, weight decay λ is set to 10^{-4}, and noise multiplier σ is set to 1.5.

CIFAR-10: Clipping Threshold R

- The performance of both methods deteriorates as clipping threshold R decreases. Among all R values, our method performs sightly better than DP-SGD.

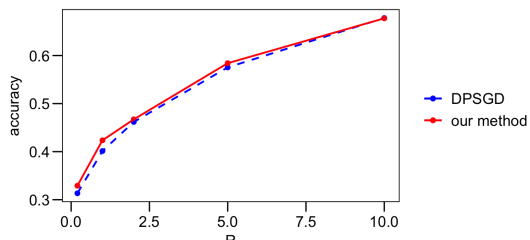
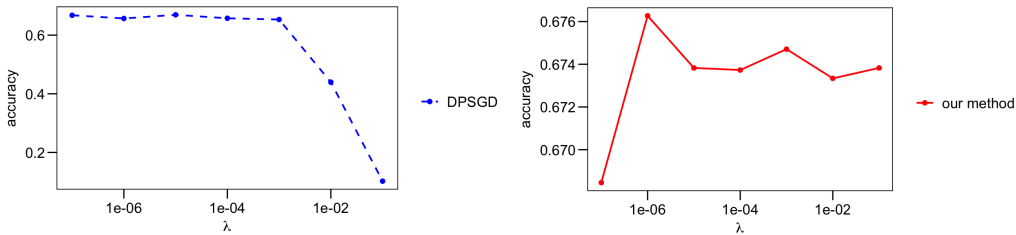


Figure: Result on accuracy for different clipping thresholds R on CIFAR-10. $R \in \{0.2, 1, 2, 5, 10\}$, and others are fixed at baseline values.

CIFAR-10: Weight Decay λ

3 Results

- For $\lambda \geq 10^{-3}$, the accuracy of DP-SGD drops sharply as λ increases, while the accuracy of our method fluctuates at a high level $\sim 67.4\%$.



(a) DP-SGD

(b) our method

Figure: Results on accuracy for different weight decay λ on CIFAR-10 using DP-SGD and our method respectively. $\lambda \in [10^{-7}, 10^{-1}]$ for both DP-SGD and our method, and others are fixed at baseline values.

CIFAR-10: Weight Decay λ

3 Results

- Our method achieves higher accuracy than DP-SGD among all different λ.

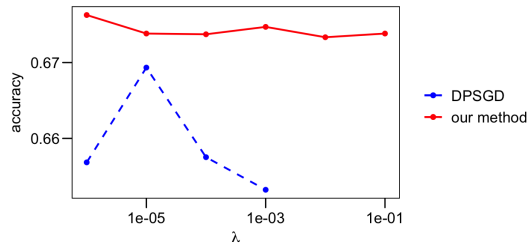


Figure: Results on accuracy for different weight decay λ on CIFAR-10. $\lambda \in [10^{-7}, 10^{-1}]$, and others are fixed at baseline values.

Table of Contents

Conclusion

- We proposed a new optimization approach for the training of deep neural networks with L2 regularization.
- In the experiments for MNIST and CIFAR-10, our method consistently outperforms DP-SGD across different clipping thresholds R.
- Our method maintains robust accuracy across different weight decay values λ, while DP-SGD performs poorly in some cases.
- Further, we would like to consider other classes of deep networks and larger datasets.

Q&A

Thank you for listening!

Table of Contents

References

[1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*, S. Halevi and T. Rabin, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284.

[2] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS'16, ACM, Oct. 2016. DOI: 10.1145/2976749.2978318.

APPENDIX: Discussion on σ

- **Experiment on MNIST:** The performance of both methods deteriorates as noise multiplier σ increases, with almost no difference in both methods.

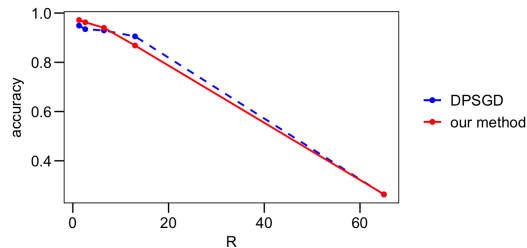


Figure: Result on accuracy for different noise multipliers σ on MNIST. $\sigma \in \{1.3, 2.6, 6.5, 13, 65\}$, and others are fixed at baseline values.

APPENDIX: Discussion on σ

- **Experiment on CIFAR-10:** The same conclusion.

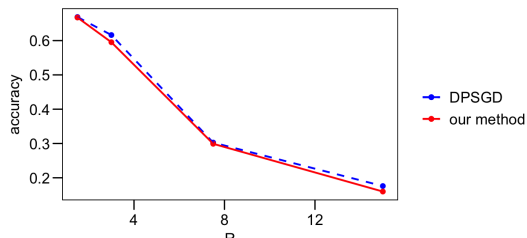


Figure: Result on accuracy for different noise multipliers σ on CIFAR-10. $R \in \{1.5, 3, 7.5, 15\}$, and others are fixed at baseline values.