Université de Lille

Thèse de Doctorat

pour obtenir le grade de :

Docteur de l'Université de Lille

dans la spécialité « Informatique et applications »

par

Antoine Barczewski

Enhancing Differentially private machine learning : Optimizations for Repeated Query scenarios

---

Amélioration de l'apprentissage automatique différentiellement confidentiel : Optimisations pour les scénarios de requêtes répétées

Thèse soutenue le 15 octobre 2025 devant le jury composé de :

Président du jury :

M.	Pierre Bourhis	Directeur de Recherche,	CNRS Cristal

Rapporteur·ices :

Mme	Marianne Clausel	Professeure,	Université de Lorraine
M.	Jean-Michel Loubes	Directeur de Recherche,	Université de Toulouse

Examinateur :

M.	Antoine Boutet	Directeur de Recherche,	Inria Lyon

Directeur de thèse :

M.	Jan Ramon	Directeur de Recherche,	Inria Lille

Inria Université de Lille CRIStAL Centre de Recherche en Informatique, Signal et Automatique de Lille

Abstract

Deep neural networks and other machine learning models have experienced unprecedented growth in recent years. Alongside this enthusiasm, there has been an increasing and well-founded concern about the privacy of the vast amounts of data required to train these models. The combination of these two factors has been a key driver of interest in privacy-preserving machine learning techniques. Differential Privacy has emerged as the gold standard for measuring privacy. This framework is now applied on a wide range of data-driven tasks, such as machine learning and collaborative analysis, where multiple stakeholders wish to query shared data without exposing their own. The main challenge in this domain lies in balancing privacy guarantees with the utility of the results. Indeed, privacy-preserving techniques often come at the cost of reduced utility.

This thesis focuses on techniques to improve machine learning models and tools for analyzing them, while ensuring a satisfactory level of privacy for the underlying data. First, it introduces an innovative approach to privacy-preserving gradient descent methods by addressing the bias introduced by existing methods. By leveraging properties of gradient regularity rather than clipping the gradient, as it is commonly done in popular methods, our approach effectively reduces bias and the noise added to the gradient. We propose a new algorithm that surpasses the state of the art across various datasets.

Second, the thesis explores techniques for computing privacy-preserving empirical cummulative distribution functions, even in cases where the data is distributed across multiple entities. This study proposes a novel method compatible with different security protocols, offering provable privacy guarantees and an analysis of computational costs. range of applications are explored, and experimental results are presented to validate the utility of these methods.

By analyzing optimization mechanisms and distribution functions, this thesis contributes to the development of more practical and efficient privacy-preserving machine learning and data analysis techniques.

Résumé

Les réseaux de neurones profonds, et autres modèles d'apprentissage automatique, ont connu ses dernières années une croissance sans précedent. Avec cet engouement, est apparue une crainte de plus en plus fondée concernant la confidentialité des masses de données nécessaires à l'entraînement de ces modèles. La combinaison de ces deux facteurs a été un moteur essentiel à l'intérêt porté aux techniques d'apprentissage automatique respectueuses de la vie privée. La *Differential Privacy* s'est imposée comme canon de la mesure de confidentialité. Cette mesure est maintenant intégrée dans un grand nombre d'interactions à la donnée comme l'apprentissage automatique ou l'analyse collaborative, où plusieurs parties prenantes souhaitent intéroger une donnée partagée sans exposer la leur. Le principal défi dans ce domaine est d'arbitrer entre les garanties de confidentialité et l'utilité du résultats. En effet, les techniques permettant de protéger la confidentialité vont généralement se faire au détriment de la précision du résultat.

Cette thèse se concentre sur les techniques permettant d'améliorer l'apprentissage automatique de modèles et les outils d'analyse de ceux-ci, tout en guarrantissant un niveau satisfaisant de confidentialité sur la donnée sous-jacente. Premièrement, elle propose une approche novatrice pour les méthodes de descente de gradient respectueuse de la confidentialité en s'attaquant au biais introduit par les methodes actuelles. En utilisant les propriétés sur la régularité du gradient plutôt que de le tronquer, comme il est d'usage dans les méthodes populaires, notre méthode parvient à limiter le biais et le bruit ajouté au gradient. Nous proposons ainsi un nouvel algorithme qui surpasse l'état de l'art sur des jeux de données variés.

Deuxièmement, la thèse couvre les techniques permettant de calculer des fonctions de répartition empirique respectueuses de la confidentialité, même dans les cas où la donnée est partagée entre plusieurs entités. Cette étude propose une nouvelle méthode compatible avec différents protocoles de sécurité, offrant des garanties de confidentialité démontrables et une analyse des coûts computationnels. De nombreuses applications différentes sont testées experimentalement, dont les résultats prouvent l'utilité de cette méthode.

Par l'analyse des mécanismes d'optimisation et des fonctions de répartition, cette

thèse participe au développement de techniques d'apprentissage automatique et d'analyse respectueuses de la vie privée, plus pratiques et efficaces.

Remerciements

First, I would like to warmly thank my PhD advisor, Jan Ramon. You offered me the opportunity to work with great freedom on topics I deeply care about, while consistently guiding me toward more rigor in my thinking and in the way I share ideas. I am sincerely grateful for your trust, your support, and everything I have learned with you.

I also wish to thank all the people at Inria Lille, and in particular the members of the MAGNET team. It has been a privilege to work in such a stimulating and collaborative environment. I would like to express my gratitude to Marc Tomassi, for welcoming me into this wonderful group.

Je tiens maintenant à remercier mes amis : Arianne, Alexandre, Alexandre, Antoine, Arnaud, Benjamin, Camille, Charles, Chloé, David, Edouard, Enguerrand, Eric, Felix, Florian, Francois, Gaspard, Guillaume, Hugo, Jules, Lorraine, Louis, Mathilde, Maxime, Nicolas, Océane, Olga, Pauline, Philippe, Pierre, Pierre-Henri, Quentin, Renan, Romain, Romain, Sami, Simon, Soline, Thomas, Thomas, Valentine, Victoire. Vous êtes la famille que j'ai choisi. Je profite de ces lignes pour vous dire que vous pouvez compter sur moi à tout moment.

Je voudrais remercier mes parents, qui m'ont toujours soutenu quelles que soient les passions que j'ai souhaité poursuivre. Je vous remercie pour cette confiance et pour votre amour. Je remercie également mes beaux-parents qui m'ont accueilli sans réserve et qui me soutiennent avec une grande bienveillance.

À mes enfants, Suzanne, Madeleine et Jean. Je vous aime. Je serai à vos côtés pour la vie.

Enfin, à Marie. À toi vont toutes mes pensées. Tu es l'origine et l'horizon de chacune de mes actions. Mon amour grandit à mesure que le temps passe avec toi, le bonheur est devenu facile.

Contents

Chapter 1

Introduction

In recent years, the intersection of machine learning and privacy has received significant attention, driven by the increasing deployment of machine learning algorithms and models across various domains. These algorithms and models have become the backbone of many industrial products and scientific discoveries, ranging from movie recommendations and fraud detection to robotics and medical research. However, the success of these models relies heavily on vast amounts of data, often containing sensitive information, raising serious concerns about data privacy.

The growing awareness of these risks has led to the emergence of privacy-preserving machine learning, where the goal is to develop models that protect the confidentiality of the underlying training data. Differential privacy (DP), introduced by Dwork et al., has become canonical for quantifying and ensuring privacy in algorithms. Differential privacy ensures that the output of a computation does not significantly differ when any single data point in the dataset is changed, thereby protecting individual privacy.

Differential privacy has emerged as the gold standard for statistical privacy, providing a formal framework to quantify and mitigate the risk of data leakage. Yet, training differentially private models introduce a trade-off: the more privacy is enforced, the less accurate models become. This trade-off is particularly challenging in high-dimensional settings, where existing algorithms struggle to produce models that are both useful and privacy-preserving.

In this thesis, we explore the challenges and potential methods for preserving data privacy in machine learning. We examine the implications of these methods on model utility and discuss how to balance these competing objectives in practice. Additionally, we delve into the role of empirical cumulative distribution functions (ECDF) in privacy-preserving machine learning, highlighting their importance and the challenges associated with computing them in a privacy-preserving manner.

1.1 Overview of Supervised Learning

Supervised learning is a fundamental concept in the field of machine learning, where the goal is to learn a function that maps inputs to outputs based on a set of labeled examples. In this framework, a machine learning model is trained to make predictions or decisions by learning from a dataset that contains both the input data and the corresponding correct output, or label. This learning process is termed "supervised" because the model is guided by the provided labels.

Supervised learning can be broadly divided into two main tasks: classification and regression. In a classification task, the goal is to predict a discrete label or category for a given input *e.g.,* predicting if an email is "spam" or "not spam". On the other hand, regression involves predicting a continuous value *e.g.,* predicting the price of a house based on its features, such as size and location.

A critical aspect of supervised learning is the separation of the available data into training and validation/test sets. The training set is used to train the model *i.e.,* this is where the model learns the relationship between the input data and the corresponding labels. The validation and test sets, on the other hand, are used to evaluate the model's performance on unseen data. The validation set is used several times during training to estimate the model performance typically for hyperparameter tuning, whereas the test set is intended to be used only once to provide a final assessment of the model's performance. The separation between training and validation/test sets ensures that the model's performance is not overly optimistic and that it generalizes well to new, unseen examples.

Loss Function. At the heart of supervised learning is the concept of a loss function. The loss function quantifies how well the model's predictions match the actual labels. In other words, it measures the "error" of the model. The choice of loss function depends on the type of problem being solved. For instance, in a classification problem, the cross-entropy loss is commonly used, while in regression tasks, the mean squared error is a popular choice. The goal of training is to minimize this loss function, thereby improving the model's performance.

Empirical Risk Minimization (ERM). The process of minimizing the loss function over the training data is known as empirical risk minimization (ERM). In ERM, the model is adjusted throughout the training to reduce the average loss over the training set, which is referred to as the empirical risk. The underlying assumption is that by minimizing the empirical risk, the model will perform well on unseen data, thereby minimizing the true risk or the expected loss on the overall data distribution. ERM, along with careful measures to ensure that the model generalizes well

and avoids what is known as overfitting, serves as a fundamental principle in many supervised learning algorithms.

Stochastic Gradient Descent (SGD). To effectively minimize the loss function during training, and minimize the empirical risk, one of the most widely used optimization techniques is Stochastic Gradient Descent (SGD). Unlike traditional gradient descent [Rumelhart et al., 1986], which computes the gradient of the loss function with respect to the entire training dataset, SGD approximates this by computing the gradient [Lagrange, 1788] for only a small, randomly selected subset of the data, known as a mini-batch. This approach significantly reduces the computational cost and allows the model to update more frequently, leading to faster convergence. While SGD introduces some noise into the optimization process due to the random sampling of mini-batches, this noise can help the model escape local minima and potentially find better solutions in the parameter space. Over time, as the model iteratively updates its parameters to minimize the loss function, it improves its ability to generalize from the training data to new, unseen examples. Variants of SGD, such as Momentum [Polyak, 1964], RMSprop [Tieleman and Hinton, 2012], and Adam [Kingma and Ba, 2014], further enhance its performance by adapting the learning rate or incorporating momentum to accelerate convergence and stabilize the learning process.

Stochastic gradient descent plays a central role in this thesis. Specifically, we will demonstrate how it can be effectively integrated with differential privacy techniques to enable privacy-preserving machine learning.

1.2 Privacy-preserving machine learning

As machine learning models grow more sophisticated and are increasingly applied to sensitive domains such as healthcare, finance, and social networks, the risk of exposing private information has increased. Privacy-preserving machine learning seeks to address these concerns by developing techniques that allow for the extraction of useful insights from data without compromising individual privacy.

Differential Privacy. One of the most influential concepts in this field is differential privacy. Introduced by Dwork et al., differential privacy (definition 3.1.3) provides a mathematically rigorous framework for quantifying and managing privacy risks. At its core, differential privacy ensures that the inclusion or exclusion of any single data point in a dataset has a minimal impact on the outcome of a computation, thereby protecting the privacy of individuals whose data is included. This property is particularly desirable when implementing machine learning models that operate on sensitive datasets.

To achieve differential privacy, several mechanisms have been devised. These mechanisms typically involve the introduction of controlled randomness into the learning process, making it difficult for adversaries to infer information about any individual data point. Commonly used mechanisms on a single value or vector include the Laplace mechanism (algorithm 2), the Exponential mechanism (algorithm 4), and the Gaussian mechanism (algorithm 3), each tailored to different types of data and privacy requirements. These mechanisms are carefully designed to balance the trade-off between privacy and utility, ensuring that while privacy is preserved, the machine learning model remains useful.

Privacy budget accounting. Often denoted as (ε, δ), the privacy budget is a tuple of parameters that quantifies the level of privacy provided by a differentially private algorithm. A smaller ε, for instance, implies stronger privacy guarantees but typically at the cost of reduced model accuracy. Managing the privacy budget effectively is essential, especially when multiple queries or computations are performed on the same dataset. As every query on the dataset reveals additional information, it increases the overall privacy budget.

Techniques such as composition theorems (theorem 3.1.4) and privacy amplification (theorem 3.2.2) are used to track and optimize the use of the privacy budget over the course of a machine learning task.

Differentially Private Stochastic Gradient Descent (DP-SGD). One of the most significant applications of differential privacy in machine learning is the development of Differentially Private Stochastic Gradient Descent (DP-SGD). DP-SGD extends SGD by incorporating differential privacy guarantees, ensuring that the gradients used to update the model do not reveal sensitive information about individual data points. The most popular variant of DP-SGD uses gradient clipping and adding noise before updating the model (algorithm 7), thus preserving privacy while still allowing the model to learn from the data. Finally, privacy accounting techniques help manage the cumulative privacy loss over multiple iterations. By carefully managing the privacy budget, DP-SGD ensures that the model remains both effective and compliant with the desired privacy constraints, striking a balance between model utility and privacy.

1.3 Contributions

This thesis focuses on the optimization and analysis of machine learning models, addressing the challenge of minimizing the privacy budget in settings that involve repeated querying. In optimization tasks, repeated querying is inherent, as gradients are computed iteratively on batches of data. Similarly, the computation of empirical

cumulative distribution functions (ECDFs) requires multiple evaluations to provide meaningful results. While the effects of repeated queries on privacy have been studied extensively, we argue that better solutions can be designed by tailoring approaches to specific applications.

In the case of optimization, this thesis demonstrates that leveraging gradient sensitivity properties can improve privacy-utility trade-offs when models are constrained to a subclass of Lipschitz-bounded functions. We prove that the bias introduced by choosing models from this subclass is less detrimental than the bias induced by gradient modifications in popular optimization methods. For ECDFs, we show that exploiting their structural properties allows for stronger privacy guarantees, even in more complex scenarios such as multi-party computation.

In summary, this thesis highlights how analyzing the structural properties of specific applications can uncover opportunities to achieve stronger privacy guarantees within more practical setups. The overarching question driving this work is as follows:

> *How can the structural properties of specific applications—here, stochastic gradient descent and ECDFs—be leveraged to improve the privacy-utility trade-off and enable broader setups, such as multi-party computation?*

This thesis answers this question positively by introducing two novel algorithms. First, we present Lip-DP-SGD, an optimization algorithm that leverages the constrained Lipschitzness of the gradient. Experimental evaluations across a diverse range of datasets demonstrate that this constraint results in less accuracy degradation and improved runtime performance compared to current state-of-the-art methods.

Second, we propose a new algorithm for privacy-preserving ECDF computation, which ensures privacy guarantees while maintaining the utility of the output. By leveraging ECDF-specific properties, this algorithm is compatible with a variety of security protocols, including function secret sharing, offering enhanced privacy and accuracy.

Through these contributions, the thesis demonstrates that application-specific structural insights can effectively improve the privacy-utility trade-off and enable the deployment of privacy-preserving techniques in a broader range of practical and collaborative scenarios.

1.4 Outline of the Thesis

The first three chapters introduce the theoretical background that is used in the thesis.

- In Chapter 2, we present the empirical risk minimization problem and provide

a formal definition of differentiability, followed by an introduction to stochastic gradient descent. We then discuss neural networks and the optimization challenges they pose. Finally, we define Lipschitzness, a key property that plays a central role in the development of our proposed optimization approach.

- Then, Chapter 3 explores the differentially private formulation of the empirical risk minimization problem. We introduce the concept of differential privacy and the foundational components that enable the design of differentially private algorithms. Finally, we demonstrate how a differentially private version of stochastic gradient descent can be employed to solve the empirical risk minimization problem while preserving privacy.

- In Chapter 4, we define the fundamental security primitives necessary to understand the challenges of multi-party computation. We introduce key security concepts, such as computational indistinguishability, as well as foundational notions in multi-party computation, including common methods for sharing secrets among stakeholders.

The next two chapters of this thesis describe our two contributions. Chapter 5 is dedicated to a novel approach for differentially private stochastic gradient descent, and Chapter 6 studies how a complete ECDF can be displayed collaboratively with strong privacy guarantees.

- Chapter 5 presents a novel approach that mitigates the bias arising from traditional gradient clipping. By leveraging a public upper bound of the Lipschitz value of the current model, we can achieve refined noise level adjustments. We present a new algorithm with improved differential privacy guarantees and a systematic empirical evaluation, showing that our new approach outperforms existing approaches also in practice.

- Chapter 6 proposes strategies to compute differentially private empirical distribution functions. While revealing complete functions is more expensive from the point of view of privacy budget, it may also provide richer and more valuable information to the learner. We prove privacy guarantees and discuss the computational cost, both for a generic strategy fitting any security model and a special-purpose strategy based on secret sharing. We survey a number of applications and present experiments.

Finally, in Chapter 7, we conclude this work by summarizing our contributions as a response to the central question stated in the previous section. Additionally, we outline several promising directions for future research.

Chapter 2

Background on Optimization in Machine Learning

Optimization is a central theme in machine learning, most machine learning tasks can be defined as the problem of finding the best model parameters to minimize (or maximize) an objective function. This section will cover key concepts and techniques related to optimization in machine learning, including Empirical Risk Minimization (ERM), Stochastic Gradient Descent (SGD), feed-forward networks and Lipschitz-continuous functions.

2.1 Empirical Risk Minimization (ERM)

Empirical Risk Minimization (ERM) is a fundamental principle in statistical learning theory and forms the basis for many machine learning algorithms. The goal of ERM is to find a model that minimizes the average loss over a given training dataset. Formally, unless made explicit otherwise, we will denote the space of all possible instances by $\mathcal{Z}$ and the space of all possible datasets by $\mathcal{Z}^*$. Let $Z \in \mathcal{Z}^*$ such that $Z = \{\mathrm{z}_i\}_{i=1}^n$ containing n instances $\mathrm{z}_i = (\mathrm{x}_i, \mathrm{y}_i) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{0, 1\}$ sampled identically and independently (i.i.d.) from an unknown distribution on $\mathcal{Z}$. We are trying to build a model $f_\theta : \mathcal{X} \to \hat{\mathcal{Y}}$ (with $\hat{\mathcal{Y}} \subseteq \mathbb{R}$) parameterized by $\theta \in \Theta$, so it minimizes the expected loss $\mathcal{L}(\theta) = \mathbb{E}_z[\mathcal{L}(\theta; \mathrm{z})]$, where $\mathcal{L}(\theta; \mathrm{z}) = \ell(f_\theta(\mathrm{x}), \mathrm{y})$ is the loss of the model f_θ on data point z. One can approximate $\mathcal{L}(\theta)$ by

$$\hat{R}(\theta; Z) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(\theta; \mathrm{z}_i\right) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\mathrm{x}_i), \mathrm{y}_i),$$

the empirical risk of model f_θ. Empirical Risk Minimization (ERM) then minimizes an objective function $F(\theta, Z)$ which adds to this empirical risk a regularization term $\psi(\theta)$ to find an estimate $\hat{\theta}$ of the model parameters:

Definition 2.1.1 (Empirical Risk Minimization (ERM))**.**

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} F(\theta; Z) := \hat{R}(\theta; Z) + \gamma\psi(\theta) \qquad (\star)$$

where $\gamma \geq 0$ is a trade-off hyperparameter.

ERM provides a framework for learning from data by selecting the model parameters that best fit the observed data. However, since ERM is based on minimizing the loss over a finite number of samples, it is important to ensure that the learned model generalizes well to unseen data.

2.2 Differentiability, Gradient, and Jacobian

Differentiability plays a central role in optimization, particularly in machine learning algorithms that rely on gradient-based methods, such as SGD.

Definition 2.2.1 (Differentiability)**.** *Let $f : \mathbb{R}^p \to \mathbb{R}^k$ be a function. The function f is said to be* differentiable *at a point $x \in \mathbb{R}^p$ if there exists a linear map $A : \mathbb{R}^p \to \mathbb{R}^k$ such that:*

$$\lim_{\|h\| \to 0} \frac{\|f(x+h) - f(x) - A(h)\|}{\|h\|} = 0$$

The map A is called the *differential* of f at x. In the case of real-valued functions on $\mathbb{R}^p$, A corresponds to the gradient of f at x, denoted as $\nabla f(x)$. Intuitively, differentiability means that at each point, the function can be well-approximated by a linear map (i.e., its tangent plane).

Definition 2.2.2 (Gradient)**.** *Let $f : \mathbb{R}^p \to \mathbb{R}$ be a function differentiable at a point $x \in \mathbb{R}^p$, the gradient $\nabla f(x)$ is the vector of partial derivatives of f at x, defined as:*

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right)$$

The gradient points in the direction of the steepest ascent of the function f at the point x. The norm of the gradient $\|\nabla f(x)\|$ indicates how steep the ascent is. For differentiable functions, the gradient provides critical information for optimization algorithms such as SGD, as it is used to update the model parameters to minimize the loss function.

In the context of vector-valued functions $f : \mathbb{R}^p \to \mathbb{R}^k$, the gradient generalizes to the Jacobian matrix.

Definition 2.2.3 (Jacobian Matrix)**.** *Let $f : \mathbb{R}^p \to \mathbb{R}^k$ be a differentiable function. The Jacobian matrix of f at a point x is the matrix of all first-order partial derivatives, given by:*

$$J_f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_p} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_k}{\partial x_1} & \frac{\partial f_k}{\partial x_2} & \cdots & \frac{\partial f_k}{\partial x_p} \end{bmatrix}$$

The Jacobian describes how a multivariate function changes at a given point and is a fundamental tool in understanding more complex relationships between variables, such as those in deep neural networks.

2.3 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is one of the most widely used optimization algorithms in machine learning, particularly for training large-scale models such as deep neural networks. Unlike traditional gradient descent, which computes the gradient of the loss function with respect to the entire dataset, SGD approximates the gradient by computing it using only a single (or a small batch of) training example(s) at each iteration.

SGD operates over T iterations, performing three key steps in each iteration: sampling a batch of data, computing the gradient of the loss with respect to the parameters, and updating the parameters based on this gradient. These steps are detailed in algorithm 1.

SGD's efficiency makes it particularly suitable for training deep neural networks, where the number of parameters can be in the millions or billions. However, the noisy nature of the gradient estimate in SGD can lead to oscillations and slower convergence, which can be mitigated through techniques such as learning rate scheduling [Bengio et al., 2012], momentum [Polyak, 1964], and adaptive learning rates [Duchi et al., 2011].

Algorithm 1 Stochastic Gradient Descent (SGD)

Input: Data set $Z \in \mathcal{Z}^*$, model f_θ, loss function ℓ, hypothesis space $\Theta \subseteq \mathbb{R}^k$, batch size $s \geq 1$, learning rate η.
Initialize $\tilde{\theta}_0$ randomly from Θ
for $t = 1$ **to** T **do**
  $V \leftarrow \emptyset$ ▷ Random sampling
  while $S = \emptyset$ **do**
    for $z \in Z$ **do**
      With probability $s/|Z|$: $V \leftarrow V \cup \{z\}$
  for $i = 1 \dots |V|$ **do**
    $\tilde{g}_{t,i} \leftarrow \nabla_{\tilde{\theta}_t} \ell(f_{\tilde{\theta}}(z_i))$ ▷ Compute the gradient
  $\tilde{g}_t \leftarrow \frac{1}{|V|} \sum_{i=1}^{|V|} \tilde{g}_{t,i}$
  $\tilde{\theta}_{t+1} \leftarrow \tilde{\theta}_t - \eta(t)\tilde{g}_t$ ▷ Update parameters
Output: $\tilde{\theta}_T$.

2.4 Neural Networks

Neural networks are a powerful subclass of machine learning models, inspired by the human brain's structure and function. They are particularly well-suited to solving complex tasks such as image recognition, natural language processing or game playing. At their core, neural networks aim to approximate functions by learning from data, iteratively adjusting internal parameters to minimize a predefined loss function.

Neural networks are typically organized as a stack of layers, where each layer performs a transformation of the data. The input is passed through this stack, layer by layer, with each layer applying a mathematical operation to the output of the previous one. This hierarchical structure allows the network to learn progressively more complex patterns from the data as it moves deeper through the network. Each layer consists of units, also known as neurons, which apply transformations using learned parameters (weights and biases).

Dense Layers (Fully Connected Layers). A dense layer, also known as a fully connected layer, is one of the simplest and most fundamental building blocks of neural networks. In a dense layer, every neuron is connected to every neuron in the previous layer. The output of a dense layer is computed as a weighted sum of the inputs, followed by a non-linear transformation via an activation function. The weight matrix defines the strength of the connections between neurons, and these weights are learned during training to minimize the loss function.

Definition 2.4.1 (Dense Layer)**.** *Let x_k be the input of a dense layer, the output*

x_{k+1} is given by:

$$x_{k+1} = W_k x_k + B_k$$

where W_k is the weight matrix, and B_k is the bias vector.

Convolutional Layers. Convolutional layers are specifically designed for tasks involving grid-like data, such as images. In these layers, the network applies a set of filters (also called kernels) that convolve over the input data, detecting local patterns like edges, textures, or shapes. Each filter moves across the input (e.g., an image) and produces an activation map highlighting where the filter detects certain features.

For the purpose of explanation, we will focus on 2D images and consider the input to the convolutional layer as a 3D tensor $x_k \in \mathbb{R}^{h \times w \times c_{\text{in}}}$, where:

- h is the height of the input,
- w is the width of the input,
- c_{in} is the number of input channels (e.g., for an image, this could represent color channels such as RGB).

The convolution layer consists of c_{out} filters (or kernels), where each filter is a 3D tensor $\mathbf{K}_i \in \mathbb{R}^{K_h \times K_w \times c_{\text{in}}}$ for $i = 1, 2, \ldots, c_{\text{out}}$. Here:

- K_h is the height of each filter,
- K_w is the width of each filter,
- c_{in} is the number of channels in the filter, which matches the input depth.

For each filter $\mathbf{K}_i$, the convolution operation computes a 2D feature map by sliding the filter across the spatial dimensions of the input (height and width) and applying the following operation at each spatial location:

$$(x_k * \mathbf{K}_i)(m, n) = \sum_{h=0}^{K_h-1} \sum_{w=0}^{K_w-1} \sum_{c=0}^{c_{\text{in}}-1} x_k(m+h, n+w, c) \cdot \mathbf{K}_i(h, w, c) \tag{2.4.1}$$

where (m, n) denotes the spatial location on the input. This computes the dot product between the filter $\mathbf{K}_i$ and a subset of the input tensor x_k.

The output of the convolutional layer is a new 3D tensor $\mathbf{Y} \in \mathbb{R}^{h_{\text{out}} \times w_{\text{out}} \times c_{\text{out}}}$, where:

- h_{out} and w_{out} are the height and width of the output feature maps, which depend on the stride s (step size between each operation 2.4.1) and padding p (border extension method) used in the convolution,
- c_{out} is the number of output channels, corresponding to the number of filters.

The spatial dimensions of the output feature maps are given by:

$$h_{\text{out}} = \left\lfloor \frac{h + 2p - K_h}{s} \right\rfloor + 1$$
$$w_{\text{out}} = \left\lfloor \frac{w + 2p - K_w}{s} \right\rfloor + 1$$

Typically, a bias term B_i is added to each feature map, so the final output of the convolution operation for filter i is:

$$x_{k+1,i} = (x_k * \mathbf{K}_i) + B_i.$$

Activation Layers. An activation layer introduces non-linearity into the network, which is critical for the network's ability to model complex, real-world data. Without activation functions, the neural network would behave as a linear model regardless of its depth. Common activation functions include:

- **Tempered Sigmoid**: $\phi_{s,T,o}(x) = \frac{s}{1+e^{-T \cdot x}} - o$, with the special case of the sigmoid function being equal to $\phi_{1,\mathbf{1},0}$
- **ReLU** (Rectified Linear Unit): $\text{ReLU}(x) = \max(0, x)$
- **Tanh**: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

The ReLU function is particularly popular in deep networks due to its simplicity and efficiency in training large models.

Pooling Layers. A pooling layer is a layer in a neural network that performs a down-sampling operation to reduce the spatial dimensions (height and width) of the input feature maps while retaining the most important information. Pooling layers are often used in Convolutional Neural Networks (CNNs) after convolutional layers to reduce the computational complexity and mitigate overfitting. There are two common types of pooling operations: *max pooling* and *average pooling*.

Definition 2.4.2 (Max Pooling)**.** *For a pooling region of size $k_h \times k_w$, the max pooling operation at spatial location (m, n) on channel c is defined as:*

$$x_{k+1}(m, n, c) = \max_{h=0,\dots,k_h-1, w=0,\dots,k_h-1} x_k(m + h, n + w, c)$$

Max pooling selects the maximum value within each pooling region. This operation is applied with a stride s, which determines how far the window moves in each step. Note that this operation is not differentiable, because max is not smooth, but it is subdifferentiable. Specifically, during backpropagation through max pooling, the gradient is assigned to the input element(s) that were the maximum during the forward pass. We define forward and backward pass in the following section.

Definition 2.4.3 (Average Pooling)**.** *For a pooling region of size $k_h \times k_w$, the average pooling operation at spatial location (m, n) on channel c is defined as:*

$$x_{k+1}(m, n, c) = \frac{1}{k_h k_w} \sum_{h=0}^{k_h-1} \sum_{w=0}^{k_w-1} x_k(m + h, n + w, c)$$

Average pooling computes the average of all values within the pooling region.

The output of the pooling layer is a 3D tensor $\mathbf{Y} \in \mathbb{R}^{H_{\text{out}} \times W_{\text{out}} \times C}$, where: - H_{out} and W_{out} are the height and width of the output feature map, computed as:

$$H_{\text{out}} = \left\lfloor \frac{H - k_H}{s} \right\rfloor + 1$$

$$W_{\text{out}} = \left\lfloor \frac{W - k_W}{s} \right\rfloor + 1$$

Normalization Layers. A normalization layer is a layer that standardizes or normalizes the input feature maps across a batch or channel. Several papers [Ioffe and Szegedy, 2015, Wu and He, 2018] have pointed out that regularization can help to improve the performance of stochastic gradient descent.

Making abstraction of some elements specific to image datasets, we can formalize it as follows.

For a vector v, we will denote the dimension of v by $|v|$, i.e., $v \in \mathbb{R}^v$.

If the k-th layer is a normalization layer, then there holds $|x_k| = |x_{k+1}|$ with x_{k+1} the output. Moreover, the structure of the normalization layer defines a partitioning $\Gamma_k = \{\Gamma_{k,1} \dots \Gamma_{k,|G|}\}$ of $[|x_k|]$, i.e., a partitioning of the components of x_k. The components

of x_k and x_{k+1} are then grouped, and we define $x_k^{(k:q)} = (x_{k,j})_{j\in\Gamma_{k,q}}$, i.e., $x_k^{(k:q)}$ is a subvector containing a group of components. Similarly, $x_{k+1}^{(k:q)} = (x_{k+1,j})_{j\in\Gamma_{k,q}}$.

Definition 2.4.4 (Normalization Layers)**.** *The k-th layer performs the following operation:*

$$x_{k+1}^{(k:q)} = f_{\theta_k}^{(k)}(x_k^{(k:q)}) = \frac{1}{\sigma^{(k:q)}}\left(x_k^{(k:q)} - \mu^{(k:q)}\right), \tag{2.4.2}$$

where

$$\begin{aligned}\mu^{(k:q)} &= \frac{1}{|\Gamma_{k,q}|}\sum_{j=1}^{|\Gamma_{k,q}|} x_{k,j},\\ \sigma^{(k:q)} &= \left(\frac{1}{|\Gamma_{k,q}|}\sum_{j=1}^{|\Gamma_{k,q}|}\left(x_{k,j} - \mu^{(k:q)}\right)^2 + \kappa\right)^{1/2},\end{aligned}$$

with κ a small constant.

Various feature normalization methods primarily vary in their definitions of the partition of features $\Gamma_{k,q}$, see Figure 2.4.1.

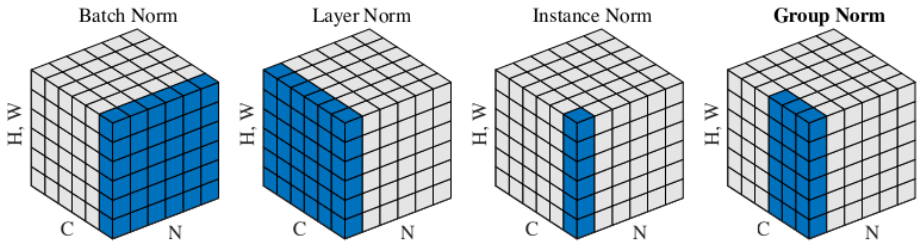


Figure 2.4.1: **Normalization Methods.** Each subplot displays a feature map tensor, where the blue pixels are normalized using the same mean and variance, calculated by aggregating the values of these pixels. Figure from [Wu and He, 2018].

Normalization layers can have learnable parameters γ and β that scale and shift the normalized output, 2.4.2 becomes:

$$x_{k+1}^{(k:q)} = f_{\theta_k}^{(k)}(x_k^{(k:q)}) = \frac{\gamma}{\sigma^{(k:q)}}\left(x_k^{(k:q)} - \mu^{(k:q)}\right) + \beta.$$

2.5 Training Neural Networks

In deep learning, the goal of training a neural network is to minimize a loss function that etimates the error between the model's predictions and the actual target values. To minimize this loss, we use gradient-based optimization algorithms, such as SGD. These algorithms require the computation of the gradient of the loss function with respect to each parameter in the model, indicating how each parameter should be adjusted to reduce the loss.

In practice, when training a deep neural network, we apply the following steps:

- Forward pass: The input data is propagated through the layers to compute the output and the loss.
- Backward pass: The loss is propagated backward to compute the gradients of the loss with respect to each parameter, using the chain rule to break down the derivatives at each layer.
- Parameter update: The gradients are used to update the parameters in a way that minimizes the loss, typically using an optimization algorithm like SGD.

Forward Pass. The forward pass is used to generate predictions from the model and is also the first step in computing the loss during training. The final loss can be computed thanks to the following recursive equation:

$$x_{k+1} = f_{\theta_k}^{(k)}(x_k)$$

where $f_{\theta_k}^{(k)} : \mathbb{R}^{n_k} \to \mathbb{R}^{n_{k+1}}$. $f_{\theta_k}^{(k)}$ is the k-th layer function parameterized by θ_k for $1 \leq k \leq K$. We denote the input of $f_{\theta_k}^{(k)}$ by x_k and its output by x_{k+1}.

Loss function. Consider a feed-forward network f_θ as definied in section 2.6. We define $\mathcal{L}_k(\theta, (x_k, y)) = \ell\left(\left(f_{\theta_K}^{(K)} \circ \ldots \circ f_{\theta_k}^{(k)}\right)(x_k), y\right)$. where ℓ is the loss function and $\mathcal{L}_k$ the model staked with the loss function from the layer k. The loss function ℓ is a mathematical function that quantifies the discrepancy between the predicted output x_{K+1} of the model f_θ and the actual target values y. The loss is designed such that smaller values indicate better agreement between the output of the model and the target. The total loss for the model across the dataset is typically expressed as an average or sum over all data points.

The choice of the loss function depends on the task that is learned. We list below some of the most popular ones:

- For regression tasks:

  - The Mean Square Error (MSE): $\ell(z) = (x_{K+1} - y)^2$

- For classification tasks:
  - The cross-entropy loss: $\ell_\tau(z) = y^\top \log(\textsc{Softmax}(x_{K+1}))/\tau$, with $\textsc{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{c} \exp(x_j)}$, c the number of classes. τ an hyperparameter known as the 'temperature'.
  - The cosine similarity: $\ell(z) = \frac{x_{K+1}^\top y}{\|x_{K+1}\|\|y\|}$, with $\|\cdot\|$ the vector norm as definied in section 2.7
  - The hinge loss: $\ell_m(z) = \sum_{\substack{i=1 \\ i \neq \arg\max y}}^{c} \max(0, m - x_{K+1} \cdot y + x_{K+1,i})/c$, with $x_{K+1,i}$ the ith element of the 1D vector x_{K+1}. m is a hyperparameter known as 'margin'.

Backward Pass. Given a loss function ℓ, which measures the difference between the predicted output x_{K+1} and the true target values y, the backward pass computes the derivatives of the loss with respect to each parameter in the network.

This process utilizes the chain rule [Stewart, 2012] to calculate these gradients layer by layer. Specifically, for $1 \leq k \leq K$

$$\frac{\partial \mathcal{L}}{\partial x_k} = \frac{\partial \ell}{\partial x_{K+1}} \frac{\partial f_{\theta_K}^{(K)}}{\partial x_k}.$$

In practice, we then use the following recursive equation to propagate the error backwards from K to 1:

$$\frac{\partial \mathcal{L}}{\partial x_k} = \frac{\partial \mathcal{L}}{\partial x_{k+1}} \frac{\partial f_{\theta_k}^{(k)}}{\partial x_k}.$$

Efficient Gradient Computation. The chain rule is essential for computing gradients in deep networks because:

- Modular gradient computation: It allows the gradient to be computed layer by layer. The gradient at each layer can be expressed in terms of the gradient at the next layer, making it computationally efficient.
- Reusing intermediate gradients: Once the gradient of the loss with respect to the output of a layer is computed, it can be reused to compute the gradient for the parameters of that layer.
- Computational efficiency through automatic differentiation: Modern deep learning frameworks (e.g., TensorFlow [Abadi et al., 2015], PyTorch [Paszke et al., 2019]) utilize the chain rule to automatically compute the gradients during backpropagation using techniques like reverse-mode automatic differentiation.

2.6 Feed-Forward Networks

An important and easy to analyze class of neural networks are the feed forward networks (FNN). A FNN is a direct acyclic graph where connections between nodes do not form cycles.

Definition 2.6.1. *A FNN $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ is a function which can be expressed as*

$$f_\theta = f^{(K)}_{\theta_K} \circ \ldots \circ f^{(1)}_{\theta_1}$$

Here, $\theta = (\theta_1 \ldots \theta_K)$, $n = n_1$ and $m = n_{K+1}$.

Common layers include fully connected layers, convolutional layers and activation layers. Parameters of the first two correspond to weight and bias matrices, $\theta_k = (W_k, B_k)$, while activation layers have no parameter, $\theta_k = ()$.

Feed-forward networks are universal function approximators, meaning that they can approximate any continuous function to arbitrary precision given enough hidden units and layers.

Multilayer Perceptrons (MLP). MLP is a sub-class of FNN. It a classical form of neural network consisting of multiple dense layers, each followed by an activation function. More formally, an MLP is definied as follows:

Definition 2.6.2 (Multilayer Perceptrons (MLP))**.** *A Multilayer Perceptrons (MLP) f_θ is a FNN such that for all $0 \leq i \leq \left\lfloor \frac{K}{2} \right\rfloor - 1$*

$$f^{(2i+1)}_{\theta_{2i+1}} \text{ is a dense layer with } \theta_{2i+1} = (W_{2i+1}, B_{2i+1})$$
$$f^{(2i)}_{\theta_{2i}} \text{ is an activation layer with } \theta_{2i} = ()$$

MLPs are used for tasks like classification and regression and are suitable for tabular data.

Convolutional Neural Networks (CNN). CNN is a sub-class of FNN. The architecture of a CNN typically consists of alternating convolutional, activation, and pooling layers, followed by one or more fully connected layers.

Definition 2.6.3 (Convolutional Neural Networks (CNN) (Informal))**.** *A general form of a CNN architecture can be represented as:*

$$\text{Input} \to [\text{Convolution} \to \text{Normalization} \to \text{Activation} \to \text{Pooling}]^c \to \text{MLP} \to \text{Output}$$

with c the number of convolutional blocks.

CNNs are widely used in computer vision tasks such as image classification, object detection, and segmentation, as well as in natural language processing for tasks like text classification.

Residual Network (ResNet). A Residual Network (ResNet) [He et al., 2016] is a type of deep neural network that introduces "shortcut" or "skip" connections to improve the training of very deep networks. The key idea behind ResNet is to allow the network to learn residual functions with reference to the layer inputs, rather than learning unreferenced functions. This approach helps mitigate the problem of vanishing gradients (gradients becoming extremely small) and allows for the training of significantly deeper networks.

Definition 2.6.4 (Residual Block Definition)**.** *Let x_k be the input to a residual block and $\mathcal{F}(x_k, \{W_i\})$ represent a series of j convolutional operations (with weights $\{W_i\}$), applied to x_k. In a residual block, the output x_{k+j} of the block is given by:*

$$x_{k+j} = \mathcal{F}(x_k, \{W_i\}) + x_k$$

where: $\mathcal{F}(x_k, \{W_i\})$ is the residual mapping to be learned.

x_k is the original input, which is added (element-wise) to the output of the residual mapping. The addition operation is referred to as a "skip connection" or "shortcut connection."

Definition 2.6.5 (Deep Residual Network Structure)**.** *A ResNet is composed of multiple residual blocks stacked together. The general architecture can be represented as:*

$$x_{k+\sum_{i=1}^{n} j_i} = \mathcal{F}_n(\cdots \mathcal{F}_2(\mathcal{F}_1(x_k)))$$

where each $\mathcal{F}_i$ represents a residual block with j_i operations a skip connection.

Note that, when the input and output dimensions are the same, the shortcut connection can be directly added to the residual mapping without any additional transformation. However, when the dimensions differ (e.g., due to downsampling), a linear projection (often a convolutional layer) is applied to match the dimensions before performing the addition.

ResNets come in different depths, such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, where the numbers indicate the total number of layers in the network.

The use of residual connections allows for the training of very deep networks (hundreds of layers), which helps capture more complex patterns in data. The skip connections help mitigate issues related to vanishing gradients and provide a smoother gradient flow during backpropagation.

2.7 Vector and Matrix Norms

To measure Lipschitzness, a crucial property in this thesis, we need to assess operator norms, which are norms defined on the space of linear operators (matrices) that quantify how much the operator can "stretch" a vector. We begin by introducing the vector norm ℓ_q:

$$\|w\|_q = \left(\sum_{j=1}^{p} |w_j|^q \right)^{1/q}, \quad \text{for all } w \in \mathbb{R}^p, \text{ and } q \geq 0 \tag{2.7.1}$$

Given a matrix $A \in \mathbb{R}^{m \times n}$ and a vector p-norm, the operator norm, also known as induced matrix norm, is defined as:

$$\|A\|_p = \sup_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p},$$

where: $\|x\|_p$ is the p-norm of the vector x, defined as in Equation 2.7.1, and the supremum (sup) is taken over all non-zero vectors $x \in \mathbb{R}^n$.

In other words, the induced matrix norm measures the largest possible factor by which a matrix A can magnify the length of a vector, where the length is measured using the p-norm.

Some commonly used induced matrix norms include:

1. Induced ℓ_1-norm:
$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{m} |A_{ij}|,$$
which is the maximum absolute column sum of the matrix.

2. Induced ℓ_∞-norm:
$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^{n} |A_{ij}|,$$
which is the maximum absolute row sum of the matrix.

3. Induced ℓ_2-norm (Spectral Norm):
$$\|A\|_2 = \sigma_{\max}(A),$$
where $\sigma_{\max}(A)$ is the largest singular value of the matrix A. This norm measures how much the matrix can stretch a vector in the Euclidean space.

Matrix norms induced by vector α- and β-norms. We can generalize the above definition. Suppose we have vector norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ for spaces K^n and K^m respectively; the corresponding operator norm $\|\cdot\|_{\alpha,\beta}$ is

$$\|A\|_{\alpha,\beta} = \sup_{x \neq 0} \frac{\|Ax\|_\beta}{\|x\|_\alpha}$$

In particular, the $\|A\|_p$ defined previously is the special case of $\|A\|_{p,p}$. In the special cases of $\alpha = 2$ and $\beta = \infty$, the induced matrix norms can be computed by

$$\|A\|_{2,\infty} = \max_{1 \leq i \leq m} \|A_{i:}\|_2$$

where $A_{i:}$ is the i-th row of matrix A. In the special cases of $\alpha = 1$ and $\beta = 2$, the induced matrix norms can be computed by

$$\|A\|_{1,2} = \max_{1 \leq j \leq n} \|A_{:j}\|_2$$

where $A_{:j}$ is the j-th column of matrix A. Hence, $\|A\|_{2,\infty}$ and $\|A\|_{1,2}$ are the maximum row and column 2-norm of the matrix, respectively.

Properties. Any operator norm is consistent with the vector norms that induce it, giving

$$\|Ax\|_\beta \leq \|A\|_{\alpha,\beta}\|x\|_\alpha$$

Suppose $\|\cdot\|_{\alpha,\beta}$; $\|\cdot\|_{\beta,\gamma}$; and $\|\cdot\|_{\alpha,\gamma}$ are operator norms induced by the respective pairs of vector norms $(\|\cdot\|_\alpha, \|\cdot\|_\beta)$; $(\|\cdot\|_\beta, \|\cdot\|_\gamma)$; and $(\|\cdot\|_\alpha, \|\cdot\|_\gamma)$. Then,

$$\|AB\|_{\alpha,\gamma} \leq \|A\|_{\beta,\gamma}\|B\|_{\alpha,\beta}$$

2.8 Lipschitz-Continuous Functions

Definition 2.8.1 (Lipschitz function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called Lipschitz continuous if there exists a constant L such that*

$$\forall x, y \in \mathbb{R}^n, \|f(x) - f(y)\|_2 \leq L\|x - y\|_2$$

The smallest value of L that satisfies the inequality above is referred to as the Lipschitz constant of f and is denoted by L^g.

For functions that are locally Lipschitz (i.e., functions that are Lipschitz when restricted to a neighborhood around any given point), the Lipschitz constant can be determined using their differential operator.

Theorem 2.8.1 (Rademacher [Rademacher, 1919])**.** *If $f : \mathbb{R}^n \to \mathbb{R}^m$ is a locally Lipschitz continuous function, then f is differentiable almost everywhere. Moreover, if f is Lipschitz continuous, then*

$$L^g = \sup_{x \in \mathbb{R}^n} \|\nabla f\|_2$$

Lipschitz continuity is crucial in machine learning, especially for optimization and generalization. Constraining a model's Lipschitz constant in deep learning shows stable training and improved generalization. Methods like weight clipping, gradient clipping, and spectral normalization help enforce Lipschitz continuity in neural networks.

Chapter 3

Background on Differential Privacy

Privacy has long been a fundamental concept, recognized across cultures as the right to keep certain aspects of one's personal life away from public exposure. Many countries establish this right in privacy laws, and in some cases, within their constitutions, safeguarding individuals from unauthorized intrusions by governments, corporations, or other entities.

On December 10, 1948, the United Nations General Assembly adopted the Universal Declaration of Human Rights (UDHR). Although the document does not explicitly mention the right to privacy, it is commonly interpreted through Article 12, which states:

> *" No one shall be subjected to arbitrary interference with their privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks. "*
>
> *— [UDHR, 1948].*

While corporations or governments may exploit various techniques to invade privacy for profit or political agendas, individuals often rely on tools like encryption and anonymity measures to keep their digital privacy. This evolving landscape underscores the growing importance of privacy in the digital age.

In the digital age, personal data is collected on a massive scale. From browsing histories and purchase records to social media posts, geolocation data, and health information, the data collected is becoming increasingly detailed and sensitive. Often, individuals are unaware of the extent to which their data is shared or the potential risks involved. This lack of transparency can lead to significant privacy concerns, especially as data plays a central role in various technological applications.

The misuse or exposure of personal data can lead to significant privacy risks, includ-

ing theft of property (e.g., stolen credit card details or passwords), identity fraud (via names, bank details, or biometric data), and exploitation of sensitive information (e.g., health, beliefs, or preferences) for discrimination, blackmail, or public humiliation. Anyone can be affected by these risks, often without their knowledge or consent. The potential consequences of such breaches highlight the urgent need for robust privacy measures.

To mitigate these risks, global attention has been increasingly directed toward privacy regulations. Initiatives like the European Union's General Data Protection Regulation (GDPR), adopted in 2018, aim to protect individuals from privacy-related harms. In addition to general frameworks, sector-specific regulations have been implemented to safeguard sensitive data in areas such as health, education, research, and finance. However, privacy protections often come at a cost to the utility of the data, creating a fundamental trade-off: how to balance privacy with the ability to derive meaningful insights from the data?

The field of privacy research, and particularly differential privacy, seeks to address this challenge. By finding effective ways to protect individual privacy while preserving the utility of data, researchers aim to not only protect individuals but also enable innovative applications that rely on sensitive information. This thesis focuses on differential privacy as a rigorous and promising framework for achieving this balance.

In this chapter, we introduce the concepts of differential privacy [Dwork et al., 2006b]. Section 3.1 begins with an overview of several privacy notions that emerged before differential privacy, highlighting their limitations. We then formally define differential privacy and explore basic mechanisms to enforce it. Additionally, we demonstrate how these mechanisms can be combined to construct more complex differentially private algorithms. In Section 3.2, we focus on training machine learning models with differential privacy.

3.1 Differential Privacy

Differential privacy is a rigorous mathematical framework for ensuring privacy. In its simplest form, it applies to algorithms that analyze datasets and compute statistics such as the mean, variance, or median. An algorithm is considered differentially private if, based on its output, it is virtually impossible to determine whether any particular individual's data is part of the dataset. In other words, the algorithm's output remains nearly identical whether or not a single individual's data is included in the dataset.

This means that the probability of the algorithm producing a certain result is almost the same for two datasets differing by just one individual. This privacy guarantee applies universally, regardless of the uniqueness of any individual's data or the rest of

the dataset. This ensures that even if an individual's details are highly unusual, or if the dataset contains intricate patterns, the privacy guarantee remains the same. As a result, differential privacy provides a strong assurance that sensitive, individual-level information about participants is not exposed.

Before delving into the formal definition of differential privacy, Section 3.1.1 reviews previous attempts to mathematically define privacy, such as anonymization, k-anonymity, and t-closeness. We highlight the limitations of these approaches through practical attacks, which motivated the development of differential privacy. In Section 3.1.2, we formally define differential privacy and discuss its pros and cons. We then state key properties including composition properties. Finally, describe basic mechanisms that ensure differential privacy.

3.1.1 Motivations for a mathematical definition of privacy

Traditional approaches to privacy, such as anonymizing datasets by removing identifiers like names or addresses, are ineffective, as auxiliary information can still enable re-identification.

A better approach is to restrict data access through a trusted system that answers queries. However, ensuring privacy in such systems is challenging, as some queries (or combinations of them) can reveal individual information. Approximate statistics might help, but reconstruction attacks [Dinur and Nissim, 2003] show that releasing too many can expose the entire dataset.

Cryptographic tools, while secure for function outputs, do not prevent the outputs themselves from leaking sensitive information, highlighting the need for robust theoretical frameworks to ensure privacy when releasing statistical data.

3.1.1 (a) Data anonymization

Data anonymization is a method aimed at preserving privacy by removing personally identifiable information (PII) from datasets before sharing or publishing them. The primary goal is to ensure that individuals cannot be directly identified from the data. A common first step in anonymization involves stripping attributes that uniquely identify individuals, such as names, social security numbers, or phone numbers. By eliminating these identifiers, the dataset appears to be unrelated to specific individuals, theoretically protecting their privacy while still allowing for aggregate analysis. We illustrate this method in Figure 3.1.1.

In this example, while an honest user cannot determine that Jean has cancer, a less ethical individual could infer Jean's identity by exploiting features that uniquely identify him. Suppose the anonymized data comes from patients who visited a hospital on a specific day, and each record includes attributes like age, gender, and visit

Name	Age	Gender	Diagnosis
Jean	28	Male	Cancer
Madeleine	37	Female	Asthma
Suzanne	72	Female	Type-2 diabetes
Karol	66	Male	Type-2 diabetes

$\longrightarrow$

Name	Age	Gender	Diagnosis
aldx	28	Male	Cancer
gsDl	37	Female	Asthma
9am0	72	Female	Type-2 diabetes
Z3ec	66	Male	Type-2 diabetes

Figure 3.1.1: Example of data anonymization, names have been hashed.

date. If only one 28-year-old male visited the hospital that day, an attacker could cross-reference this information with the anonymized database to confirm that Jean has cancer. Such uniquely identifiable combinations of features are known as *quasi-identifiers*. When auxiliary datasets are available, attackers can use them to link quasi-identifiers to sensitive information, an attack known as a *linkage attack*. For instance, [Sweeney, 2000] demonstrated that 87% of Americans could be uniquely identified using just their birth date, gender, and ZIP code.

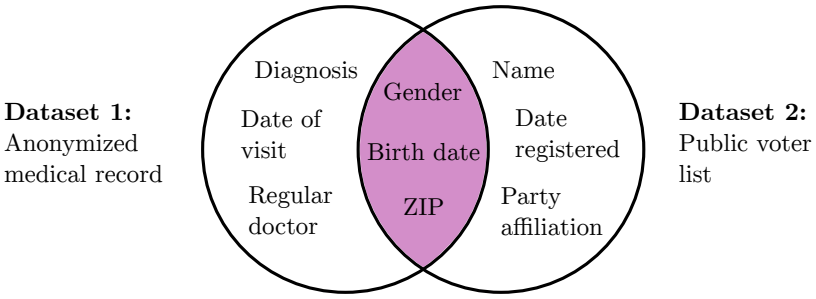


Figure 3.1.2: Linkage Attacks: An anonymized dataset (excluding names) can be linked to a public dataset (including names) using only a few attributes. [Sweeney, 2000] estimated that age, ZIP code, and birth date are sufficient to uniquely identify 87% of Americans. Figure inspired from C. Palamidessi.

Personally Identifiable Information (PII) may seem like the obvious choice for quasi-identifying features. However, even seemingly non-personal attributes can serve as quasi-identifiers. For instance, [Narayanan and Shmatikov, 2007] demonstrated that publicly available movie ratings from IMDB were sufficient to identify a large portion of individuals in the dataset shared by Netflix for their $1M machine learning competition. In this context, data anonymization seems to undermine its own purpose, as achieving true anonymity would require anonymizing all features, effectively rendering the data useless.

3.1.1 (b) k-anonymity and its variants

To address the limitations of basic anonymization, k-anonymity was proposed as a privacy-preserving technique by [Sweeney, 2002]. The key idea behind k-anonymity is to protect individual privacy by ensuring that each record in a dataset is indistinguishable from at least $k-1$ other records with respect to quasi-identifiers.

To achieve k-anonymity, two primary methods are employed: (1) *suppression*, which removes specific values of the quasi-identifiers, and (2) *generalization*, which replaces specific values with broader categories (e.g., replacing a precise age of "28" with an age range of "25-30"). Additionally, dummy records may be added to the dataset to ensure that each record is part of a group of at least k indistinguishable records. We illustrate this procedure in Figure 3.1.3 with the same dataset as in the previous example.

Name	Age	Gender	Diagnosis
Jean	28	Male	Cancer
Madeleine	37	Female	Asthma
Suzanne	72	Female	Type-2 diabetes
Karol	66	Male	Type-2 diabetes

$\longrightarrow$

Name	Age	Gender	Diagnosis
-	20 - 35	-	Cancer
-	35 - 50	-	Asthma
-	65 - 80	-	Type-2 diabetes
-	65 - 80	-	Type-2 diabetes

Figure 3.1.3: Example of 2-anonymity w.r.t. to the quasi-identifiers (age and gender) achieved by suppressing the gender and generalizing the age.

While k-anonymity provides a stronger guarantee of privacy compared to simple anonymization, it has limitations. It does not account for attribute correlations or outliers, and it can still be vulnerable to attacks such as *homogeneity attacks* (Exploit uniform sensitive attributes) or *background knowledge attacks* (Leverage external prior knowledge). For example, Jean's cancer diagnosis can still be inferred if he is the only individual in his age group.

l-Diversity extends k-anonymity by requiring that each group contains at least l distinct sensitive values, reducing the risk of attribute disclosure. However, l-diversity has limitations, such as being ineffective against attacks involving globally skewed distributions or scenarios where the sensitive attribute lacks sufficient diversity (e.g., Jean is alone in his group). To address these issues, t-closeness ensures that the distribution of sensitive attributes in each group closely matches their distribution in the entire dataset, reducing information leakage further. Despite these stronger guarantees, both models face practical challenges. Enforcing t-closeness often heavily reduces data utility and scalability, as finding optimal partitions is NP-hard, making it computationally infeasible for large or complex datasets. While these methods enhance privacy, their utility and efficiency remain significant concerns in real-world applications.

3.1.1 (c) Aggregate statistics through an interface

Aggregate statistics, such as averages, totals, or histograms, can help ensure privacy by summarizing data without revealing information about specific individuals. By reporting only aggregated values, these methods reduce the risk of directly linking sensitive information to any one person. However, this approach has significant limitations. Sophisticated attacks can combine multiple aggregate statistics with external auxiliary information to infer individual data. That includes differencing attacks, where combining queries reveals individual information; membership inference attacks, which determine if a specific person is in a dataset [Shokri et al., 2016]; and reconstruction attacks, which use numerous queries to infer parts of the dataset [Carlini et al., 2021]. Additionally, aggregate statistics often fail to provide formal guarantees of privacy, leaving datasets vulnerable to elaborate forms of information leakage.

3.1.2 Differential Privacy

We have seen that a robust definition of privacy must consider auxiliary knowledge, ensuring resilience against any information an adversary may have, while also controlling cumulative information leakage from multiple analyses to prevent significant breaches. A randomized mechanism addresses auxiliary information by adding noise to obfuscate individual data, even when external knowledge is available. It also limits its cumulative information leakage from repeated queries, preventing attackers from assembling together sensitive details.

A randomized mechanism generates answers based on a carefully chosen probability distribution to ensure privacy while maintaining utility. Achieving this balance requires rigorous definitions of both privacy and utility. In the rest of this section, we formally define the concept of differential privacy and its proporties.

3.1.2 (a) Setting

We recall that we denote the space of all possible instances by $\mathcal{Z}$ and the space of all possible datasets by $\mathcal{Z}^*$. We will denote by $[N] = \{1 \dots N\}$ the set of the N smallest positive integers. Let $\mathcal{A}$ be a randomized algorithm which we define as follows.

Definition 3.1.1 (randomized algorithm)**.** *A randomized algorithm $\mathcal{A}$ is a mapping $\mathcal{A} : \mathcal{Z}^* \to \mathcal{O}$ where $\mathcal{O}$ is a probability space. In other words, for any dataset $Z \in \mathcal{Z}^*, \mathcal{A}(Z)$ is a random variable taking values in $\mathcal{O}$.*

We consider a setting where a trusted curator manages a dataset $Z \in \mathcal{Z}^*$ of n records. One can ask a query f to the dataset Z. For any $z \in Z$, $f(z)$ is the *true answer* of

the query f on Z. Let $\mathcal{A} : \mathcal{Z}^* \to \mathcal{O}$ be a randomized algorithm. For any $z \in Z$, $\mathcal{A}(z)$ is the *reported answer* for the query on Z. We illustrate this setting in Figure 3.1.4

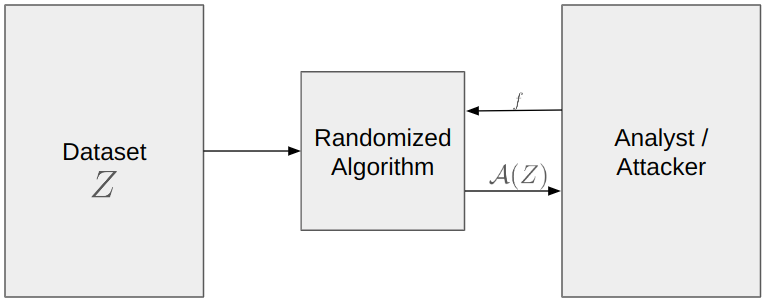


Figure 3.1.4: Setting of a randomized algorithm $\mathcal{A}$ giving a reported answer to the query f.

3.1.2 (b) Definition of Differential Privacy

Informally, differential privacy is defined by ensuring that the presence or absence of an individual's data has minimal impact on what an adversary observes. Specifically, for any two datasets Z_1 and Z_2 differing by a single entry, the output distributions of $\mathcal{A}$ on Z_1 and Z_2 should be nearly "similar".

We see that the definition of differential privacy requires to define what "similar" means and to define properly Z_1 and Z_2.

Definition 3.1.2 (adjacent datasets)**.** *We say two datasets $Z_1, Z_2 \in \mathcal{Z}^*$ are adjacent, denoted $Z_1 \sim Z_2$, if they differ in at most one element. We denote by $\mathcal{Z}^*_\sim$ the space of all pairs of adjacent datasets.*

Definition 3.1.3 (differential privacy [Dwork et al., 2006b])**.** *Let $\varepsilon > 0$ and $\delta > 0$. Let $\mathcal{A} : \mathcal{Z}^* \to \mathcal{O}$ be a randomized algorithm taking as input datasets from $\mathcal{Z}^*$. The algorithm $\mathcal{A}$ is (ε, δ)-differentially private ((ε, δ)-DP) if for every pair of adjacent datasets $(Z_1, Z_2) \in \mathcal{Z}^*_\sim$, and for every subset $S \subseteq \mathcal{O}$ of possible outputs of $\mathcal{A}$,*

$$P(\mathcal{A}(Z_1) \subseteq S) \leq e^\varepsilon P(\mathcal{A}(Z_2) \subseteq S) + \delta. \tag{3.1.1}$$

If $\delta = 0$ we also say that $\mathcal{A}$ is ε-DP.

Differential Privacy (DP) offers significant improvements over earlier privacy definitions by getting rid of reliance on assumptions about the adversary. For instance, in k-anonymity, the value of k depends on the adversary's capabilities, making the guarantee context-dependent. In contrast, DP provides a robust privacy guarantee that is

independent of what the adversary knows (their capability) or aims to achieve (their goal). Additionally, DP quantifies privacy leakage using a precise and well-defined metric, (ε, δ), allowing a clear and measurable bound on potential information exposure.

$(\varepsilon, 0)$-DP guarantees that for every execution of the algorithm $\mathcal{A}(Z)$, the output is nearly equally likely, for a small ε, to occur across all neighboring datasets. This strict guarantee is referred to as *pure* ε-DP.

The parameter δ can be understood as an upper bound on the probability of a catastrophic failure (e.g., the complete dataset leaks). This concept is often referred to as *approximate* differential privacy. Setting $\varepsilon = 0$ is equivalent to requiring that $P(\mathcal{A}(Z_1) = O) - P(\mathcal{A}(Z_2) = O) \leq \delta$. (ε, δ)-indistinguishability can be characterized as "ε-differential privacy holding with probability, over randomness of the algorithm, at least $1 - \delta$," providing a useful interpretation of (ε, δ)-differential privacy.

For meaningful privacy guarantees, δ should be $o(1/n)$. Setting δ on the order of $1/n$ ensures that privacy is guaranteed even if the records of a small number of individuals are released. As for ε, besides the principle the smaller the better, a rule of thumb is to say $\varepsilon = 1$ is typically considered a good privacy guarantee and $\varepsilon = 0.1$ ($e^{\varepsilon} \approx 1.1$) is a very strong guarantee.

Pros and cons. DP offers a structured way to quantify and manage the risks of participating in a dataset, addressing the concern that privacy may already be irreversibly compromised. By ensuring that whatever an adversary learns about an individual could just as easily be inferred from the data of others, DP guarantees that mechanisms cannot leak individual-specific information. This robustness holds regardless of the adversary's auxiliary knowledge, making it a powerful and general framework. Additionally, DP composes well, meaning repeated queries maintain controlled privacy loss. However, DP also has limitations: it provides no protection for information spanning multiple rows, and it cannot guarantee that individuals won't be harmed by the results of an analysis, such as policy decisions based on aggregate statistics. Furthermore, DP is sometimes criticized for being overly strict, as it applies regardless of the actual distribution of the queried dataset, hence some relaxed variants such as the Bayesian differential privacy [Triastcyn and Faltings, 2019]. Despite these drawbacks, DP represents a significant step forward by explicitly estimating the cost of participation in a dataset and providing rigorous privacy guarantees.

3.1.2 (c) Properties

Robustness to auxiliary knowledge. DP guarantees are resilient to any auxiliary knowledge, as it limits the relative advantage an adversary can gain from observing an algorithm's output. This robustness holds even if the adversary knows the entire

dataset except for a single record or has access to all external knowledge sources, both current and future. Moreover, the algorithm $\mathcal{A}$ itself can be public, as only the randomness needs to remain concealed. This transparency allows for open discussions of algorithms and their privacy guarantees.

Robustness to post-processing. The other key aspect contributing to the robustness of differential privacy is its post-processing property. This property ensures that the privacy guarantees of an algorithm remain, even after further processing of its output, provided that the processing does not depend on the original data or the algorithm itself.

Theorem 3.1.1 (Post-Processing [Dwork and Roth, 2013])**.** *Let $\mathcal{A} : \mathcal{Z}^* \to \mathcal{O}$ be (ε, δ)-DP and let $f : \mathcal{O} \to \mathcal{O}'$ be a function that is independent from the data and the randomness of $\mathcal{A}$. Then*

$$f \circ \mathcal{A} : \mathcal{Z}^* \to \mathcal{O}' \text{ is } (\varepsilon, \delta)\text{-DP.}$$

Proof. See proof of Proposition 2.1 in [Dwork and Roth, 2013]. □

The output of a differentially private algorithm remains just as private, regardless of how it is analyzed or used by data users. This guarantee holds independently of the attacker's strategy or computational power.

One can still measure the overall privacy loss even if one composes $K > 0$ algorithms $A_1, \ldots, A_K$ operating on the same dataset. Notably, we account for the possibility that the output of each mechanism may depend on the outputs of the previous ones. More formally, we aim at measuring the privacy loss of

$$\mathcal{A}^{(K)} : Z \to \mathcal{A}_K \circ \cdots \circ \mathcal{A}_k \circ \cdots \circ \mathcal{A}_1 \tag{3.1.2}$$

Theorem 3.1.2 (Simple Composition)**.** *[Dwork and Roth, 2013] Let $\mathcal{A}_1, \ldots, \mathcal{A}_K$ be K independently chosen algorithms where $\mathcal{A}_k$ satisfies $(\varepsilon_k, \delta_k)$-DP. For any dataset Z, let $\mathcal{A}^{(K)}$ be defined as in Equation 3.1.2.*

$$\text{Then } \mathcal{A}^{(K)} \text{ is } (\varepsilon, \delta)\text{-DP with } \varepsilon = \sum_{k=1}^{K} \varepsilon_k \text{ and } \delta = \sum_{k=1}^{K} \delta_k.$$

Proof. See Appendix B.1 in [Dwork and Roth, 2013]. □

The simple composition theorem enables the management of cumulative privacy loss across multiple analyses performed on the same dataset, including intricate multi-step algorithms.

Definition 3.1.4 (Privacy Budget (Informal))**.** *A database is assigned an initial privacy budget. Each time a query is answered, a part of the privacy budget is spent using an (ε, δ)-differentially private mechanism. Once the budget is fully consumed, no further queries can be made.*

In Theorem 3.1.2 the algorithms are picked independently, however, in the case of adaptively chosen algorithms, the overall privacy budget can be curbed.

Theorem 3.1.3 (Advanced Composition)**.** *[Dwork and Roth, 2013] Let $\varepsilon, \delta, \delta' > 0$. If at each iteration $k \in \{1, \ldots, K\}$, the selected algorithm $\mathcal{A}_k$ is (ε, δ)-DP, then $\mathcal{A}^{(K)}$ is $(\varepsilon', K\delta + \delta')$-DP with*

$$\varepsilon' = \sqrt{2K \ln(1/\delta')}\varepsilon + K\varepsilon\left(e^{\varepsilon} - 1\right)$$

Proof. See proof of Theorem 3.20 in [Dwork and Roth, 2013]. □

The advanced composition results for differential privacy are not the tightest, as they provide relatively loose upper bounds on the overall privacy budget. Alternative formulations of (ε, δ)-DP, such as Rényi Differential Privacy (RDP) [Mironov, 2017], offer more refined bounds.

Definition 3.1.5 (Rényi Differential Privacy (RDP))**.** *[Mironov, 2017] Let $\alpha > 1$ denote the Rényi divergence order, and let $\mathcal{A}$ be a randomized algorithm. For any two neighboring datasets Z_1 and Z_2 in $\mathcal{Z}^*_{\sim}$, $\mathcal{A}$ is said to satisfy (α, ε)-Rényi Differential Privacy if:*

$$D_\alpha(\mathcal{A}(Z_1) \| \mathcal{A}(Z_2)) \leq \varepsilon,$$

where $D_\alpha(P\|Q)$ is the Rényi divergence of order α between two distributions P and Q, defined as:

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{z\sim Q}\left[\left(\frac{P(z)}{Q(z)}\right)^{\alpha}\right],$$

where P and Q are probability distributions over the output space of $\mathcal{A}$, and the expectation is taken over the distribution Q.

Theorem 3.1.4 (Composition with RDP)**.** *[Mironov, 2017] Let $\alpha > 1$ and $\mathcal{A}^{(K)}$ be a sequence of (α, ε_k)-Rényi differentially private algorithms. Then the algorithm $\mathcal{A}^{(K)}$ as defined in Equation 3.1.2 satisfies (α, ε)-Rényi differentially private with parameter $\varepsilon = \sum_{k=1}^{K} \varepsilon_k$.*

Proof. See proof of Proposition 1 in [Mironov, 2017]. □

Theorem 3.1.5 (From RDP to (ε, δ)-DP)**.** *[Mironov, 2017] If $\mathcal{A}$ is an (α, ε)-RDP mechanism, it also satisfies $\left(\varepsilon + \frac{\log 1/\delta}{\alpha - 1}, \delta\right)$-differential privacy for any $0 < \delta < 1$.*

Proof. See proof of Proposition 3 in [Mironov, 2017]. □

RDP allows for more precise privacy accounting under composition. When converting its privacy guarantees back to (ε, δ)-DP, it reduces a logarithmic factor in δ and achieve improved constants.

One final property is Group Differential Privacy (Group DP). It extends the standard notion of differential privacy to scenarios where the goal is to hide the participation of a group of individuals rather than a single individual. This is particularly useful when a single participant contributes multiple records to the dataset, or when the data of a group, such as members of a family, is strongly correlated. Group DP ensures that the privacy guarantee applies to the entire group collectively, rather than treating each record independently. Unlike composition, which tracks privacy loss over multiple analyses, Group DP focuses on protecting the presence or absence of all records associated with a group in a single analysis. This provides robust guarantees in situations where correlations within a group could otherwise expose sensitive information. More formally,

Theorem 3.1.6 (Group DP)**.** *Any (ε, δ)-DP algorithm $\mathcal{A}$ is $\left(K\varepsilon, Ke^{K\varepsilon}\delta\right)$-DP for groups of size K, for all $\mathcal{S} \subseteq \mathcal{O}$:*

$$P[\mathcal{A}(Z_1) \in \mathcal{S}] \leq e^{K\varepsilon} P\left[\mathcal{A}\left(Z_2\right) \in \mathcal{S}\right] + Ke^{K\varepsilon}\delta$$

Proof. The claim follows from Definition 3.1.3 applied K times between K adjacents datasets. □

With the properties of differential privacy established, we now turn to the mechanisms that implement these guarantees, balancing privacy and utility in practice.

3.1.2 (d) Mechanisms

Differential privacy mechanisms introduce carefully calibrated randomness into the outputs of queries or algorithms, effectively masking the contribution of any single individual in the dataset. By doing so, they guarantee the formal privacy properties of differential privacy while still allowing meaningful statistical insights. In this section, we explore some of the foundational mechanisms used to achieve differential privacy,

highlighting their design principles, applications, and trade-offs between privacy and utility.

To determine the appropriate level of randomness required to obfuscate any individual's contribution, we first focus on measuring the maximum variation a query can have due to the inclusion or exclusion of a single individual. This variation is referred to as the sensitivity of the query.

Definition 3.1.6 (Sensitivity)**.** *Let $f : \mathcal{Z}^* \to \mathcal{O}$ be a query and $\|\cdot\|_q$ be the q-norm on $\mathcal{O}$. We define the sensitivity of f linked to the norm $\|\cdot\|_q$ as*

$$\Delta_q(f) = \max_{(Z_1, Z_2) \in \mathcal{Z}^*_\sim} \|f(Z_1) - f(Z_2)\|_q \tag{3.1.3}$$

We denote $\Delta_1(f)$ and $\Delta_2(f)$ the ℓ_1 and ℓ_2 sensitivities of f.

Sensitivity is central in designing mechanisms that produce a differentially private estimate of the query f. It is used to calibrate the noise to release the result of a query under a specified privacy budget. The two main mechanisms based on the sensitivity of the underlying quer are the Laplace and the Gaussian mechanisms.

We first focus on the Laplace machanism. This mechanism relies on the Laplace distribution.

Definition 3.1.7 (Laplace Distribution)**.** *The Laplace distribution Lap(λ) (centered at 0) with scale λ is the distribution with probability density function:*

$$p(x; \lambda) = \frac{1}{2\lambda} \exp\left(-\frac{|x|}{\lambda}\right), \quad x \in \mathbb{R}. \tag{3.1.4}$$

The Laplace mechanism involves adding independent Laplace noise to each entry of $f(Z)$, $Z \in \mathcal{Z}^*$, with the noise calibrated on the global ℓ_1 sensitivity and the privacy parameter ε.

Algorithm 2 Laplace Mechanism(Z, f, ε)

1: **Input:** dataset $Z \in \mathcal{Z}^*$, query f with results in $\mathbb{R}^K$, privacy parameter ε.
2: $\Delta \leftarrow \Delta_1(f)$
3: $b = (b_1, \ldots, b_K)$ with $b_k \sim Lap(\Delta/\varepsilon)$ for $k \in [K]$
4: **Output:** $f(Z) + b$.

The Laplace mechanism guarantees pure differential privacy.

Theorem 3.1.7 (DP guarantee of Laplace Mechanism)**.** *[Dwork and Roth, 2013] Let $\varepsilon \geq 0$, $f : \mathcal{Z}^* \to \mathbb{R}^K$ a query with ℓ_1-sensitivity Δ.* Laplace Mechanism$(\cdot, f, \varepsilon)$ *is ε-differentially private.*

Proof. See proof of Theorem 3.6 in [Dwork and Roth, 2013]. □

Laplace noise ensures pure differential privacy, but its distribution is less convenient than the Gaussian distribution. The Gaussian distribution, sums well, is often used in various algorithms, and fits with Rényi differential privacy.

Definition 3.1.8 (Gaussian Distribution)**.** *The Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean $\mu \in \mathbb{R}$ and variance $\sigma^2 \geq 0$ is the distribution with probability density function:*

$$p\left(x; \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \quad x \in \mathbb{R}.$$

Thanks to the Gaussian mechanism we can use the Gaussian distribution to ensure DP.

Algorithm 3 Gaussian Mechanism$(Z, f, \varepsilon, \delta)$

1: **Input:** dataset $Z \in \mathcal{Z}^*$, query f with results in $\mathbb{R}^K$, privacy parameters (ε, δ).
2: $\Delta \leftarrow \Delta_2(f)$
3: $b = (b_1, \ldots, b_K)$ with $b_k \sim \mathcal{N}(0, \sigma^2)$, with $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}\Delta}{\varepsilon}$ for $k \in [K]$
4: **Output:** $f(Z) + b$.

Note that the Gaussian mechanism is calibrated on the ℓ_2-Sensitivity of f and satisfies approximate differential privacy.

Theorem 3.1.8 (DP guarantee of Gaussian mechanism)**.** *[Dwork and Roth, 2013] Let $\varepsilon \geq 0$ and $\delta \geq 0$, $f : \mathcal{Z}^* \to \mathbb{R}^K$ a query with ℓ_2-sensitivity Δ.* Gaussian Mechanism$(\cdot, f, \varepsilon, \delta)$ *is (ε, δ)-differentially private.*

Proof. See Appendix A.1 in [Dwork and Roth, 2013]. □

The Laplace and Gaussian mechanisms fall under the category of *global sensitivity* methods. They are specifically designed for real-valued queries, making them effective for ensuring DP guarantees when releasing raw statistics. However, for problems involving optimization or selection tasks where queries are not necessarily real-valued, methods like the Exponential Mechanism are a better fit.

Instead of releasing numerical outputs with added noise, the Exponential Mechanism selects an output (e.g., a category, choice, or ranked result) from a set of possible outputs $\mathcal{O}$ based on a scoring function.

Definition 3.1.9 (Scoring function)**.**

$$s : \mathcal{Z}^* \times \mathcal{O} \to \mathbb{R}$$

The scoring function evaluates the "quality" or "utility" of each output relative to the dataset, and the mechanism assigns probabilities to outputs based on their scores. The exponential mechanism is based on the maximum possible change in an output's score when a single record is added or removed.

Definition 3.1.10 (Sensitivity of Scoring function)**.** *The sensitivity of a* s*:* $\mathcal{Z}^* \times \mathcal{O} \to \mathbb{R}$ *is*

$$\Delta(s) = \max_{O \in \mathcal{O}} \max_{(Z_1, Z_2) \in \mathcal{Z}^*_{\sim}} |s(Z_1, o) - s\,(Z_2, o)|$$

The Exponential Mechanism ensures privacy by skewing the selection probability to prioritize higher scores, while still preserving some randomness.

Algorithm 4 EXPONENTIAL MECHANISM$(Z, \mathcal{O}, s, \varepsilon)$

1: **Input:** dataset $Z \in \mathcal{Z}^*$, query f with results in $\mathbb{R}^K$, privacy parameters (ε, δ).
2: $\Delta \leftarrow \Delta(s)$
3: **Output:** o with probability $P(o) = \frac{\exp\left(\frac{s(Z,o)\cdot\varepsilon}{2\Delta}\right)}{\sum_{o' \in \mathcal{O}} \exp\left(\frac{s(Z,o')\cdot\varepsilon}{2\Delta}\right)}$.

The exponential mechanism satisfies pure differential privacy.

Theorem 3.1.9 (DP guarantee of Exponential mechanism)**.** *[Dwork and Roth, 2013] Let* $\varepsilon \geq 0$*,* $s : \mathcal{Z}^* \times \mathcal{O} \to \mathbb{R}$ *a scoring function with sensitivity* Δ*.* EXPONENTIAL MECHANISM$(\cdot, \mathcal{O}, s, \varepsilon)$ *is* ε*-differentially private.*

Proof. See proof of Theorem 3.10 in [Dwork and Roth, 2013]. □

3.2 Differentially Private Machine Learning

Machine learning often relies on aggregated statistics but remains vulnerable to privacy leaks. As discussed in Section 3.1.1 (c), aggregated statistics provide no privacy guarantees by themselves, leaving room for privacy attacks such as membership inference, differencing, and reconstruction attacks.

In this section, we will explore existing techniques that prevent from these risks by integrating traditional machine learning operations with differential privacy methods.

3.2.1 Differentially Private Empirical Risk Minimization

We remind here the Equation $\star$ that defines Empirical Risk Minimization (ERM).

Definition 3.2.1 (Empirical Risk Minimization (ERM))**.**

$$\hat{\theta} \in \arg\min_{\theta \in \Theta} F(\theta; Z) := \hat{R}(\theta; Z) + \gamma\psi(\theta) \qquad (\star')$$

where $\gamma \geq 0$ is a trade-off hyperparameter.

In the context of differential privacy, our goal is to obtain a (ε, δ)-private approximation $\tilde{\theta}$ of $\hat{\theta}$, as defined in Equation $\star'$.

3.2.1 (a) Output perturbation

Output perturbation [Chaudhuri and Monteleoni,] directly applies the Gaussian Mechanism, as defined in Algorithm 3, to the computed result $\hat{\theta}$ of ERM. By adding carefully calibrated Gaussian noise to $\hat{\theta}$, this approach ensures (ε, δ)-differential privacy for the released parameters $\tilde{\theta}$. The process is formally described in Algorithm 5.

Algorithm 5 OUTPUT PERTURBATION$(Z, f, \varepsilon, \delta)$

1: **Input:** dataset $Z \in \mathcal{Z}^*$, objective function F with parameters $\theta \in \mathbb{R}^p$ and tradeoff parameter γ, privacy parameters (ε, δ).
2: $\hat{\theta} \leftarrow \arg\min_{\theta \in \mathbb{R}^p} F(\theta; Z)$
3: $b = (b_1, \ldots, b_K)$ with $b_k \sim \mathcal{N}(0, \sigma^2)$, with $\sigma = \frac{2\sqrt{2\ln(1.25/\delta)}}{n\gamma\varepsilon}$ for $k \in [K]$
4: **Output** $\tilde{\theta}$**:** $F(Z) + b$.

Algorithm 5 ensures differential privacy.

Theorem 3.2.1 (Output perturbation DP guarantees)**.** *Let $\varepsilon, \delta > 0$ and $\Theta = \mathbb{R}^p$.* OUTPUT PERTURBATION$(\cdot, \ell, \psi, \varepsilon, \delta)$ *is (ε, δ)-DP.*

Proof. The claim follows from Theorem 3.1.8. □

Algorithm 5 suffers from two significant limitations. First, for Theorem 3.2.1 to apply, it is necessary to compute the exact solution to Equation $\star'$, which is often computationally infeasible in practice. Second, output perturbation is incompatible with federated learning frameworks, as it does not ensure the privacy of the gradients, which are shared among data owners during the training process.

3.2.1 (b) Differentially Private Stochastic Gradient Descent

Differentially Private Stochastic Gradient Descent (DP-SGD) is the most widely used algorithm for answering Equation $\star'$. Initially proposed by [Song et al., 2013], its utility optimality was later proved by [Bassily et al., 2014]. We describe DP-SGD in Algorithm 6.

Algorithm 6 DP-SGD[Bassily et al., 2014]

Input: Data set $Z \in \mathcal{Z}^*$, model f_θ, loss function ℓ L-Lipschitz, convex hypothesis space $\Theta \subseteq \mathbb{R}^k$, privacy parameters ε and δ, batch size $s \geq 1$, learning rate η, projetion to convex set Θ Π_Θ.
Initialize $\tilde{\theta}_0$ randomly from Θ
$\sigma \leftarrow \frac{16L\sqrt{T\ln(2/\delta)\ln(2.5T/\delta n)}}{n\varepsilon}$ ▷ Set Gaussian noise variance
for $t = 1$ **to** T **do**
  $V \leftarrow \emptyset$ ▷ Random sampling
  while $S = \emptyset$ **do**
    for $z \in Z$ **do**
      With probability $s/|Z|$: $V \leftarrow V \cup \{z\}$
  Draw $b_t \sim \mathcal{N}(0, \sigma^2 \mathbb{I}_k)$
  for $i = 1 \dots |V|$ **do**
    $\tilde{g}_{t,i} \leftarrow \nabla_{\tilde{\theta}_t} \ell(f_{\tilde{\theta}}(z_i))$
  $\tilde{g}_t \leftarrow \frac{1}{|V|}\left(\sum_{i=1}^{|V|} \tilde{g}_{t,i} + b_t\right)$
  $\tilde{\theta}_{t+1} \leftarrow \Pi_\Theta(\tilde{\theta}_t - \eta(t)\tilde{g}_t)$ ▷ Update and project on convex set
Output: $\tilde{\theta}_T$.

Algorithm 6 relies on two fundamental blocks to provide differential privacy: amplification by subsampling [Balle et al., 2018] and the Gaussian Mechanism as defined in Algorithm 3.

Theorem 3.2.2 (Amplification by subsampling)**.** *[Balle et al., 2018] Let $(Z, Z') \in \mathcal{Z}^* \times \mathcal{Z}^*$ and $\mathcal{M} : Z \to Z'$ be a random algorithm such that $\mathcal{M}(Z)$ returns a random subset of $|V|$ records sampled uniformly without replacement from Z. Let $\mathcal{A}$ be an (ε, δ)-DP algorithm. Then $\mathcal{A} \circ \mathcal{M}$ satisfies $\left(\varepsilon', \frac{|V|}{n}\delta\right)$-DP with $\varepsilon' = \ln\left(1 + \frac{|V|}{n}(e^\varepsilon - 1)\right)$.*

Proof. See Appendix B in [Balle et al., 2018]. □

Note that other sampling methods can provide this amplification effect as discribed by [Balle et al., 2018]. Thanks to the amplification and the Gaussian mechanisms and the advanced composition theorem 3.1.3, DP-SGD ensures differential privacy.

Theorem 3.2.3 (DP guarantee of DP-SGD)**.** DP-SGD *as defined in Algorithm 6 is* (ε, δ)*-differential private.*

Proof. See proof of Theorem 2.1 in [Bassily et al., 2014]. □

3.2.1 (c) Differentially Private Stochastic Gradient Descent with gradient clipping

Theorem 3.2.3 is based on the advanced composition theorem. A very popular variant of DP-SGD provides tighter privacy budget bounds by leveraging the privacy accountant framework based on RDP, as described in Theorem 3.1.4, also referred to as the Moment Accountant [Abadi et al., 2016]. This variant also uses gradient clipping to control sensitivity at the instance level, making it applicable to both Lipschitz and non-Lipschitz loss functions. This approach is detailed in Algorithm 7.

Algorithm 7 DP-SGDwith gradient clipping [Abadi et al., 2016]

Input: Data set $Z \in \mathcal{Z}^*$, model f_θ, loss function ℓ, hypothesis space $\Theta \subseteq \mathbb{R}^k$, noise multiplier σ, clipping norm C, batch size $s \geq 1$, learning rate η.
Initialize $\tilde{\theta}_0$ randomly from Θ
for $t = 1$ **to** T **do**
  $V \leftarrow \emptyset$ ▷ Random sampling
  while $S = \emptyset$ **do**
    for $z \in Z$ **do**
      With probability $s/|Z|$: $V \leftarrow V \cup \{z\}$
  Draw $b_t \sim \mathcal{N}(0, \sigma^2 C^2 \mathbb{I}_k)$
  for $i = 1 \dots |V|$ **do**
    $\tilde{g}_{t,i} \leftarrow \nabla_{\tilde{\theta}_t} \ell(f_{\tilde{\theta}}(z_i))$
    $\tilde{g}_{t,i} \leftarrow C\tilde{g}_{t,i} / \|\tilde{g}_{t,i}\|_2$ ▷ Clip gradient at the instance level
  $\tilde{g}_t \leftarrow \frac{1}{|V|} \left(\sum_{i=1}^{|V|} \tilde{g}_{t,i} + b_t \right)$
  $\tilde{\theta}_{t+1} \leftarrow \tilde{\theta}_t - \eta(t)\tilde{g}_t$
Output: $\tilde{\theta}_T$ and compute (ε, δ) with privacy accountant.

Theorem 3.2.4 (DP guarantee of DP-SGD with gradient clipping)**.** *[Abadi et al., 2016] There exist constants* c_1 *and* c_2 *so that given the sampling probability* $q = s/|Z|$ *and the number of steps* T*, for any* $\varepsilon < c_1 q^2 T$*, Algorithm 7 is* (ε, δ)*-differentially private for any* $\delta > 0$ *if we choose*

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

Proof. See end of Section 3.2 in [Abadi et al., 2016]. □

Regularizations. Several papers [Ioffe and Szegedy, 2015, Wu and He, 2018] have pointed out that regularization can help to improve the performance of stochastic gradient descent. Although batch normalization [Ioffe and Szegedy, 2015] does not provide protection against privacy leakage, group normalization [Wu and He, 2018] has the potential to do so [De et al., 2022]. [De et al., 2022] combines group normalization, large batch size, weight standardization [Qiao et al., 2020], augmentation multiplicity [De et al., 2022], and parameter averaging [Polyak and Juditsky, 1992] with DP-SGD to achieve state of the art performances. Note that we propose an improvement to this algorithm in Chapter 5.

Chapter 4

Background on Multi-Party Computation

Multi-Party Computation (MPC) is a cryptographic framework that enables multiple parties to jointly compute a function over their private inputs without revealing those inputs to one another. The fundamental goal of MPC is to ensure the privacy of individual inputs while guaranteeing the correctness of the computed output.

More formally, let $P_1, P_2, \ldots, P_n$ be n participants, each holding private datasets $Z_1, Z_2, \ldots, Z_n$, and let f be a public function to be computed. MPC ensures that the output $f(Z_1, Z_2, \ldots, Z_n)$ is learned by the parties without any additional information about the private datasets being revealed beyond what can be inferred from the output.

MPC protocols are evaluated based on the following core principles:

- **Privacy**: No party learns anything about another's input beyond what is revealed by the output of f.
- **Correctness**: The output of the computation is guaranteed to be correct, even in the presence of adversarial behavior.
- **Security against adversaries**: Security models in MPC consider two types of adversaries:
  - **Semi-honest adversaries**: These adversaries abides by the protocol's rules but may try to infer additional information about others using the data they observe during the execution.
  - **Malicious adversaries**: These adversaries can deviate from the protocol's expected behavior, for example, by sending incorrect values or intentionally dropping out. Malicious users can also collude with each other,

effectively forming a single adversary with access to all the information obtained by the colluding parties.

- **Efficiency**: Protocols aim to minimize computational and communication overhead while maintaining security.

4.1 Security Notions

Security parameter. In our cryptographic primitives, we use λ to denote the security parameter. A function is said to be negligible in λ if, for any positive polynomial f, it becomes smaller than $\frac{1}{f(\lambda)}$ for sufficiently large values of λ. The term 1^λ refers to a string of 1's with length λ, and $\mathcal{A}(1^\lambda)$ describes an algorithm $\mathcal{A}$ that takes an input of size $O(1^\lambda)$, meaning exponential in λ. When it is clear from the context, we may omit explicitly stating λ when referring to a 'negligible function,' implicitly meaning 'a function negligible in λ.'

Computational Indistinguishability. Computational indistinguishability is central in cryptography, as it underpins the security of many cryptographic primitives. Two distributions $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable if no efficient algorithm $\mathcal{A}$ can distinguish between them with a probability significantly greater than random guessing (up to a negligible advantage). More formally,

Definition 4.1.1 (Computational Indistinguishability)**.** *[Goldreich, 2006] Let $\mathcal{X}$ and $\mathcal{Y}$ be two distributions over the same sample space, and let $\mathcal{A}$ be a probabilistic polynomial-time (PPT) algorithm (also referred to as a distinguisher). We say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable if, for all PPT distinguishers $\mathcal{A}$, the following holds:*

$$|P\left[\mathcal{A}(x) = 1 \mid x \leftarrow \mathcal{X}\right] - P\left[\mathcal{A}(y) = 1 \mid y \leftarrow \mathcal{Y}\right]| \leq \mathit{negl}(\lambda),$$

where $\mathit{negl}(\lambda)$ is a negligible function of the security parameter λ, and $x \leftarrow \mathcal{X}$ (or $y \leftarrow \mathcal{Y}$) means that x (or y) is sampled from the distribution $\mathcal{X}$ (or $\mathcal{Y}$).

4.2 MPC Notions

Pseudorandom Generator. A pseudorandom generator (PRG) [Goldreich, 2006] is a deterministic polynomial-time algorithm $G : \{0,1\}^\lambda \to \{0,1\}^\nu$, where $\nu > \lambda$, that takes a short random seed s of length λ and produces a longer output string of length ν. The output of G is computationally indistinguishable (4.1.1) from a truly

random string of length ν. Formally, G is a pseudorandom generator if, for every PPT distinguisher D, the following holds:

$$|\mathrm{P}[D(G(s)) = 1] - \mathrm{P}[D(r) = 1]| \leq \mathrm{negl}(\lambda),$$

where s is uniformly sampled from $\{0,1\}^\lambda$, r is uniformly sampled from $\{0,1\}^\nu$, and $\mathrm{negl}(\lambda)$ is a negligible function. This guarantees that no efficient adversary can distinguish the output of G from a truly random string with non-negligible advantage.

Pseudorandom Function. A pseudorandom function (PRF) [Goldreich, 2006] is a keyed function $F : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^{n_3}$, where $\{0,1\}^{n_1}$ denotes the key space, $\{0,1\}^{n_2}$ denotes the input space, and $\{0,1\}^{n_3}$ denotes the output space. The function F is considered pseudorandom if, for every PPT distinguisher D, the following holds:

$$\left|P\left[D^{F(k,\cdot)}\left(1^\lambda\right) = 1\right] - P\left[D^{f(\cdot)}\left(1^\lambda\right) = 1\right]\right| \leq \mathrm{negl}(\lambda),$$

where the first probability is taken over a uniformly chosen key $k \in \{0,1\}^{n_1}$, and the second probability is taken over a uniformly chosen function f from the set of all functions mapping $\{0,1\}^{n_2}$ to $\{0,1\}^{n_3}$. This definition formalizes the notion that $F(k,\cdot)$, with a randomly chosen key k, is computationally indistinguishable from a truly random function f.

4.2.1 Secret Sharing

Secret sharing [Goldreich, 2006] is a foundational building block of MPC. It involves splitting a secret s into t shares $s_1, s_2, \ldots, s_t$, such that the original secret can only be reconstructed if a sufficient number of shares are combined. These shares are distributed among the parties in a way that ensures no individual party can independently recover the secret. Consequently, no party should possess more than $t-1$ shares.

Additive Secret Sharing. Additive secret sharing [Goldreich, 2006] is a scheme for distributing a secret $s \in \mathbb{Z}_k$ among n parties by splitting it into n shares $x_1, x_2, \ldots, x_n$, such that the shares satisfy $\sum_{i=1}^{n} x_i = x \mod 2^k$. Each share x_i is chosen uniformly at random from $\mathbb{Z}_k$, except for the final share, which is determined to ensure the sum condition holds. This ensures that any subset of fewer than n shares provides no information about the secret s, as each share is indistinguishable from a random value in $\mathbb{Z}_k$. Arithmetic operations on the shares, such as addition and multiplication, are performed modulo 2^k. The scheme achieves n-out-of-n sharing, meaning all n shares are required to reconstruct the secret. Extensions of this scheme, such

as replicated secret sharing, allow for greater fault tolerance by distributing multiple copies of shares across parties.

Shamir Secret Sharing. [Shamir, 1979] Let s be a secret to be shared among n parties such that at least t parties are required to reconstruct the secret (t-out-of-n scheme). Shamir's secret sharing scheme operates over a finite field $\mathbb{F}_q$, where $q > n$, and encodes the secret s as the constant term of a random polynomial $P(y)$ of degree $t-1$. Formally, $P(y)$ is defined as:

$$P(y) = s + a_1 y + a_2 y^2 + \cdots + a_{t-1} y^{t-1},$$

where $a_1, a_2, \ldots, a_{t-1}$ are chosen uniformly at random from $\mathbb{F}_q$. The n shares are computed by evaluating the polynomial $P(y)$ at n distinct, nonzero values $y_1, y_2, \ldots, y_n$, such that the i-th share is the pair $(y_i, P(y_i))$. Each party is given exactly one share.

The reconstruction of the secret requires at least t shares. Using any t pairs $(y_i, P(y_i))$, the polynomial $P(y)$ can be uniquely reconstructed using Lagrange interpolation. The secret s is then recovered as the constant term $P(0)$. Without at least t shares, the secret s remains information-theoretically secure, as fewer than t points reveal no information about $P(y)$.

Function Secret Sharing (FSS). Function Secret Sharing (FSS) [Boyle et al., 2015] extends the principles of secret sharing to functions rather than data, enabling the secure evaluation of functions in distributed settings. While additive secret sharing and Shamir secret sharing allow a secret value to be split among multiple parties, FSS focuses on sharing a function such that each party holds a share of the function, and together, they can compute the function's output without revealing individual inputs or the entire function. Specifically, FSS generates secret shares of a function a function f into two or more share functions $f_1, f_2, \ldots, f_p$ by constructing function keys $k_1, k_2, \ldots, k_p$ for each participant, such that the original function $f(x)$ can be computed by combining the outputs of the share functions at x. Each party holds one share of the function and computes its output independently, allowing for distributed computation while preserving privacy.

More formally, a FSS scheme for $\mathcal{F}$ a class of functions is a pair of PPT algorithms (Gen, Eval) so that:

- Gen $\left(1^\lambda, f\right)$: For the security parameter 1^λ and $f \in \mathcal{F}$, Gen outputs p keys, $(k_1, \ldots, k_p)$.
- Eval (i, k_i, x) : For $x \in \mathit{Domain}(f)$, Eval outputs y_i, the party's share of $f(x)$.

satisfying the following correctness *i.e.,* , $\sum_{i=1}^{p} y_i = f(x)$, and secrecy *i.e.,* , keys k_i are computationally indistinguishable.

FSS can be applied to structured functions that allow efficient splitting into independent components, typically functions that can be expressed as sums or combinations of simpler terms. Functions like additive functions, piecewise functions, boolean functions or comparaision functions are good candidates for FSS. However, FSS is generally not applicable to arbitrary functions, especially those that are highly non-linear, unstructured, or require extensive statefulness.

FFS is a compact way of securely computing and sharing values of a function. It is particularly useful in scenarios where it is necessary to offload computation to multiple parties without revealing sensitive information. It has applications in private set intersection, distributed machine learning, and secure data aggregation. By minimizing communication and computational overhead, FSS contributes to the efficiency of many modern MPC protocols.

Chapter 5

DPSGD with weight clipping

Recent advancements in DP-SGD have explored replacing traditional gradient clipping with Lipschitz constraints to bound sensitivity. While these approaches are conceptually appealing, they have so far failed to match the performance of standard gradient clipping-based DP-SGD methods. A key limitation is that parameter-wise Lipschitz constants can become excessively large in deep architectures, resulting in overly conservative noise scaling and degraded model utility. In this work, we introduce a novel algorithm that addresses this limitation by incorporating data-dependent gradient scaling, rather than relying solely on worst-case Lipschitz bounds. This adjustment enables the use of deeper neural networks while maintaining strong privacy guarantees. Our method leverages Lipschitz constraints more effectively and achieves state-of-the-art performance on standard vision benchmarks, outperforming existing DP-SGD variants.

This Chapter is mostly based on the paper: Antoine Barczewski, Jan Ramon. DP-SGD with weight clipping. CAp (Conférence sur l'Apprentissage automatique) 2024, SSFAM (Société Savante Française d'Apprentissage Machine); AFRIF (Association Française pour la Reconnaissance et l'Interprétation des Formes), Jul 2024, Lille (France), France. ⟨10.48550/arXiv.2310.18001⟩. ⟨hal-04614505⟩
The code corresponding to this Chapter is available at https://gitlab.inria.fr/abarczew/ab_technical/-/tree/master/empirical_optimum/lip_norm.

5.1 Introduction

In this chapter we propose a novel approach for training a machine learning model with privacy guarantees as we introduced in Section 3.2.1.

A popular class of algorithms to realize differential privacy while performing SGD is the DP-SGD algorithm [Abadi et al., 2016] and its variants. Essentially, these algo-

rithms iteratively compute gradients, add differential privacy noise, and use the noisy gradient to update the model as described in Algorithm 7. To determine the level of differential privacy achieved, one uses an appropriate composition rule to bound the total information leaked in the several iterations, please refer to Section 3.1.2 (c) for more details.

To achieve differential privacy with a minimum amount of noise, it is important to be able to bound precisely the sensitivity of the information which the participants will observe. One approach is to bound the sensitivity of the gradient by assuming the objective function is Lipschitz continuous [Bassily et al., 2014]. Various improvements exist in the case one can make additional assumptions about the objective function. For example, if the objective function is strongly convex, one can bound the number of iterations needed and in that way avoid to have to distribute the available privacy budget over too many iterations [Bassily et al., 2019]. In the case of DNN, the objective function is not convex and typically not even Lipschitz continuous. Therefore, a common method is to 'clip' contributed gradients [Abadi et al., 2016], i.e., to divide gradients by the maximum possible norm they may get. These normalized gradients have bounded norm and hence bounded sensitivity.

Recent works have shown that DP-SGD can be performed without gradient clipping by leveraging Lipschitz constraints on the model [Bethune et al., 2023]. Thanks to the Lipschitz constraints enforced by weight clipping, the sensitivity of the gradient can be computed in a data-independent way, which is conceptually a significant improvement over the classic DP-SGD algorithm. However, this method still suffers from a significant performance drop when applied to deep architectures.

In this chapter, we argue that gradient clipping may not lead to optimal statistical results (see Section 5.3), and we propose instead to use weight clipping. Unlike [Bethune et al., 2023], our method achieves competitive performance with state-of-the-art DP-SGD methods, while still relying on Lipschitz constraints to avoid gradient clipping. Moreover, we also propose to consider the maximum gradient norm given the current position in the search space rather than the global maximum gradient norm, as this leads to additional advantages. In particular, our contributions are as follows:

- We introduce a novel approach, applicable to any feed-forward neural network, to compute gradient sensitivity. This approach, built upon [Bethune et al., 2023], eliminates the need for gradient clipping in DP-SGD and mitigates the effect of exploding Lipschitz values on deep architectures.
- We present a new algorithm, Lip-DP-SGD, that enforces bounded sensitivity of the gradients , and (2) adaptively upper bounds the gradient norm depending on the (publicly known) current model. We argue that our approach, based on

weight clipping, does not suffer from the bias which the classic gradient clipping can cause.

- We present an empirical evaluation, confirming that on a range of popular datasets our proposed method outperforms existing ones.

The remainder of this chapter is organized as follows. First, we present our new method in Section 5.2 and present an empirical evaluation in Section 5.3. Then, we discuss related work in Section 5.4. Finally, we provide conclusions and directions for future work in Section 6.7. Note that, a number of basic concepts, definitions and notations are defined in Chapter 2 and Chapter 3.

5.2 Our approach

In this work, we constrain the objective function to be Lipschitz and leverage this structure to estimate sensitivity. Unlike traditional DP-SGD, which controls sensitivity by clipping gradients at the per-sample level, our approach computes sensitivity in a model-dependent but data-independent manner. This strategy builds upon the work of [Bethune et al., 2023], who established a link between input and parameter Lipschitz constants to enable differentially private learning.

Extending their insight, we introduce a layer-wise gradient scaling mechanism that significantly tightens the upper bound on sensitivity—a key contribution of our method. In Subsection 5.2.1, we describe how we compute upper bounds on Lipschitz constants. Subsection 5.2.2 revisits the use of backpropagation for estimating gradient sensitivity, highlighting our novel gradient scaling step. Finally, in Subsection 5.2.3, we present Lip-DP-SGD, a new algorithm that achieves differential privacy without requiring gradient clipping, thanks to the use of scaled gradients.

5.2.1 Estimating lipschitz values

In this section we bound Lipschitz values of different types of layers. We treat linear operations (e.g., linear transformations, convolutions) and activation functions as different layers. We present results with regard to any p-norms with $p \in \{1, 2, +\infty\}$.

Loss function and activation layer. Examples of Lipschitz losses encompass Softmax Cross-entropy, Cosine Similarity, and Multiclass Hinge. When it comes to activation layers, layers composed of an activation function, several prevalent ones, such as ReLU, tanh, and Sigmoid are 1-Lipschitz with respect to all p-norms. We provide a detailed list in table 5.1.

Normalization layer. To be able to easily bound sensitivity, we define the operation of a normalization layer $f_{\theta_k}^{(k)}$ slightly differently than Eq (2.4.2):

$$x_{k+1}^{(k:q)} = f_{\theta_k}^{(k)}(x_k^{(k:q)}) = \frac{x_k^{(k:q)} - \mu^{(k:q)}}{\max(1, \sigma^{(k:q)})}. \tag{5.2.1}$$

It is easy to see that the sensitivity is bounded by

$$\left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial x_k} \right\|_p \leq \max_{q \in [|\Gamma_k|]} \frac{1}{\max\left(1, \sigma^{(k:q)}\right)} \leq 1. \tag{5.2.2}$$

Note that a group normalization layer has no trainable parameters.

Linear layers. [Gouk et al., 2020] presents a formulation for the Lipschitz value of linear layers in relation to p-norms. This formulation can be extended to determine the Lipschitz value with respect to the parameters. Therefore, if $f_{\theta_k}^{(k)}$ is a linear layer, then

$$\begin{aligned} \left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k} \right\|_p &= \left\| \frac{\partial (W_k^\top x_k + B_k)}{\partial (W_k, B_k)} \right\|_p = \|(\vec{x_k}, 1)\|_p, \\ \left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial x_k} \right\|_p &= \left\| \frac{\partial (W_k^\top x_k + B_k)}{\partial x_k} \right\|_p = \|W_k\|_p, \end{aligned} \tag{5.2.3}$$

with $\vec{x_k}$ the serialized vector of x_k.

Convolutional layers. There are many types of convolutional layers, e.g., depending on the data type (strings, 2D images, 3D images ...), shape of the filter (rectangles, diamonds ...). Here we provide as an example only a derivation for convolutional layers for 2D images with rectangular filter. In that case, the input layer consists of $n_k = c_{in}hw$ nodes and the output layer consists of $n_{k+1} = c_{out}hw$ nodes with c_{in} input channels, c_{out} output channels, h the height of the image and w the width. Then, $\theta_k \in \mathbb{R}^{c_{in} \times c_{out} \times h' \times w'}$ with h' the height of the filter and w' the width of the filter. Indexing input and output with channel and coordinates, i.e., $x_k \in \mathbb{R}^{c_{in} \times h \times w}$ and $x_{k+1} \in \mathbb{R}^{c_{out} \times h \times w}$ we can then write

$$x_{k+1,c,i,j} = \sum_{d=1}^{c_{in}} \sum_{r=1}^{h'} \sum_{s=1}^{w'} x_{k,d,i+r,j+s} \theta_{k,c,d,r,s}$$

where components out of range are zero.

Theorem 5.2.1. *The convolved feature map* $(\theta * \cdot) : \mathbb{R}^{n_k \times |x_k|} \to \mathbb{R}^{n_{k+1} \times n \times n}$*, with zero or circular padding, is Lipschitz and*

$$\|\nabla_{\theta_k}(\theta_k * x_k)\|_p \leq \sqrt{h'w'}\|x_k\|_p \text{ and } \|\nabla_{x_k}(\theta_k * x_k)\|_p \leq \sqrt{h'w'}\|\theta_k\|_p \tag{5.2.4}$$

with w' *and* h' *the width and the height of the filter.*

Proof. The output $x_{k+1} \in \mathbb{R}^{c_{out} \times n \times n}$ of the convolution operation is given by:

$$x_{k+1,c,r,s} = \sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} x_{k,d,r+i,s+j}\theta_{k,c,d,i,j}$$

There follows, for any $p \in \{1, 2, +\infty\}$:

$$\begin{aligned}
\|x_{k+1}\|_p^p &= \sum_{c=0}^{c_{out}-1} \sum_{r=1}^{n} \sum_{s=1}^{n} \left| \sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} x_{k,d,r+i,s+j}\theta_{k,c,d,i,j} \right|^p \\
&\leq \sum_{c=0}^{c_{out}-1} \sum_{r=1}^{n} \sum_{s=1}^{n} \left(\sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} |x_{k,d,r+i,s+j}|^p \right) \left(\sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} |\theta_{k,c,d,i,j}|^p \right) \text{ (triangle inequality)} \\
&= \left(\sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} \sum_{r=1}^{n} \sum_{s=1}^{n} |x_{k,d,r+i,s+j}|^p \right) \left(\sum_{c=0}^{c_{out}-1} \sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} |\theta_{k,c,d,i,j}|^p \right) \\
&\leq h'w' \left(\sum_{d=0}^{c_{in}-1} \sum_{r=1}^{n} \sum_{s=1}^{n} |x_{k,d,r,s}|^p \right) \left(\sum_{c=0}^{c_{out}-1} \sum_{d=0}^{c_{in}-1} \sum_{i=0}^{h'-1} \sum_{j=0}^{w'-1} |\theta_{k,c,d,i,j}|^p \right) \\
&= h'w'\|x_k\|_p^p\|\theta_k\|_p^p
\end{aligned}$$

Since $\theta_k * \cdot$ is a linear operator:

$$\|(\theta_k * x_k) - (\theta'_k * x_k)\|_p = \|(\theta_k - \theta'_k) * x_k\|_p \leq \|\theta_k - \theta'_k\|_p (h'w')^{\frac{1}{p}} \|x_k\|_p$$

Finally, the convolved feature map is differentiable so the spectral norm of its Jacobian is bounded by its Lipschitz value:

$$\|\nabla_{\theta_k}(\theta_k * x_k)\|_p \leq (h'w')^{\frac{1}{p}} \|x_k\|_p$$

Analogously,

$$\|\nabla_{x_k}(\theta_k * x_k)\|_p \leq (h'w')^{\frac{1}{p}} \|\theta_k\|_p$$

□

From theorem 5.2.1, we state that,

$$\left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial x_k} \right\|_p \leq (h'w')^{\frac{1}{p}} \|\theta_k\|_p \tag{5.2.5}$$

$$\left\| \frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k} \right\|_p \leq (h'w')^{\frac{1}{p}} \|\vec{x_k}\|_p \tag{5.2.6}$$

Residual connections Resnet architectures [He et al., 2016] are usually based on residual blocks:

$$f_{\theta_{j+k}}^{(j+k)}(x_j) = x_j + (f_{\theta_{j+k-1}}^{(j+k-1)} \circ \ldots \circ f_{\theta_j}^{(j)})(x_j)$$

[Gouk et al., 2020] shows that the Lipschitz value of residual blocks is bounded by

$$\left\| \frac{\partial f_{\theta_{k+j}}^{(k+j)}}{\partial x_j} \right\|_p \leq 1 + \prod_{i=j}^{j+k-1} \left\| \frac{\partial f_{\theta_i}^{(i)}}{\partial x_i} \right\|_p . \tag{5.2.7}$$

We summarize the upper bounds of the Lipschitz values, either on the input or on the parameters, for each layer type in table 5.1. We can conclude that networks for which the norms of the parameter vectors θ_k are bounded, are Lipschitz networks as introduced in [Miyato et al., 2018], i.e., they are FNN for which each layer function $f_{\theta_k}^{(k)}$ is Lipschitz. We will denote by $\Theta_{\leq C}$ and by $\Theta_{=C}$ the sets of all paremeter vectors θ for f_θ such that $\|\theta_k\| \leq C$ and $\|\theta_k\| = C$ respectively, for $k = 1 \ldots K$.

Table 5.1: Summary table of the upper bounds of the Lipschitz values, either on the input (Lip. on x_k) or on the parameters (Lip. on θ_k), for each layer type. Note that for the loss, the Lipschitz value is solely dependent on the output x_{K+1}. Please refer to section 2.5 for more details on loss functions.

Layer	Definition	Lip. on x_k	Lip. on θ_k
Dense	$\theta_k^\top x_k$	$\|\theta_k\|$	$\|x_k\|$
Convolutional	$\theta_k * x_k$	$\sqrt{h'w'}\|\theta_k\|$	$\sqrt{h'w'}\|x_k\|$
Normalization	$(x_k^{(k:q)} - \mu^{(k:q)}) / \max(\alpha, \sigma^{(k:q)})$	$1/\alpha$	-
ReLU	$\max(x_k, 0)$	1	-
Sigmoid	$1/(1+e^{-x_k})$	1	-
Tanh	$(e^x - e^{-x})(e^x + e^{-x})$	1	-
Softmax Cross-entropy	$y^\top \log(\textsc{softmax}(x_{K+1}))/\tau$	$\sqrt{2}/\tau$	-
Cosine Similarity	$x_{K+1}^\top y / (\|x_{K+1}\|\|y\|)$	$1/\min\|x_{K+1}\|$	-
Multiclass Hinge	$\sum_{\substack{i=1 \\ i \neq \arg\max y}}^{c} \max(0, m - x_{K+1} \cdot y + x_{K+1,i})/c$	1	-

5.2.2 Backpropagation

Consider a feed-forward network f_θ. We define $\mathcal{L}_k(\theta, (x_k, y)) = \ell\left(\left(f_{\theta_K}^{(K)} \circ \ldots \circ f_{\theta_k}^{(k)}\right)(x_k), y\right)$. For feed-forward networks, the chain rule gives:

$$\frac{\partial \mathcal{L}_k}{\partial x_k} = \frac{\partial \ell}{\partial x_{K+1}} \frac{\partial f_{\theta_K}^{(K)}}{\partial x_k}. \tag{5.2.8}$$

Any matrix or vector norm is submultiplicative, especially the $\|\cdot\|_p$ norm with $p \in \{1, 2, +\infty\}$, hence:

$$\left\|\frac{\partial \mathcal{L}_k}{\partial x_k}\right\|_p \leq \left\|\frac{\partial \ell}{\partial x_{K+1}}\right\|_p \left\|\frac{\partial f_{\theta_K}^{(K)}}{\partial x_k}\right\|_p. \tag{5.2.9}$$

In section 5.2.1, we show that the Lipschitz value with regard to the input of $f_{\theta_k}^{(k)}$ is bounded and the bound depends on the norm of the parameters in the case of linear or convolutional layers. Let c_k be the Lipschitz value with regards to the input of any layer k. As f_{θ_k} is Lipschitz constrained, when the layer k has parameters, c_k is a linear function of $\min(C, \|\theta_k\|)$, as stated in eqs. (5.2.3) and (5.2.5), with C the maximum weight norm. If $\left\|\frac{\partial \ell}{\partial x_{K+1}}\right\|_p \leq \tau$, where ℓ represents the loss then,

$$\left\|\frac{\partial \mathcal{L}_k}{\partial x_k}\right\|_p \leq \tau \prod_{i=k}^{K} c_i. \tag{5.2.10}$$

We now consider the induced $\|\cdot\|_{2,p}$ norm ($\|A\|_{\alpha,\beta} = \sup_{x \neq 0} \frac{\|Ax\|_\beta}{\|x\|_\alpha}$) for studiying $\frac{\partial \mathcal{L}_k}{\partial \theta_k}$. Induced norms are consistent, then one can prove that $\|AB\|_{\alpha,\gamma} \leq \|A\|_{\beta,\gamma}\|B\|_{\alpha,\beta}$ [Cape et al., 2017] which, applied to the chain rule, gives:

$$\left\|\frac{\partial \mathcal{L}_k}{\partial \theta_k}\right\|_{2,p} \leq \left\|\frac{\partial \mathcal{L}_{k+1}}{\partial x_{k+1}}\right\|_p \left\|\frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k}\right\|_{2,p} \tag{5.2.11}$$

Combining 5.2.10 and 5.2.11 provides an upper bound of the $2,p$-norm of the gradient at layer k,

$$\left\|\frac{\partial \mathcal{L}_k}{\partial \theta_k}\right\|_{2,p} \leq \tau \prod_{i=k+1}^{K} c_i \left\|\frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k}\right\|_{2,p} \tag{5.2.12}$$

If $\theta_k \neq \emptyset$, then eqs. (5.2.3) and (5.2.6) show that the upper bound of $\left\|\frac{\partial f_{\theta_k}^{(k)}}{\partial \theta_k}\right\|_{2,p}$ depends on $\|\vec{x_k}\|_2$. Hence,

$$\left\| \frac{\partial \mathcal{L}_k}{\partial \theta_k} \right\|_{2,p} \leq \tau X_k \prod_{i=k+1}^{K} c_i, \tag{5.2.13}$$

with $X_k = \sqrt{hw} \max_{x_k \in V_k} \|\vec{x_k}\|_2$ in case of a convolutional layer and $X_k = \max_{x_k \in V_k} \|\vec{x_k}\|_2$ in case of a linear layer, with $V_k = \{(f_k \circ \ldots \circ f_1)(x) | (x, y) \in V\}$. Note that in the case of other layers, such as normalization or activation layers, there is no gradient with respect to the parameters, hence eq. (5.2.13) is not applicable.

[Bethune et al., 2023] estimates gradient sensitivity by computing an upper bound on the norm using Equation (5.2.13), where the right-hand side is data-independent to ensure privacy. As a result, X_k is estimated by a value we call here X'_k, the maximum input norm across the entire input space at one specific iteration. Formally, their method computes for each layer k:

$$X'_k \leftarrow \max_{\|x\| \leq X'_{k-1}} \|f_k(\theta_k, x)\|_2$$

a process referred to as *input bounds propagation*. Although they normalize the first layer's input, the *input bounds propagation* does not prevent the norm from compounding across layers—each layer scales outputs by the next layer's Lipschitz constant and adds bias terms, which is not Lipschitz-constrained. This issue is amplified in deep architectures like ResNets, where norms grow linearly with the number of residual blocks.

In contrast, we scale the gradient at each layer using directly X_k which is computed batch-wise, and bound the norm of this scaled gradient. By construction, $X_k \leq X'_k$, leading to tighter sensitivity estimates and reduced noise. Formally, we define the scaled gradient at layer k as:

$$\forall (x, y) \in \mathcal{X} \times \mathcal{Y}, \quad \left\| \frac{\nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x), y)}{X_k} \right\|_{2,p} \leq \tau \prod_{i=k+1}^{K} c_i \tag{5.2.14}$$

5.2.3 Lip-DP-SGD

We introduce in algorithm 8 a novel differentially private stochastic gradient descent algorithm, called Lip-DP-SGD, that leverages the estimation of the per-layer scaled gradient norm upper bound to provide differential privacy without gradient clipping.

Differential Privacy. The left-hand side of eq. (5.2.14) represents the gradient scaled by the maximum input norm. The $\ell_{2,p}$-sensitivity of this scaled gradient is

Algorithm 8 LIP-DP-SGD: Differentially Private Stochastic Gradient Descent with Lipschitz constrains.

Input: Data set $Z \in \mathcal{Z}^*$, feed-forward model f_θ, loss function ℓ with Lipschitz value τ, hypothesis space $\Theta \subseteq \mathbb{R}^k$, number of epochs T, noise multiplier σ, batch size $s \geq 1$, learning rate η, norm $p \in \{1, 2, \infty\}$, max weight norm C.
Initialize $\tilde{\theta}$ randomly from Θ
$(c_k, \tilde{\theta_k})_{k=1}^K \leftarrow$ CLIPWEIGHTS($\tilde{\theta}$,C, p)
for $t \in [T]$ **do**
    $(\Delta_k)_{k=1}^K \leftarrow (\tau \prod_{i=k}^K c_i)_{k=1}^K$ ▷ Lipschitz value at layer k, see eq. (5.2.5)
    $V \leftarrow \emptyset$ ▷ Poisson sampling
    while $V = \emptyset$ **do**
        for $z \in Z$ **do**
            With probability $s/|Z|$: $V \leftarrow V \cup \{z\}$
    for $k = 1 \dots K$ **do** ▷ Compute gradient per layer
        $X_k \leftarrow \max_{i \in |V|} \|f_{\tilde{\theta}_{k-1}}(x_i)\|_2$ ▷ Max input norm of layer k, see eq. (5.2.13)
        Draw $b_k \sim \mathcal{N}(0, \sigma^2 \Delta_k^2 \mathbb{I})$
        $\tilde{g}_k \leftarrow \frac{1}{|V|} \left(\frac{1}{X_k} \sum_{i=1}^{|V|} \nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x_i), y_i) + b_k \right)$ ▷ DP gradient, see eq. (5.2.14)
        $\tilde{\theta}_k \leftarrow \tilde{\theta}_k - \eta(t) \tilde{g}_k$ ▷ Update
    $(c_k, \tilde{\theta_k})_{k=1}^K \leftarrow$ CLIPWEIGHTS($\tilde{\theta}$,C, p) ▷ Enforce Lipschitzness
Output: $\tilde{\theta}$ and compute (ϵ, δ) with privacy accountant.
function CLIPWEIGHTS($\tilde{\theta}$, C)
    for $k = 1 \dots K$ **do**
        if $\theta_k \neq \emptyset$ **then**
            $c_k \leftarrow \min(C, \|\tilde{\theta}_k\|_p)$
            $\tilde{\theta}_k \leftarrow c_k \tilde{\theta}_k / \|\tilde{\theta}_k\|_p$
        else
            $c_k \leftarrow 1$
    return $(c_k, \tilde{\theta_k})_{k=1}^K$

bounded, with the upper-bound depending solely on the Lipschitz constant of the loss function τ, the maximum parameter norm C, and/or the parameter norm $\|\tilde{\theta_k}\|_{2,p}$.

Theorem 5.2.2. *Given a feed-forward model f_θ composed of Lipschitz constrained operators and a Lipschitz loss ℓ,* LIP-DP-SGD *is differentially private.*

Indeed, the scaled gradient's sensitivity is determined without any privacy costs, as it depends only on the current parameter values (which are privatized in the previous step, and post-processing privatized values does not take additional privacy budget) and not on the data. If the selected norm is the $2, 2$-norm, the Gaussian mechanism

can be applied to ensure privacy by utilizing the sensitivity of the scaled gradient. Otherwise, the exponential mechanism can be employed to achieve differential privacy [McSherry and Talwar, 2007] using a custom score function.

Privacy accounting. Lip-DP-SGD adopts the same privacy accounting as DP-SGD. Specifically, the accountant draws upon the privacy amplification [Kasiviswanathan et al., 2011] brought about by Poisson sampling and the Gaussian moment accountant [Abadi et al., 2016]. It's worth noting that while we utilized the Renyi Differential Privacy (RDP) accountant [Abadi et al., 2016, Mironov et al., 2019] in our experiments, Lip-DP-SGD is versatile enough to be compatible with alternative accountants. Note that RDP is primarily designed to operate with the Gaussian mechanism, but [Wang et al., 2019] demonstrate its applicability with other differential privacy mechanisms, particularly the exponential mechanism.

Requirements. As detailed in Section 5.2.1, the loss and the model operators need to be Lipschitz. We have enumerated several losses and operators that meet this criterion in Table 5.1. While we use ClipWeights to characterize Lipschitzness [Yoshida and Miyato, 2017, Miyato et al., 2018] in our study 5.2.1, other methods are also available, as discussed in [Arjovsky et al., 2017].

ClipWeights. The ClipWeights function is essential to the algorithm, ensuring Lipschitzness, which facilitates model sensitivity estimation. As opposed to standard Lipschitz-constrained networks [Yoshida and Miyato, 2017, Miyato et al., 2018] which increase or decrease the norms of parameters to make them equal to a predefinied value, our approach normalizes weights only when their current norm exceeds a threshold. This results in adding less DP noise for smaller norms. Importantly, as $\tilde{\theta}$ is already made private in the previous iteration, its norm is private too. Note that, when the norm used is the 2-norm i.e., ClipWeights is a spectral normalization, we perform also a Björck orthogonalization for fast and near-orthogonal convolutions [Li et al., 2019].

Computing norms The $\ell_{2,1}$ and $\ell_{2,\infty}$ norms can be computed exactly in linear time relative to the number of elements. However, calculating the $\ell_{2,2}$ norm, which corresponds to the largest singular value of the matrix, is infeasible using standard singular value decomposition techniques. To address this efficiently, we use techniques based on power method [Gouk et al., 2020] that leverage the backward computational graph of modern deep learning libraries like [Abadi et al., 2015, TensorFlow] and [Paszke et al., 2019, PyTorch].

5.2.4 Avoiding the bias of gradient clipping

We show that Lip-DP-SGD converges to a local minimum in $\Theta_{\leq C}$ while DP-SGD suffers from bias and may converge to a point which is not a local minimum of Θ.

We use the word 'converge' here somewhat informally, as in each iteration independent noise is added the objective function slightly varies between iterations and hence none of the mentioned algorithms converges to an exact point. We here informally mean approximate convergence to a small region, assuming a sufficiently large data set Z and/or larger ϵ such that privacy noise does not significantly alter the shape of the objective function. Our argument below hence makes abstraction of the noise for simplicity, but in the presence of small amounts of noise a similar argument holds approximately, i.e., after sufficient iterations Lip-DP-SGD will produce θ values close to a locally optimal θ^* while DP-SGD may produce θ values in a region not containing the relevant local minimum.

First, let us consider convergence.

Theorem 5.2.3. *Let F be an objective function as defined in equation $\star$, and Z, f_θ, $\mathcal{L}$, $\Theta = \Theta_{\leq C}$, T, $\sigma = 0$, s, η and C be input parameters of* Lip-DP-SGD *satisfying the requirements specified in Section 5.2.3. Assume that for these inputs* Lip-DP-SGD *converges to a point θ^* (in the sense that $\lim_{k,T\to\infty} \theta_k = \theta^*$). Then, θ^* is a local optimum of $F(\theta, Z)$ in $\Theta_{\leq C}$.*

Proof sketch. We consider the problem of finding a local optimum in $\Theta_{\leq C}$:

$$\begin{array}{ll} \text{minimize} & F(\theta, Z) \\ \text{subject to} & \|\theta\|_2 \leq C \end{array}$$

We introduce a slack variable ζ:

$$\begin{array}{ll} \text{minimize} & F(\theta, Z) \\ \text{subject to} & \|\theta\|_2 + \zeta^2 = C \end{array}$$

Using Lagrange multipliers, we should minimize

$$F(\theta, Z) - \lambda(\|\theta\|_2 + \zeta^2 - C)$$

An optimum in θ, λ and ζ satisfies

$$\nabla_\theta F(\theta, Z) - \lambda\theta = 0 \tag{5.2.15}$$
$$\|\theta\|_2 + \zeta^2 - C = 0 \tag{5.2.16}$$
$$2\lambda\zeta = 0 \tag{5.2.17}$$

From Eq 5.2.17, either $\lambda = 0$ or $\zeta = 0$ If $\zeta > 0$, θ is in the interior of $\Theta_{\leq C}$ and there follows $\lambda = 0$ and from Eq 5.2.15 that $\nabla_\theta F(\theta, Z) = 0$. For such θ, Lip-DP-SGD

does not perform weight clipping. If the learning rate is sufficiently small, and if it converges to a θ with norm $\|\theta\|_2 < C$ it is a local optimum. On the other hand, if $\zeta = 0$, there follows from Eq 5.2.16 that $\|\theta\|_2 = C$, i.e., θ is on the boundary of $\Theta_{\leq C}$. If θ is a local optimum in $\Theta_{\leq C}$, then $\nabla_\theta F(\theta, Z)$ is perpendicular on the ball of vectors θ with norm C, and for such θ Lip-DP-SGD will add the multiple $\eta(t).\nabla_\theta F(\theta, Z)$ to θ and will next scale θ back to norm C, leaving θ unchanged. For a θ which is not a local optimum in $\Theta_{\leq C}$, $\nabla_\theta F(\theta, Z)$ will not be perpendicular to the ball of C-norm parameter vectors, and adding the gradient and bringing the norm back to C will move θ closer to a local optimum on this boundary of $\Theta_{\leq C}$. This is consistent with Eq 5.2.15 which shows the gradient with respect to θ for the constrained problem to be of the form $\nabla_\theta F(\theta, Z) - \lambda\theta$. □

Theorem 5.2.3 shows that in a noiseless setting, if Lip-DP-SGD converges to a stable point that point will be a local optimum in $\Theta_{\leq C}$. In the presence of noise and/or stochastic batch selection, algorithms of course do not converge to a specific point but move around close to the optimal point due to the noise in each iteration, and advanced methods exist to examine such kind of convergence. The conclusion remains the same: Lip-DP-SGDwill converge to a neighborhood of the real local optimum, while as we argue DP-SGD will often converge to a neighborhood of a different point.

Second, we argue that DP-SGD introduces bias. This was already pointed out in [Chen et al., 2020]'s examples 1 and 2. In Section 5.3.4 we also showed experiments demonstrating this phenomenon. Below, we provide a simple example which we can handle (almost) analytically.

A simple situation where bias occurs and DP-SGD does not converge to an optimum of F is when errors aren't symmetrically distributed, e.g., positive errors are less frequent but larger than negative errors.

Consider the scenario of simple linear regression. A common assumption of linear regression is that instances are of the form (x_i, y_i) where x_i is drawn from some distribution P_x and $y_i = ax_i + b + e_i$ where e_i is drawn from some zero-mean distribution P_e. When no other evidence is available, one often assume P_e to be Gaussian, but this is not necessarily the case. Suppose for our example that P_x is the uniform distribution over $[0, 1]$ and P_e only has two possible values, in particular $P_e(9) = 0.1$, $P_e(-1) = 0.9$ and $P_e(e) = 0$ for $e \notin \{9, -1\}$. So with high probability there is a small negative error e_i while with small probability there is a large positive error, while the average e_i is still 0. Consider a dataset $Z = \{(x_i, y_i)\}_{i=1}^n$. Let us consider a model $f(x) = \theta_1 x\theta_2$ and let us use the square loss $\mathcal{L}(\theta, Z) = \sum_{i=1}^n \ell(x_i, y_i)/n$ with $\ell(\theta, x, y) = (\theta_1 x + \theta_2 - y)^2$. Then, the gradient is

$$\nabla_\theta \ell(\theta, x, y) = (2(\theta_1 x + \theta_2 - y)x, 2(\theta_1 x + \theta_2 - y))$$

For an instance (x_i, y_i) with $y_i = ax_i + b + e_i$, this implies

$$\nabla_\theta \ell(\theta, x_i, y_i) = (2((\theta_1 - a)x_i + (\theta_2 - b) - e_i)x_i, 2((\theta_1 - a)x_i + (\theta_2 - b) - e_i))$$

For sufficiently large datasets Z where empirical loss approximates population loss, the gradient considered by Lip-DP-SGD will approximate

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\theta, Z) &\approx \sum_{e \in \{10,\}} P_e(e) \int_0^1 \nabla_\theta \ell(\theta, x, ax + b + e) \mathrm{d}x \\
&= \sum_{e \in \{10,\}} P_e(e) \int_0^1 (2((\theta_1 - a)x + (\theta_2 - b) - e)x, 2((\theta_1 - a)x + (\theta_2 - b) - e)) \, \mathrm{d}x \\
&= \int_0^1 \big(2((\theta_1 - a)x^2 + (\theta_2 - b)x - x\mathbb{E}[e]), 2((\theta_1 - a)x + (\theta_2 - b) - \mathbb{E}[e])\big) \, \mathrm{d}x \\
&= (2((\theta_1 - a)/3 + (\theta_2 - b)/2), 2((\theta_1 - a)/2 + (\theta_2 - b)))
\end{aligned}$$

This gradient becomes zero if $\theta_1 = a$ and $\theta_2 = b$ as intended.

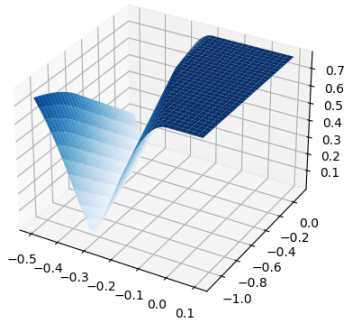
However, if we use gradient clipping with threshold $C = 1$ as in DP-SGD, we get:

$$\begin{aligned}
\tilde{g} &\approx \sum_{e \in \{10,\}} P_e(e) \int_0^1 clip_1 \left(\nabla_\theta \ell(\theta, x, ax + b + e)\right) \mathrm{d}x \\
&= \sum_{e \in \{10,\}} P_e(e) \int_0^1 clip_1 \left((2((\theta_1 - a)x + (\theta_2 - b) - e)x, 2((\theta_1 - a)x + (\theta_2 - b) - e))\right) \mathrm{d}x
\end{aligned}$$

While for a given e for part of the population $(\theta_1 - a)x + \theta_2 - b$ may be small, for a fraction of the instances the gradients are clipped. For the instances with $e = 9$ this effect is stronger. The result is that for $\theta_1 = a$ and $\theta_2 = b$ the average clipped gradient $\tilde{g}$ does not become zero anymore, in particular $\|\tilde{g}\| = 0.7791$. In fact, $\tilde{g}$ becomes zero for $\theta_1 = a{+}0.01765$ and $\theta_2 = b{+}0.94221$. Figure section 5.2.4 illustrates this situation.

Furthermore, [Chen et al., 2020] shows an example showing that gradient clipping can introduce bias. Hence, DP-SGD does not necessarily converge to a local optimum of $F(\theta, Z)$, even when sufficient data is available to estimate θ. While Lip-DP-SGD can only find models in $\Theta_{\leq C}$ and this may introduce another suboptimality, as our experiments will show this is only a minor drawback in practice, while also others observed that Lipschitz networks have good properties [Béthune et al., 2023]. Moreover, it is easy to check whether Lip-DP-SGD outputs parameters on the boundary of $\Theta_{\leq C}$ and hence the model could potentially improve by relaxing the weight norm constraint. In contrast, it may not be feasible to detect that DP-SGD is outputting potentially suboptimal parameters. Indeed, consider a federated learning setting (e.g.,

Figure 5.2.1: An example of gradient clipping causing bias, here the average gradient becomes zero at $(0, 0)$ while the average clipped gradient is 0 at another point, causing convergence of DP-SGD to that point rather than the correct one.



[Bonawitz et al., 2017]) where data owners collaborate to compute a model without revealing their data. Each data owner locally computes a gradient and clips it, and then the data owners securely aggregate their gradients and send the average gradient to a central party updating the model. In such setting, for privacy reasons no party would be able to evaluate that gradient clipping introduces a strong bias in some direction. Still, our experiments show that in practice at the time of convergence for the best hyperparameter values clipping is still active for a significant fraction of gradients (See Figure 5.3.3(a))

5.3 Experimental results

In this section, we conduct an empirical evaluation of our approach.

5.3.1 Experimental setup

We consider the following experimental questions:

- Q1 How does Lip-DP-SGD, our proposed technique, compare against the conventional DP-SGD as introduced by [Abadi et al., 2016]?
- Q2 What is the effect of allowing $\|\theta_k\| < C$ rather than normalizing $\|\theta_k\|$ to C? This question seems relevant given that some authors (e.g., [Béthune et al., 2023]) also suggest to consider networks which constant gradient norm rather than maximal gradient norm, i.e., roughly with θ in $\Theta_{=C}$ rather than $\Theta_{\leq C}$.

Norms. In this section, we focus on the $\ell_{2,2}$ norm. All subsequent results are based on this norm. Although computing this norm is more computationally intensive, it allows for the use of the Gaussian mechanism, which is standard in this field.

Hyperparameters. We selected a number of hyperparameters to tune for our experiments, aiming at making a fair comparison between the studied techniques while minimizing the distractions of potential orthogonal improvements. To optimize these hyperparameters, we used Bayesian optimization [Balandat et al., 2020].

In the literature, there are a wide range of improvements possible over a direct application of SGD to supervised learning, including general strategies such as pre-training, data augmentation and feature engineering, and DP-SGD specific optimizations such as adaptive maximum gradient norm thresholds. All of these can be applied in a similar way to both Lip-DP-SGD and DP-SGD and to keep our comparison sufficiently simple, fair and understandable we didn't consider the optimization of these choices.

We did tune hyperparameters inherent to specific model categories, in particular the initial learning rate $\eta(0)$ (to start the adaptive learning rate strategy $\eta(t)$) and (for image datasets) the number of groups, and hyperparameters related to the learning algorithm, in particular the (expected) batch size s, the Lipschitz upper bound of the normalization layer α and the threshold C on the gradient norm respectively weight norm.

The initial learning rate $\eta(0)$ is tuned while the following $\eta(t)$ are set adaptively. Specifically, we use the strategy of the Adam algorithm [Kingma and Ba, 2014], which update each parameter using the ratio between the moving average of the gradient (first moment) and the square root of the moving average of its squared value (second moment), ensuring fast convergence.

We also investigated varying the hyperparameter τ of the cross entropy objective function, but the effect of this hyperparameter turned out to be insignificant.

Both the clipping threshold C for gradients in DP-SGD and the clipping threshold C for weights in Lip-DP-SGD can be tuned for each layer separately. While this offers improved performance, it does come with the cost of consuming more of the privacy budget, and substantially increasing the dimensionality of the hyperparameter search space. In a few experiments we didn't see significant improvements in allowing per-layer varying of C_k, so we didn't further pursue this avenue.

Table 5.2 summarizes the search space of hyperparameters. It's important to note that we did not account for potential (small) privacy losses caused by hyperparameter search, a limitation also acknowledged in other recent works such as [Papernot and Steinke, 2022].

Table 5.2: Summary of hyperparameter space.

Hyperparameter	Range
Noise multiplier σ	[0.4, 5]
Weight clipping threshold C	[1, 15]
Gradient clipping threshold C	[1, 15]
Batch size s	[32, 512]
$\eta(0)$	[0.0001, 0.01]
Number of groups (group normalization)	[8, 32]
α (group normalization)	$[0.1/(\lvert x_k^{(k:q)}\rvert), 1/(\lvert x_k^{(k:q)}\rvert)]$

Datasets and models. We carried out experiments on both tabular datasets and datasets with image data. First, we consider a collection of 7 real-world tabular datasets (names and citations in Table 5.4). For these, we trained multi-layer perceptrons (MLP). A comprehensive list of model-dataset combinations is available in Table 5.3. To answer question Q2, we also implemented Fix-Lip-DP-SGD, a version of Lip-DP-SGD limited to networks whose weight norms are fixed, i.e., $\forall k : \|\theta_k\| = C$, obtained by setting $u_k^{(\theta)} \leftarrow C$ in Line 20 in Algorithm 8.

Second, the image datasets used include MNIST [Deng, 2012], Fashion-MNIST [Xiao et al., 2017], and CIFAR-10 [Krizhevsky et al., 2009]. We trained convolutional neural networks (CNNs) for the first two datasets and a Wide-ResNet [Zagoruyko and Komodakis, 2016] for the latter (see details in Table 5.3). We implemented all regularization techniques as described in [De et al., 2022], including group normalization [Wu and He, 2018], large batch size, weight standardization [Qiao et al., 2020], augmentation multiplicity [De et al., 2022], and parameter averaging [Polyak and Juditsky, 1992]. These techniques are compatible with Lipschitz-constrained networks, except for group normalization, for which we proposed an adapted version in Equation (5.2.1).
We opted for the accuracy to facilitate easy comparisons with prior research.

Table 5.3 shows details of the models we used to train on tabular and image datasets. We consider 7 tabular datasets: adult income [Becker et al., 1996], android permissions [Mathur et al., 2022], breast cancer [Wolberg et al., 1995], default credit [Yeh, 2016], dropout [Realinho et al., 2021], German credit [Hofmann, 1994] and nursery [Rajkovic, 1997]. See Table 5.4 for the number of instances and features for each tabular dataset.

Infrastructure. All experiments were orchestrated across dual Tesla P100 GPU platforms (12GB capacity), operating under CUDA version 10, with a 62GB RAM provision for Fashion-MNIST and CIFAR-10. Remaining experiments were performed on an E5-2696V2 Processor setup, equipped with 8 vCPUs and a 52GB RAM cache. The total runtime of the experiments was approximately 50 hours, which corresponds

Table 5.3: Summary table of datasets with respective models architectures details.

Dataset	Image size	Model	Loss	No. of Parameters
Tabular Datasets	-	MLP	CE	140 to 2,120
MNIST	28x28x1	ConvNet	CE	1,625,866
FashionMNIST	28x28x1	ConvNet	CE	1,625,866
CIFAR-10	32x32x3	WideResnet	CE	8,944,266

to an estimated carbon emission of 1.96 kg [Lacoste et al., 2019].

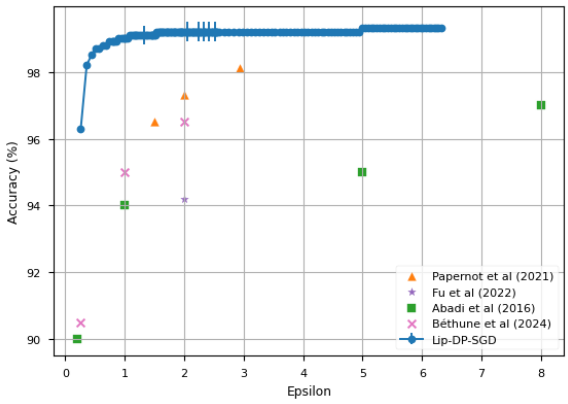
Table 5.4: Accuracy and ϵ per dataset and method at $\delta = 1/n$, in bold the best result and underlined when the difference with the best result is not statistically significant at a level of confidence of 5%.

Methods Datasets (#instances n × #features p)	ϵ	DP-SGD	Lip-DP-SGD	Fix-Lip-DP-SGD
Adult Income (48842x14) [Becker et al., 1996]	0.414	0.824	**0.831**	0.804
Android (29333x87) [Mathur et al., 2022]	1.273	0.951	**0.959**	0.945
Breast Cancer (569x32) [Wolberg et al., 1995]	1.672	0.773	**0.798**	0.7
Default Credit (30000x24) [Yeh, 2016]	1.442	0.809	**0.816**	0.792
Dropout (4424x36) [Realinho et al., 2021]	1.326	0.763	**0.819**	0.736
German Credit (1000x20) [Hofmann, 1994]	3.852	0.735	<u>0.746</u>	0.768
Nursery (12960x8) [Rajkovic, 1997]	1.432	0.919	**0.931**	0.89

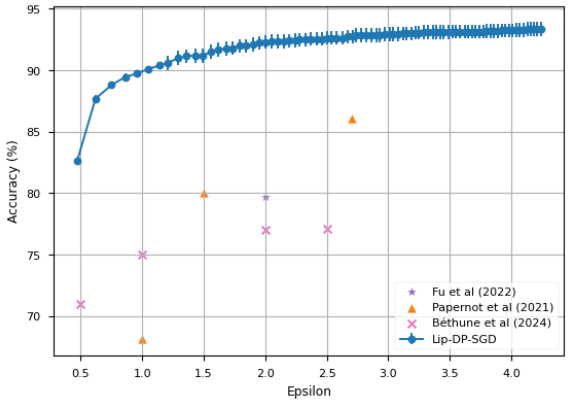
5.3.2 Results

Image datasets. In Figure 5.3.1, Lip-DP-SGD surpasses all state-of-the-art results on all three image datasets. Previous performances were based on DP-SGD as introduced by [Abadi et al., 2016], either combined with regularization techniques [De et al., 2022] or with bespoke activation functions [Papernot et al., 2021]. Note that while we present results using the same set of hyperparameters for MNIST and Fashion-MNIST, the results for CIFAR-10 come from a Pareto front of two sets of hyperparameters. For epsilon values below 4.2, the results come from a Wide-ResNet-16-4, and for epsilon values above 4.2, they come from a Wide-ResNet-40-4. See the complete list of hyperparameters in Table 5.2.

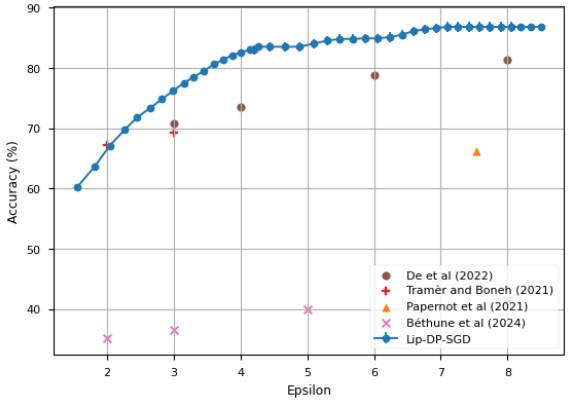
We also include the results reported by [Bethune et al., 2023], although their method does not achieve competitive accuracy when compared to state-of-the-art DP-SGD approaches. Notably, the performance gap between [Bethune et al., 2023] and Lip-DP-SGD widens as task complexity increases, for instance, in the CIFAR-10 benchmark. As discussed in Section 5.2, the upper bound on the gradient norm employed



(a) MNIST



(b) Fashion-MNIST



(c) CIFAR-10

Figure 5.3.1: Accuracy results, with a fixed $\delta = 10^{-5}$, for the MNIST (5.3.1(a)), Fashion-MNIST (5.3.1(b)), and CIFAR-10 (5.3.1(c)) test datasets. The plots show the median accuracy over 5 runs, with vertical lines indicating the standard error of the mean. See Table 5.2 for details on model specifications and hyperparameters.

in [Bethune et al., 2023] imposes limitations on model depth, restricting them to relatively shallow architectures. For example, their MLP-Mixer model used for CIFAR-10 consists of only 93,184 parameters, whereas our method supports deeper models such as Wide-ResNet-16-4, which contains 8,944,266 parameters.

Tabular datasets. In Table 5.4, we perform a Wilcoxon Signed-rank test, at a confidence level of 5%, on 10 measures of accuracy for each dataset between the DP-SGD based on the gradient clipping and the Lip-DP-SGD based on our method. Lip-DP-SGD consistently outperforms DP-SGD in terms of accuracy. This trend

holds across datasets with varying numbers of instances and features, including tasks with imbalanced datasets like Dropout or Default Credit datasets. While highly impactful for convolutional layers, group normalization does not yield improvements for either DP-SGD or Lip-DP-SGD in the case of tabular datasets.

Additionally, Table 5.4 presents the performance achieved by constraining networks to Lipschitz networks, where the norm of weights is set to a constant, denoted as Fix-Lip-DP-SGD. The results from this approach are inferior, even when compared to DP-SGD.

Conclusion. In summary, our experimental results demonstrate that Lip-DP-SGD sets new state-of-the-art benchmarks on the three most popular vision datasets, outperforming DP-SGD. Additionally, Lip-DP-SGD also outperforms DP-SGD on tabular datasets using MLPs, where it is advantageous to allow the norm of the weight vector θ to vary rather than normalizing it to a fixed value, leveraging situations where it can be smaller.

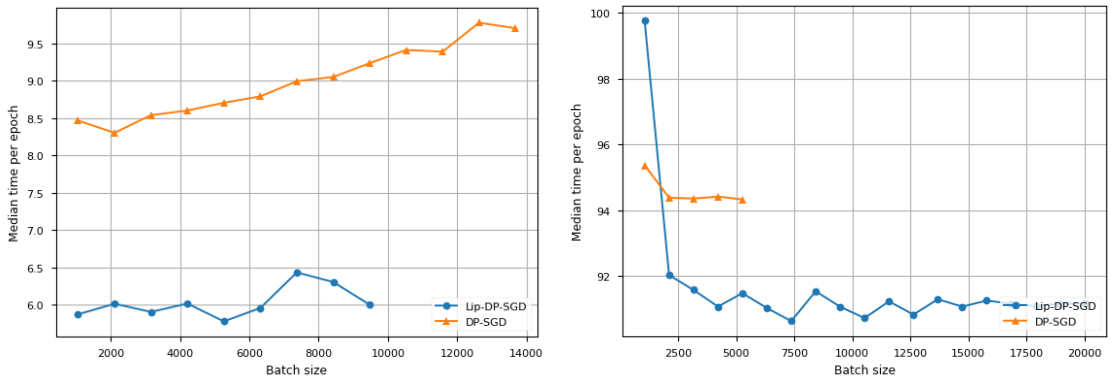
5.3.3 Runtime

Our experiments didn't show significant deviations from the normal runtime behavior one can expect for neural network training. As an illustration, we compared on the MNIST dataset and on the CIFAR-10 dataset the median epoch runtime of DP-SGD with Lip-DP-SGD. Both implementations utilize Opacus [Yousefpour et al., 2021] and PyTorch [Paszke et al., 2019], employing the same set of hyperparameters, such as augmentation multiplicity, to ensure a fair comparison. We measure runtime against the logical batch size, limiting the physical batch size to prevent memory errors as recommended by the PyTorch documentation [Paszke et al., 2019]. Figure 5.3.2 shows how Lip-DP-SGD is more efficient in terms of runtime compared to DP-SGD, especially for big batch sizes. This is mainly due to the fact that DP-SGD requires to clip the gradient at the sample level, slowering down the process.

5.3.4 Gradient clipping behavior

In Section 5.2.4 we argued that DP-SGD introduces bias. There are several ways to demonstrate this. For illustration we show here the error between the true average gradient

$$g_k^{\text{Lip-DP-SGD}} = \frac{1}{|V|} \sum_{i=1}^{|V|} \nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x_i))$$



(a) MNIST (b) CIFAR-10

Figure 5.3.2: Median runtime in seconds per batch size on one epoch over the MNIST dataset 5.3.2(a) and the CIFAR-10 dataset 5.3.2(b) comparing DP-SGD (in orange) and Lip-DP-SGD (in blue).

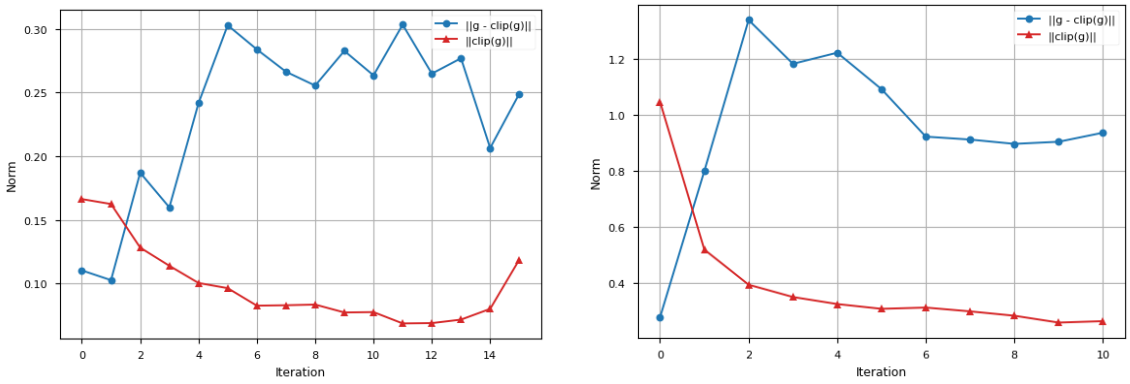
i.e., the model update of Algorithm 8 without noise, and the average clipped gradient

$$g_k^{\text{DP-SGD}} = \frac{1}{|V|} \sum_{i=1}^{|V|} \text{clip}_C \left(\nabla_{\tilde{\theta}_k} \ell(f_{\tilde{\theta}}(x_i)) \right),$$

i.e., the model update of Algorithm 7 without noise.

Figure 5.3.3 shows the error $\left\| g_k^{\text{Lip-DP-SGD}} - g_k^{\text{DP-SGD}} \right\|$ together with the norm of the DP-SGD model update $\left\| g_k^{\text{DP-SGD}} \right\|$.

One can observe for both considered datasets that while the model converges and the average clipped gradient decreases, the error between DP-SGD's average clipped gradient and the true average gradient increases. At the end, the error in the gradient caused by clipping is significant, and hence the model converges to a different point than the real optimum.



(a) Dropout (b) Adult Income

Figure 5.3.3: Norm of the average error $g - \text{clip}(g)$ (in blue) and norm of the average of $\text{clip}(g)$ (in red) across training iterations on the Dropout dataset 5.3.3(a) (averaged over 500 instances) and the Adult Income dataset 5.3.3(b) (averaged over 500 instances).

5.4 Related Work

DP-SGD. DP-SGD algorithms have been developped to guarantee privacy on the final output [Chaudhuri et al., 2011], on the loss function [Kifer et al., 2012] or on the publishing of each gradient used in the descent [Bassily et al., 2014, Abadi et al., 2016].

To keep track of the privacy budget consumption, [Bassily et al., 2014] relies on the strong composition theorem [Dwork et al., 2010b] while [Abadi et al., 2016] is based on the moment accountant and gives much tighter bounds on the privacy loss than [Bassily et al., 2014].

This has opened an active field of research that builds upon [Abadi et al., 2016] in order to provide better estimation of the hyperparameters e.g., the clipping norm [McMahan et al., 2017, Andrew et al., 2022], the learning rate [Koskela and Honkela, 2020], or the step size of the privacy budget consumption [Lee and Kifer, 2018, Chen and Lee, 2020, Yu et al., 2019]; or to enhance performance with regularization techniques [De et al., 2022]. Gradient clipping remains the standard approach, and most of these ideas can be combined with our improvements.

Lipschitz continuity. Lipschitz continuity is an essential requirement for differential privacy in some private SGD algorithms [Bassily et al., 2014]. However, since deep neural networks (DNNs) have an unbounded Lipschitz value [Scaman and Virmaux, 2019], it is not possible to use it to scale the added noise. Several techniques have been proposed to enforce Lipschitz continuity to DNNs, especially in the context of generative adversarial networks (GANs) [Miyato et al., 2018, Gouk et al., 2020]. These

techniques, which mainly rely on weight clipping, can be applied to build DP-SGD instead of the gradient clipping method, as described in Section 5.2 discusses how [Bethune et al., 2023] proposes several concepts related to our Fix-Lip-DP-SGD variant. However, their use of an upper bound on the input norm limits their ability to achieve competitive results and extend beyond the 2-norm. In contrast, we demonstrate that Lip-DP-SGD outperforms Fix-Lip-DP-SGD (Table 5.4), showcases the strong synergy between weight normalization and regularization techniques (see Figure 5.3.1), enables the theoretical use of the $\ell_{2,1}$ and the $\ell_{2,\infty}$-norms, and illustrates how the scaled gradient (Equation (5.2.14)) we employ unlocks high-accuracy for differentially private machine learning.

5.5 Conclusion

In this chapter we proposed a new differentially private stochastic gradient descent algorithm without gradient clipping. We derived a methodology to scale the gradient and estimate the scaled gradient sensitivity, which is a key component of our algorithm. An important advantage of weight clipping over gradient clipping is that it avoids the bias introduced by gradient clipping and the algorithm converges to a local optimum of the objective function. We showed empirically that this yields a significant improvement in practice and we argued that this approach circumvent the bias induced by classical gradient clipping.

Chapter 6

Differentially Private empirical cumulative distribution function

The need to balance learning and protecting sensitive training data has led to increasing interest in privacy-preserving machine learning techniques. In this chapter, we focus on the federated setting, where multiple parties, each holding their own training instances, aim to collaborate on learning a model without exposing their individual data.

We introduce strategies for computing differentially private empirical distribution functions. Although revealing full functions may require a higher privacy budget, it offers richer and more valuable insights for the learner. We establish formal privacy guarantees and analyze the computational costs for both a general-purpose strategy compatible with any security model and a specialized approach based on secret sharing. Additionally, we explore a variety of applications and validate our methods through experimental results.

The code corresponding to this chapter is available at https://gitlab.inria.fr/abarczew/ab_technical/-/tree/master/medical-statistics-and-privacy/dp-cum/code/2024.

6.1 Introduction

We have seen in Chapter 3 how Differential Privacy (DP) has become the gold standard to measure the privacy of an algorithm. For a wide range of machine learning strategies, versions have been proposed which output models with an appropriate amount of noise to satisfy differential privacy. A setting of particular interest is the federated learning setting where there is a large number of parties which each own one or more training instances and which want to jointly compute a statistic or model

without revealing their own data.

In this chapter we are interested in empirical cumulative distribution functions (ECDFs). Consider a setting where every party in a group has one (or more) values. Then, the empirical distribution function returns for every threshold the number of values which are smaller than that threshold. Cumulative distribution functions play an important role in statistics and machine learning, e.g., in tail bounds and in statistics based on rankings. An ECDF which is often used for machine learning validation is the receiver-operator characteristic (ROC) curve. Often an ECDF is an intermediate result from which a single number is computed, e.g., the area under the ROC curve is an important metric capturing the behavior of a classifier in a single number. Nevertheless, knowledge of the complete ECDF is in many cases an asset of its own value, e.g., for a ROC curve depending on the cost structure of the problem at hand the beginning or end of the curve may be the most interesting.

We propose differentially private ECDFs. For a fixed privacy level, publishing a complete ECDF requires more noise than publishing a single aggregated value, but is also much more informative. Moreover, it turns out that the amount of noise to be added only needs to be logarithmic in the required precision, which is better than pointwise adding independent noise to ECDF evaluations. Our work here is inspired by but also improves on earlier work on continual observation of statistics in data streams, which was initiated using ϵ-differential privacy by [Dwork et al., 2010a] and later extended towards among others indefinite time periods and other privacy notions such as Renyi differential privacy [Cardoso and Rogers, 2022]. Our work is to some extent orthogonal to these extensions: for simplicity of explanation we will adopt the classic ϵ-differential privacy, but our work can be combined with several of these other ideas.

We will also investigate federated algorithms to compute differentially private ECDFs. In particular, we are interested in both evaluating ECDFs and evaluating inverse ECDFs. We explore two avenues. First, we investigate an approach starting from a secure aggregation protocol as a building block. The advantage of such approach is that it inherits the security guarantees of the aggregation protocol. One can plug in an aggregator which is cheaper but assumes parties are honest-but-curious, or one can plug in an aggregator which is more expensive but robust against malicious behavior. The result then is a federated algorithm to compute an ECDF having the same security guarantees. We hope to contribute in this way to an evolution to a more modular organization of building blocks where there is no need for a different algorithm for every different problem and security setting. Second, we will investigate strategies to compute ECDF evaluations more efficiently compared to pointwise federated evaluations, at the cost of making some assumptions on the aggregation protocol used.

In summary, our contributions are:

- We propose a strategy for generating differentially privately a complete ECDF. We prove that the resulting curve is ϵ-differentially private.
- In the process of doing so, we make an improvement of a factor 2 over differential privacy guarantees for continually observed statistics as shown in [Cardoso and Rogers, 2022].
- As a differentially private version of a non-decreasing function is not necessarily non-decreasing itself, we propose a strategy to smooth differentially private functions. This makes the function non-decreasing again, and in some cases may even reduce the error induced by the differential privacy noise.
- We discuss strategies for efficient implementations. In particular, we propose both a generic algorithm which can start from any secure aggregation operator and inherit its security/privacy model and a specific strategy strategy based on secret sharing which has asymptotically better complexity.
- We present in more detail two applications. First, we discuss how to make ϵ-differentially private ROC curves using our technique. Second, we present a differentially private Hosmer-Lemeshow statistic.
- We illustrate our techniques with a number of experiments.

The remainder of this chapter is organized as follows. In Section 6.2 we introduce basic concepts and notations. Next, in Section 6.3 we present differentially private ECDFs and prove the corresponding privacy guarantees and in Section 6.4 we discuss federated algorithms to compute differentially private ECDFs. In Section 6.6 we present experiments illustrating our approach. Finally, in Section 6.7 we conclude and outline future work.

6.2 Preliminaries

6.2.1 Notation

We denote by $[m, n] = \{z \in \mathcal{Z} \mid m \leq z \leq n\}$ the set of integers between and including m and n, and by $[n] = [1, n]$ the set of the first n positive integers. For a boolean expresseion b, we denote by $\mathbb{I}[b]$ its truth value, i.e., $\mathbb{I}[true] = 1$ and $\mathbb{I}[false] = 0$. For a set X, we denote its indicator function by $\mathbb{1}_X$, i.e., $\mathbb{1}_X(x) = \mathbb{I}[x \in X]$.

We consider datasets $X = \{x_i\}_{i=1}^n \in \mathcal{X}^n$ where $\mathcal{X}$ is a space of instances and $n \in \mathbb{N}$ is a positive integer. We assume there are n parties P_i with $i \in [n]$, each owning

one instance x_i. Our results will generalize easily to the case where every party may own multiple instances. We assume the instances x_i include sensitive data and the parties do not want to reveal them. Still, the parties want to collaborate to compute statistics of common interest.

Definition 6.2.1 (U-statistic)**.** *Let $m \geq 1$ be a positive integer and let $\phi : \mathcal{X}^m \to \mathbb{R}$ be a symmetric function. The U-statistic with kernel ϕ is the function mapping samples $\{x_i\}_{i=1}^n \in \mathcal{X}^n$ on*

$$U_\phi(X) = \binom{n}{m}^{-1} \sum_{1 \leq i_1 < \cdots < i_r \leq n} \phi(x_{i_1}, \ldots, x_{i_m}). \tag{6.2.1}$$

We say U_ϕ is a U-statistic of order m.

Of particular interest are U-statistics of order 1, which are averages of the form $U_\phi(X) = (1/n) \sum_{i=1}^n \phi(x_i)$.

Let $\eta = (\eta_i)_{i \in \mathcal{I}}$ be a vector of random variables for some set $\mathcal{I}$ of indices. Then, to compute differentially private statistics it will be convenient to define

$$\hat{U}_\phi(X, I) = U_\phi(X) + \sum_{i \in I} \eta_i$$

Many strategies have been proposed to compute such averages with higher or lower security and privacy guarantees. The simplest but least secure is the classical trusted curator setting where all input is sent to a trusted party which makes the average and sends it back. Strategies such as [Shi et al., 2011, Bonawitz et al., 2017, Chan et al., 2012] focus on securely computing an average without disclosing the data to any party, assuming a honest-but-curious setting, i.e., parties are assumed to follow the protocol honestly even if they are curious and may try to infer information from what they observe. The strategy proposed in [Dwork et al., 2006a] aim at better verifiability, but induces a cost quadratic in the number of parties. The algorithms in [Jayaraman et al., 2018, Sabater and Ramon, 2021] integrate more strongly noise addition in the secure aggregation step, but relies on the additional assumption that two servers do not collude. Recently, the shuffle model of privacy [Cheu et al., 2019, Erlingsson et al., 2019, Hartmann and West, 2019, Balle et al., 2020, Ghazi et al., 2020] has been studied, where inputs are passed via a trusted/secure shuffler that obfuscates the source of the messages, leading to an alternative trust model. It is also possible to distribute trust over the participating parties rather than relying on a limited number of servers for secret keeping [Sabater et al., 2022].

We will here simply assume the existence of an operation $UStat(\phi, X, \eta : I)$ that computes and publishes $\hat{U}_\phi(X, I)$ in an appropriate way compliant with the considered

attack model. Accordingly, the cost of $UStat$ will depend on the strategy used, e.g., if the parties are assumed to be honest-but-curious the cost may be lower than if the method needs to be robust against certain malicious behavior. We assume the operation $UStat$ has the necessary facilities to draw, store and keep secret value of the random variables in η. As our generic algorithm will only use U-statistics and will perform subsequent calculations in the open, our approach will inherit its attack model directly from $UStat$.

Definition 6.2.2 (Empirical cumulative distribution function)**.** *Let $\phi : \mathcal{X} \to \mathbb{R}$. Given a sample X, the empirical cumulative distribution function (ECDF) of ϕ, denoted by F_ϕ, is the function $F_\phi(X;\cdot) : \mathbb{R} \to [0,1]$ with*

$$F_\phi(X,t) = \frac{1}{n}\left|\{i \in [n] \mid \phi(x_i) \leq t\}\right|$$

6.2.2 Related work

Several works have addressed the problem of estimating histograms and quantiles under differential privacy. Notably, [Lei, 2011] and [Alabi et al., 2023] propose efficient methods for differentially private estimation of histograms and quantiles on large datasets. While effective in their settings, these methods are not designed for scenarios involving repeated queries or the release of full functional outputs, which limits their applicability in such contexts.

In contrast, our approach estimates quantiles by applying an inverse functional to a DP-ECDF. A related approach is introduced in [Drechsler et al., 2022], where a private ECDF is used to construct confidence intervals for the median. However, their method relies on a tree-based mechanism ([Dwork et al., 2010a]), which introduces a larger noise budget by requiring twice as many noise terms compared to our technique.

Furthermore, our work incorporates a novel compliance analysis with respect to attack models that inherits its attack model directly from an operation of U-statistics. Although [Chaudhuri et al., 2024] present a general framework for the private estimation of U-statistics—including both degenerate and non-degenerate cases—they do not address the challenge posed by repeated querying.

6.3 Method

6.3.1 Private ECDF

Let $\phi : \mathcal{X} \to \mathbb{R}$. Let $\phi^{\min} = \min_{i\in[n]} \phi(x_i)$ and $\phi^{\max} = \max_{i\in[n]} \phi(x_i)$. Let $N \in \mathbb{N}$ and let $\tau \in \mathbb{R}^{[N]}$ be an ordered set of points on which one may want to evaluate F_ϕ, e.g., one may take $\tau = \{t \in \psi\mathbb{Z} \mid \phi^{\min} \leq t \leq \phi^{\max}\}$ for some precision parameter

ψ, or if the relative error is more important than the absolute error one may take $\tau = \{e^{\psi z} \mid z \in \mathbb{Z} \wedge \phi^{\min} \leq e^{\psi z} \leq \phi^{\max}\}$. We assume that $\tau_1 = \phi^{\min}$ and $\tau_N = \phi^{\max}$.

Let $L = \lceil \log_2(N) \rceil$. Let $\mathcal{I}[L] = \{(j,l) \mid l \in [0,L] \wedge j \in [\lceil 2^{L-l} \rceil]\}$. Let $\eta = (\eta_{j,l})_{(j,l)\in\mathcal{I}[L]}$ be a set of random variables with $\eta_{j,l} \sim Lap((L+1)/\epsilon)$ for $(j;l) \in \mathcal{I}[L]$. Then, we define the function $\hat{F}_\phi$ by

$$\hat{F}_\phi(X, \tau_i) = F_\phi(X, \tau_i) + \frac{1}{|X|} \sum_{l=0}^{L} \eta_{\lceil i/2^l \rceil, l} \tag{6.3.1}$$

While it is possible to reduce the errors obtained later by up to 15% by using a base different from 2 following [Cardoso and Rogers, 2022], using base 2 simplifies our explanation and the later algorithms.

Theorem 6.3.1. *Publishing $\hat{F}_\phi(X, \tau_i)$ for all $i \in [N]$ is ϵ-DP. The expected squared error is $\mathbb{E}[(F_\phi(x) - \hat{F}_\phi(x))^2] = 2(L+1)^3/\epsilon^2$.*

The proof follows to a large extent the ideas from [Dwork et al., 2010a, Cardoso and Rogers, 2022], but achieves a better result by explaining the difference between outputs for adjacent datasets by not only positive but also negative changes in the noise terms. Our ideas also allow for improving Theorem 1 in [Cardoso and Rogers, 2022] on differentially private continual observation of statistics in data streams, we provide more details in the following proof.

Before proving Theorem 6.3.1, we first introduce some additional definitions and lemmas. For $L \in \mathbb{N} \setminus \{0\}$, let

$$Z_{L,d} = \left(\mathbb{1}_{[d+(j-1)2^l+1, d+j2^l]}\right)_{(j,l)\in\mathcal{I}[L]}$$

be a vector of functions indexed by $\mathcal{I}[L]$ where $\mathbb{1}_X : \mathbb{N} \to \{0,1\}$ is a function with $\forall x \in X : \mathbb{1}_X(x) = 1$ and $\forall x \in \mathbb{N} \setminus X : \mathbb{1}_X(x) = 0$. Note that $[d, d-1] = \emptyset$ and hence $\mathbb{1}_{[d,d-1]} = 0$.

Lemma 6.3.1. *Let $L \in \mathbb{N} \setminus \{0\}$, $d \in \mathbb{N}$ and $b - d \in \left[2^L\right]$, then there exist vectors $\chi^s \in \{-1,0,1\}^{\mathcal{I}[L]}$, $s \in \{+,-\}$, with the number of non-zero elements $\|\chi^s\|_0$ at most $\lceil (L+1)/2 \rceil$ such that $\mathbb{1}_{[d+1,b]} = \chi^-.Z_{L,d}$ and $\mathbb{1}_{[b,d+2^L]} = \chi^+.Z_{L,d}$.*

Proof. We need to prove that we can write the functions $\mathbb{1}_{[d+1,b]}$ and $\mathbb{1}_{[b,d+2^L]}$ as weighted sums (with coefficients -1 or $+1$) of elements of $Z_{L,d}$. We proceed by induction.

Base cases. Consider first $L = 0$ and $L = 1$. In both cases it is easy to verify that $\mathbb{1}_{[d+1,b]}$ and $\mathbb{1}_{[b,d+2^L]}$ are both elements of $Z_{2,d}$ and hence χ^+ and χ^- can be set to suitable base vectors, i.e., $\|\chi^+\|_0 = \|\chi^-\|_0 = 1$.

Induction step. Suppose that $L \geq 2$ and assume that the lemma has been proven for $L' \leq L-2$. There are four cases, depending on $b' = \lceil (b-d)/2^{L-2} \rceil \in [4]$.

- Consider first $b' = 0$, i.e., $b - d \leq 2^{L-2}$:

  (χ^-) Applying the lemma for $L' = L-2$ we can write $\mathbb{1}_{[d+1,b]}$ as a weighted sum of $\lceil (L'+1)/2 \rceil < \lceil (L+1)/2 \rceil$ elements of $Z_{L',d} \subseteq Z_{L,d}$.

  (χ^+) As $\mathbb{1}_{[b,d+2^L]} = \mathbb{1}_{[d+1,d+2^L]} - \mathbb{1}_{[d+1,b-1]}$, we can write $\mathbb{1}_{[b,d+2^L]}$ as a weighted sum of at most $\lceil (L+1)/2 \rceil$ elements of $Z_{L,d}$: $\mathbb{1}_{[d+1,d+2^L]} \in Z_{L,d}$ and either $\mathbb{1}_{[d+1,b-1]} = 0$ or $b - 1 \in [2^{L-2}]$ implying that $\mathbb{1}_{[d+1,b-1]}$ as in case (χ^-) above.

- Consider next $b' = 1$, i.e., $2^{L-2} + 1 \leq b - d \leq 2^{L-1}$.

  χ^- As $\mathbb{1}_{[d+1,b]} = \mathbb{1}_{[d+1,d+2^{L-2}]} + \mathbb{1}_{[(d+2^{L-2})+1,b]}$, $\mathbb{1}_{[b,d+2^L]} \in Z_{L,d}$ it suffices to apply the induction hypothesis and note that we can write $\mathbb{1}_{[(d+2^{L-2})+1,b]}$ as a sum of $\lceil (L+1)/2 \rceil - 1$ elements of $Z_{L-2,d+2^{L-2}} \subseteq Z_{L,d}$.

  χ^+ $\mathbb{1}_{[b,d+2^L]} = \mathbb{1}_{[b,d+2^{L-1}]} + \mathbb{1}_{[d+2^{L-1}+1,d+2^L]}$ and $\mathbb{1}_{[b,d+2^{L-1}]} \in Z_{L,d}$ hence it suffices to apply the induction hypothesis to $\mathbb{1}_{[d+2^{L-1}+1,d+2^L]}$.

- Cases $b' = 2$ and $b' = 3$ are analoguous to cases $b' = 1$ and $b' = 0$ respectively.

This completes the proof. □

Theorem 6.3.1 Publishing $\hat{F}_\phi(X, \tau_i)$ for all $i \in [N]$ is ϵ-DP. The expected squared error is $\mathbb{E}[(F_\phi(x) - \hat{F}_\phi(x))^2] = 2(L+1)^3/\epsilon^2$.

Proof. Consider two adjacent datasets $X^{(1)}$ and $X^{(2)}$. As the datasets are adjacent, they differ in only one instance, i.e., there exists a dataset X' and instances $x_\Delta^{(1)}$ and $x_\Delta^{(2)}$ such that $X^{(s)} = X' \cup \{x_\Delta^{(s)}\}$ for $s \in \{1, 2\}$.

The inner product of $\mathcal{I}[L]$-indexed vectors $\eta Z_{L,0}$ is a function mapping any $i \in [2^L]$ to

$$\sum_{(j,l)} \{\eta_{j,l} \mid (j-1)2^l + 1 \leq i \leq j2^l\} = \sum_{l=0}^{L} \eta_{\lceil i/2^l \rceil, l}.$$

Then, defining $F_\phi(X, \tau) = (F_\phi(X, \tau_i))_{i \in [2^L]}$ and $\hat{F}_\phi(X, \tau) = \left(\hat{F}_\phi(X, \tau_i)\right)_{i \in [2^L]}$, where we set $\forall i \in [N+1, 2^L] : \tau_i = \phi^{\max}$, we can rewrite Eq (6.3.1) as

$$\hat{F}_\phi(X, \tau) = F_\phi(X, \tau) + \frac{\eta Z_{L,0}}{n} \tag{6.3.2}$$

For $s \in 1, 2$, there holds

$$nF_\phi(X^{(s)}, \tau) - (n-1)F_\phi(X^{(s)}, \tau) = \mathbb{1}_{[t_s+1, 2^L]} \tag{6.3.3}$$

where $t_s = \max\left\{i \in [N] \mid x_\Delta^{(s)} > \tau_i\right\}$. Without loss of generality we assume that $X_\Delta^{(2)} < x_\Delta^{(1)}$. Combining Eq (6.3.2) and twice Eq (6.3.3) we get

$$n\hat{F}_\phi(X^{(2)}, \tau) = nF_\phi(X^{(1)}, \tau) + \mathbb{1}_{[t_1+1, t_2]} + \eta Z_{L,0} \tag{6.3.4}$$

Consider the largest $l_\mathrm{m} \in [0, L]$ for which there exists a $j_\mathrm{m} \in \mathbb{N}$ such that $t_1 \leq j_\mathrm{m} 2^{l_\mathrm{m}} < t_2$. Notice that j_m is odd, as else we would have $(j_\mathrm{m}/2)2^{l_\mathrm{m}+1} = j_\mathrm{m} 2^{l_\mathrm{m}}$ and l_m would not be maximal. Also, there holds $(j_\mathrm{m} - 1)2^{l_\mathrm{m}} < t_1$ as else we would have $t_1 \leq (j_\mathrm{m} - 1)2^{l_\mathrm{m}}$ with $j_\mathrm{m} - 1$ even and again l_m would not be maximal. Similarly we can infer $t_2 \leq (j_\mathrm{m} + 1)2^{l_\mathrm{m}}$. As $t_2 \leq 2^L$ there follows $l_\mathrm{m} \leq L - 1$. Let $d_1 = (j_\mathrm{m} - 1)2^{l_\mathrm{m}}$ and $d_2 = j_\mathrm{m} 2^{l_\mathrm{m}}$. Applying twice Lemma 6.3.1 we can conclude there exist vectors $\chi_1^+, \chi_2^- \in \{-1, 0, +1\}^{\mathcal{I}[l_\mathrm{m}]}$ with $\|\chi_1^+\|_0 \leq \lceil L/2 \rceil$ and $\|\chi_2^-\|_0 \leq \lceil L/2 \rceil$ such that $\chi_1^+ . Z_{l_\mathrm{m}, d_1} = \mathbb{1}_{[t_1+1, d_2]}$ and $\chi_2^- . Z_{l_\mathrm{m}, d_2} = \mathbb{1}_{[d_2+1, t_2]}$. As the elements of the vectors Z_{l_m, d_1} and Z_{l_m, d_2} also occur in $Z_{L,0}$, we can conclude that there exists a vector $\chi \in \{-1, 0, +1\}^{\mathcal{I}[L]}$ with $\|\chi\|_0 \leq L + 1$ such that

$$\chi . Z_{L,0} = \mathbb{1}_{[t_1+1, t_2]}. \tag{6.3.5}$$

We now express the probability of observing $\hat{F}_\phi(X^{(2)})$ given $X^{(2)}$ and compare it the probability of making the same observation given $X^{(1)}$. Using Eqs (6.3.4) and (6.3.5) and setting $\Gamma(y, X^{(1)}) = y - n.\hat{F}_\phi(X^{(1)}, \tau)$,

$$\begin{aligned}
&P\left(n.\hat{F}_\phi(X^{(2)}, \tau) = y\right) \\
&= P\left(nF_\phi(X^{(1)}, \tau) + \mathbb{1}_{[t_1+1, t_2]} + \eta Z_{L,0} = y\right) \\
&= P\left(\chi Z_{L,0} + \eta Z_{L,0} = \Gamma(y, X')\right) \\
&= \int_u P(\eta = u)\, \mathbb{I}\left[\chi Z_{L,0} + \eta Z_{L,0} = \Gamma(y, X')\right] \\
&= \int_u P(\eta = u + \chi)\, \mathbb{I}\left[\eta Z_{L,0} = \Gamma(y, X')\right]
\end{aligned}$$

Also,

$$\begin{aligned}
&P\left(n.\hat{F}_\phi(X^{(2)}, \tau) = y\right) \\
&= \int_u P(\eta = u)\, \mathbb{I}\left[\eta Z_{L,0} = \Gamma(y, X')\right]
\end{aligned}$$

The probability ratio for a given η is

$$\begin{aligned}
&\left|\log\left(\frac{P(\eta = u + \chi)}{P(\eta = u)}\right)\right| \\
&= \left|\sum_{(j,l)} \log\left(\frac{P(\eta_{j,l} = u_{j,l} + \chi_{j,l})}{P(\eta_{j,l} = u_{j,l})}\right)\right| \\
&\leq \sum_{(j,l):\chi_{j,l}\neq 0} \left|\log\left(\frac{P(\eta_{j,l} = u_{j,l} + \chi_{j,l})}{P(\eta_{j,l} = u_{j,l})}\right)\right| \\
&\leq (L+1)\frac{\epsilon}{L+1} \\
&= \epsilon
\end{aligned}$$

We can conclude

$$\left|\log\left(\frac{P\left(n.\hat{F}_\phi(X^{(2)}, \tau) = y\right)}{P\left(n.\hat{F}_\phi(X^{(2)}, \tau) = y\right)}\right)\right| \leq \epsilon$$

as both probabilities are integrals over functions who only differ by a factor e^ϵ. This proves the privacy guarantee.

The expected squared error made by adding the noise is

$$\begin{aligned}
&\mathbb{E}\left[\left(\hat{F}_\phi(x) - F_\phi(x)\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum_{l=0}^{L} \eta_{\lceil i/2^l \rceil, l}\right)^2\right] \\
&= \sum_{l=0}^{L} \text{var}\left(\eta_{\lceil i/2^l \rceil, l}\right) \\
&= (L+1).2\left(\frac{L+1}{\epsilon}\right)^2 \\
&= 2(L+1)^3/\epsilon^2
\end{aligned}$$

□

The above result has some implications for the more commonly studied problem of releasing statistics under continual observation [Cardoso and Rogers, 2022]. In this problem, one considers data streams $(x_i)_{i=1}^N$ and wants to report at any time step t the partial sum $s_t = \sum_{i=1}^t x_i$. Two data streams are considered adjacent if they differ at most in one time step. A change at a time step t^* changes all partial sums

between t^* and N, i.e., the sums in a half-open interval. In our setting, we also considered datasets adjacent if an instance changes its value, which means the partial sums change between its old value and its new value, i.e., the sums in a closed interval. Applying our technique above to this problem, we get the following results:

Theorem 6.3.2. *Let datasets $X^{(1)}, X^{(2)} \in \mathcal{X}^N$ be adjacent if there exists at most one i such that $X_i^{(1)} \neq X_i^{(2)}$. Let $L = \lceil \log_2(N) \rceil$. Then, publishing $\hat{s}_t = \sum_{i=1}^{t} x_i + \sum_{l=0}^{L} \eta_{\lceil i/2^l \rceil, l}$ where $\eta_{j,l} \sim Lap(\lceil (L+1)/2 \rceil / \epsilon$ is ϵ-differentially private. Similarly, with $\eta_{j,i} \sim \mathcal{N}(0, \lceil (L+1)/2 \rceil z^2)$ the publishing is $1/2z^2$-zCDP (as defined in [Cardoso and Rogers, 2022]).*

Proof. The proof is a direct application of Lemma 6.3.1 using similar ideas as in the proof of Theorem 6.3.1. □

The proof of Theorem 1 in [Cardoso and Rogers, 2022] concludes that at most L terms in the partial sums they disclose will change between adjacent datasets, and hence need to compose L differential private mechanisms. Even though they focus on Renyi differential privacy rather than ϵ-differential privacy, our idea allows in their setting too to reduce the number of changed terms with about a factor 2 (when using base 2) and hence to improve the privacy guarantee.

6.3.2 Smooth DP ECDF

While we know that F_ϕ is a non-decreasing function, due to the noise addition this does not hold anymore for $\hat{F}_\phi$. We can correct this problem by finding the non-decreasing function $\acute{F}_\phi$ which minimizes $\mathcal{L}(\hat{F}_\phi, \acute{F}_\phi)$ for an appropriate loss function $\mathcal{L}$. As N may be large, in practice we may only be interested in finding an appropriate $\acute{F}_\phi$ for a limited set of points $(\tau_i)_{i \in B}$ with $B \subseteq [N]$.

In particular, we define $\acute{F}_\phi$ the following optimization problem:

$$\begin{aligned} &\text{minimize} \sum_{(i,j) \in \mathcal{I}[L]} \nu_{i,j}^p \\ &\text{s.t. } \forall i \in B : \acute{F}_\phi(X, \tau_i) = \hat{F}_\phi(X, \tau_i) + \sum_{l=0}^{L} \nu_{\lceil i/2^l \rceil, l} \\ &\quad \acute{F}_\phi(X, \min(B)) \geq 0 \\ &\quad \acute{F}_\phi(X, \max(B)) \leq 1 \\ &\quad \forall i, j \in B : i < j \Rightarrow \acute{F}_\phi(X, \tau_i) \leq \acute{F}_\phi(\tau_j) \end{aligned} \tag{6.3.6}$$

Both $p = 1$ and $p = 2$ are plausible here. As the loglikelihood of a vector of Laplace noise variables is proportional to its 1-norm, $p = 1$ may be appealing. Still, we also will see a number of applications where the 2-norm (i.e., $p = 2$) is more appropriate.

Computing $\acute{F}$ from $\hat{F}$ is a post-processing step after achieving differential privacy, so it does not reduce the privacy guarantee. On the other hand, it makes later processing

requiring a non-decreasing function possible and may reduce the error induced by the DP noise (see Section 6.6.3).

6.4 Algorithms

In this section we will investigate algorithms to compute ECDFs and their inverse. First, observe that we can write an ECDF evaluation as a U-statistic evaluation:

$$F_\phi(X, t) = U_{\phi_{\leq t}}(X)$$

where $\phi_{\leq t}(x) = \mathbb{I}[\phi(x) \leq t]$. In other words, if we want to evaluate $F_\phi(X, t)$, we ask for each instance $x \in X$ whether $\phi(x)$ is smaller than t, and then count the positive answers. Similarly, for the differential private version we can write

$$\hat{F}_\phi(X, \tau_i) = \hat{U}_{\phi_{\leq \tau_i}}(X, \mathcal{I}[L, i]) \tag{6.4.1}$$

where $\mathcal{I}[L, i] = \{(\lceil i/2^l \rceil, l) \mid l \in [0, L]\}$.

6.4.1 Pointwise evaluation

Starting from any secure aggregation protocol implementing $\hat{U}_\cdot(\cdot, \cdot)$, Eq (6.4.1) allows for evaluating $\hat{F}_\phi$ on a number τ_i with $i \in [N]$ with the same security and privacy guarantees as that basic secure aggregation protocol. If one wants to evaluate $\hat{F}_\phi$ on a set of values $\{\tau_i\}_{i \in B}$ with $B \subseteq [N]$, then we can just repeat the secure aggregation protocol. In each iteration all parties must participate, which implies the communication cost is $O(n|B|)$.

6.4.2 Function secret sharing

While this scheme is generic as it allows for any secure aggregation protocol, it is possible under particular attack models to improve on its complexity. In particular, while protocols based on homomorphic encryption, multi-party computing or secret sharing may allow to aggregate values without revealing them, in their basic form they need a contribution of (and hence communication with) the parties storing the data in cleartext for each new query which must be answered, in our case for each x on which we want to evaluate $\hat{F}_\phi(x)$ using a secure aggregation. Recently, function secret sharing (FSS) (Section 4.2.1) techniques were proposed which allow to secretly share complete functions. Here, we will use FSS for comparison functions, which was proposed in [Boyle et al., 2016].

Function secret sharing protocols consist of two operations: *gen* and *eval*. Given a function f from the appropriate class, $gen(f)$ returns a set of $m \geq 2$ keys $(k^{(1)} \dots k^{(m)})$.

These keys are indistinguishable (for algorithms running in polynomial time) from randomly drawn keys and can therefore be distributed over servers which are trusted to not collude. To evaluate the function on some input x, the server who received $k^{(j)}$ ($j \in [m]$) can apply $y^{(j)} = eval(k^{(j)}, x)$. These outputs $y^{(j)}$ are still indistinguishable from numbers drawn from some random distribution, but have the property that $f(x) = \sum_{j=1}^{m} y^{(j)}$.

In our case, we exploit the class of comparison functions, i.e., the class of functions $\mathcal{F}_N^{<} = \{\phi_{\geq \tau_i} \mid i \in [N]\}$ where $\phi_{\geq t}(x) = \mathbb{I}[\phi(x) \geq t]$. The work [Boyle et al., 2016] proposes functions *gen* and *eval* for this class. The *gen* function returns a pair of keys, but can be extended to returning a larger number $m > 2$ of keys. Every party P_i with private data x_i applies the *gen* function to $\phi_{\geq (x_i)}$, i.e., sets $(k_i^{(1)}, k_i^{(2)}) = gen(\phi_{\geq (x_i)})$. For $j \in [m]$, server j receives all j-th keys, i.e., the set $K^{(j)} = \{k_i^{(j)} \mid i \in [n]\}$. Whenever one wants to evaluate the ECDF at some point, all servers $j \in [m]$ compute $Y^{(j)} = \sum_{i=1}^{n} eval(k_i^{(j)})$, and then jointly sum $\sum_{j \in [m]} Y^{(j)}$ and add the appropriate DP noise. The length of the generated keys is $O(L(\lambda + \log(n)))$ where λ is a security parameter typically larger than $\log(n)$.

The cost of this algorithm can be divided in two phases. First, the preprocessing phase where keys are generated is dominated by the sending of a key from each data owner party to each server. Second, the evaluation phase is relatively cheap, for every evaluation the user sends to all servers the number τ_i on which to evaluate the ECDF, the servers communicate among themselves for the addition and then send the answer back to the user. The communication cost is hence linear in m. The computation cost involves all servers going over the key with the input τ_i, and hence the computation cost is linear in the key length for every server.

The size of the keys k_i is constant but considerable (typically a few kilobits) so this approach is not recommended for small n or N, but the communication cost is constant for the parties owning the data and only linearly in the number of evaluations $|B|$ for the m servers, giving a total communication cost of $O(n + m|B|)$ which is for a constant m asymptotically better than the generic approach with the additional advantage that after the initial distribution of the keys only the m servers need to stay online to answer queries during the computation.

6.4.3 Inverse ECDF evaluation

Next to evaluating an ECDF, one also often needs to evaluate the inverse ECDF ${F_\phi}^{-1}$, i.e., one would like to know what is the value corresponding to a particular quantile. A natural strategy is to apply binary search, as illustrated by Algorithm 9.

Algorithm 9

function INV-ECDF-DP(p)
  $a \leftarrow 0$; $b \leftarrow 1$
  while $b - a > \psi$ **do** ▷ ψ = precision
    $m \leftarrow (a + b)/2$ ▷ Consider middle
    if $\acute{F}_\phi(X, m) < p$ **then** ▷ Split interval
      $a \leftarrow m$
    else
      $b \leftarrow m$
  ▷ Until interval small enough
  return $(a + b)/2$

6.5 Applications

6.5.1 The ROC curve and the area under it

The ROC curve [Fawcett, 2006] is a popular way to visualize the characteristics of a classifier. It gives a more complete view than a single performance measure such as accuracy or the area under the ROC curve.

As before, let $\mathcal{X}$ be a space of instances. Let $c^* : \mathcal{X} \to \{0, 1\}$ be a function assigning to each instance x its true class label. Let $c_0 : \mathcal{X} \to \mathbb{R}$ be a function assigning to each instance an estimated score, where instances with a lower score are more likely to be positive (have class 1) and instance with a higher score are more likely to be negative (have class 0). Let $\hat{c}(t, x) = \mathbb{I}[c_0(x) \leq t]$.

Let $X \in \mathcal{X}^n$ be a dataset and let t be a threshold, the true positive rate is

$$TPR(X, t) = \frac{\{x \in X \mid c^*(x) = 1 \wedge \hat{c}(t, x) = 1\}}{\{x \in X \mid c^*(x) = 1\}}$$

and the false positive rate is

$$FPR(X, t) = \frac{\{x \in X \mid c^*(x) = 0 \wedge \hat{c}(t, x) = 1\}}{\{x \in X \mid c^*(x) = 0\}}$$

The ROC curve plots TPR against FPR, so $(r_F, r_T) \in ROC$ iff there is a threshold t such that $r_T = TPR(X, t)$ and $r_F = FPR(X, t)$. Let $c_{max} = \max_{x \in X} c_0(x)$. Let $\phi_{TP}(x) = \mathbb{I}[c^*(x) = 1 \wedge \hat{c}(t, x) = 1]$ and $\phi_{FP}(x) = \mathbb{I}[c^*(x) = 0 \wedge \hat{c}(t, x) = 1]$. Then, $TPR(X, t) = F_{\phi_{TP}}(X, t)/F_{\phi_{TP}}(X, c_{max})$ and $FPR(X, t) = F_{\phi_{FP}}(X, t)/F_{\phi_{FP}}(X, c_{max})$. So publishing $\hat{F}_{\phi_{TP}}$ and $\hat{F}_{\phi_{FP}}$ is sufficient to transmit an approximate ROC curve. Moreover, if we add sufficient noise to make both functions $\epsilon/2$-differentially private, their combined disclosure is 2ϵ-differentially private.

Figure 6.5.1 illustrates this process on a relatively small dataset (see Section 6.6.2) where the effect of DP noise is clearly visible. As both coordinates of a point in the ROC curve are ECDFs to which noise is added, one can see that $\hat{F}_\phi$ has both horizontal and vertical deviations from the true ROC curve. In this figure, one can also see to some extent the organization of the noise as binary tree: the first half of the DP curve seems to be above the true ROC curve, while the latter half is lower, suggesting the noise variable that was added to the first half got a clearly higher value. The smoothed curves resolve this problem and stay in this case much closer to the true ROC curve.

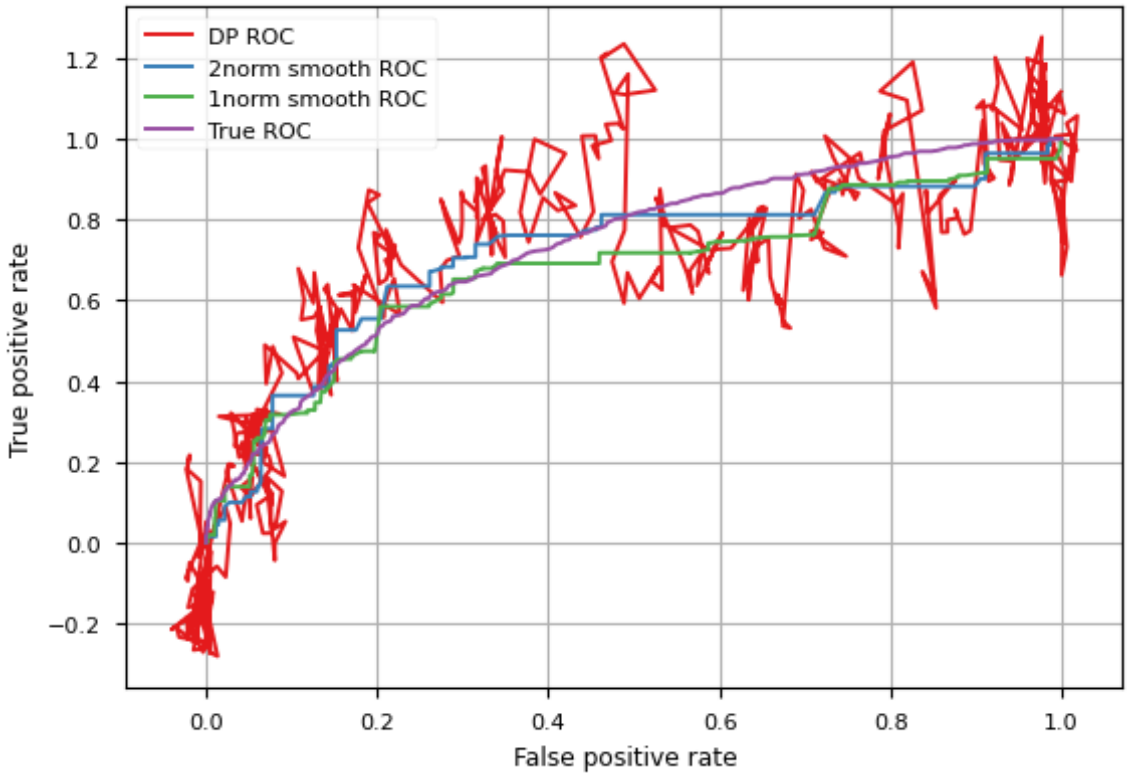


Figure 6.5.1: ROC curve for logistic regression on the Heart disease dataset, and ϵ-DP curves with $\epsilon = 0.5$.

As the size of the dataset increases, the impact of the noise decreases and even for smaller values of ϵ the differentially private curve gives a good picture of the true one, e.g., see Figures 6.5.2 and 6.5.3.

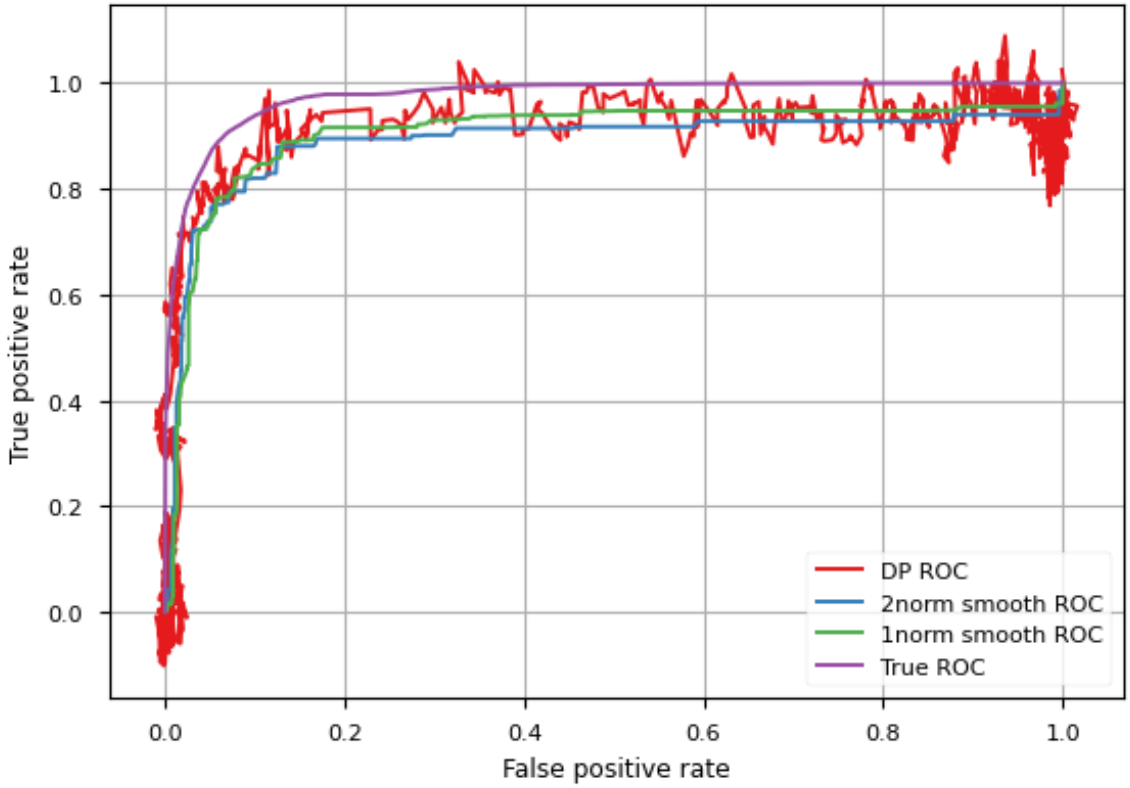


Figure 6.5.2: ROC curve for logistic regression on the Bank dataset, and ϵ-DP curves with $\epsilon = 0.2$.

6.5.2 Calibration and the Hosmer-Lemeshow statistic

The Hosmer-Lemeshow test is a statistical test which is popular in health science [Hosmer et al., 2013] and other domains. While it has some limitations, there is no universal agreement on what is the best alternative, and the Hosmer-Lemeshow statistic is still quite commonly used. It is often used as calibration test for logistic regression, but may also be applied to other machine learning models [Meyfroidt et al., 2011].

While applying a non-decreasing function to the output of a classifier will not change its ROC curve, it will impact its callibration. For models outputting a probability that an instance is positive, it is desirable that the estimated probability of being positive is close to the true probability. The Hosmer-Lemeshow statistic can evaluate such goodness-of-fit. To compute it, a first step is to rank all instances in increasing order of predicted probability of being positive. Then, this ranked list is partitioned into Q equally sized quantile groups, where typically $Q = 10$, so group 1 contains the

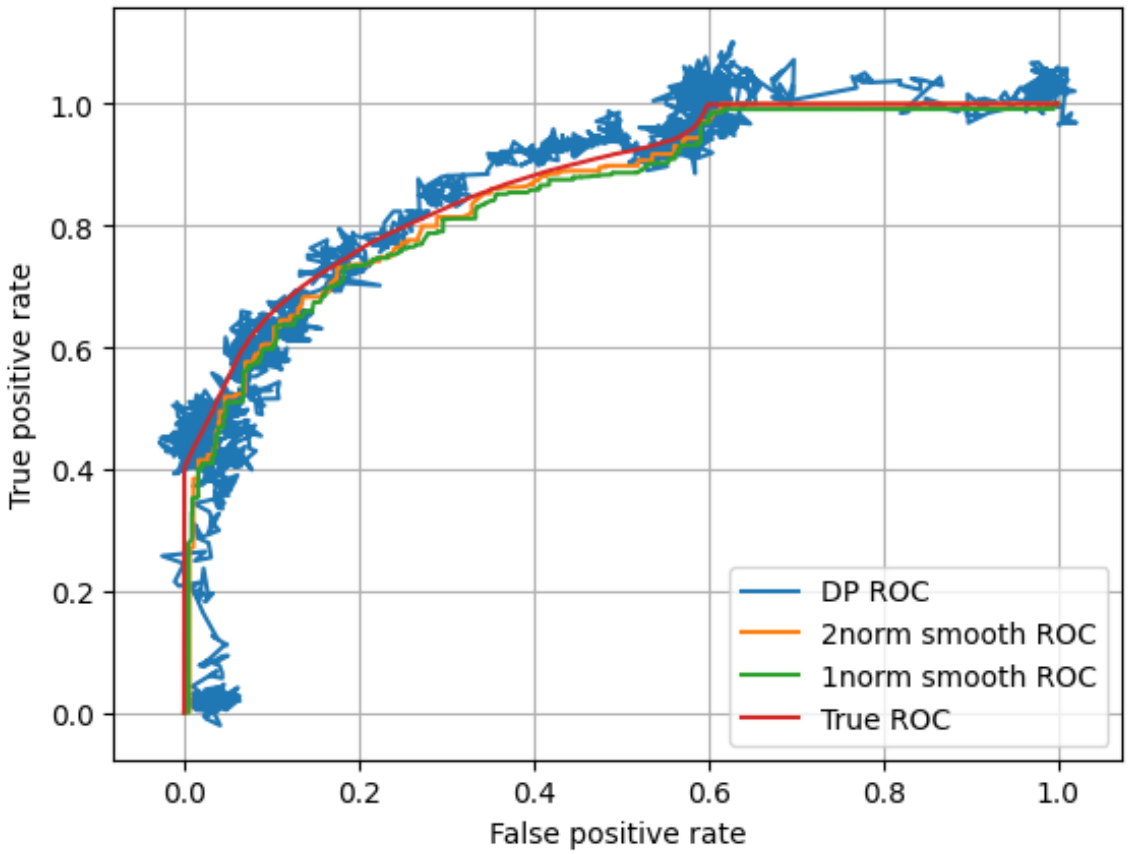


Figure 6.5.3: ROC curve for logistic regression on the Diabetic dataset, and ϵ-DP curves with $\epsilon = 0.05$.

instances which are predicted to be most likely negative and group Q contains the instances which are predicted to be most likely positive. Then, the Hosmer-Lemeshow statistic is defined by

$$H = \sum_{i=1}^{Q} \left(\frac{(O_{1i} - E_{1i})^2}{E_{1i}} + \frac{(O_{0i} - E_{0i})^2}{E_{0i}} \right) \tag{6.5.1}$$

Where O_{1i} is the observed number of positive instances, E_{1i} is the predicted number of positive instances, O_{0i} is the observed number of negative instances and E_{0i} is the predicted number of negative instances in group i. So a group is a collection of instances where the expected probability lies in a specific interval, and the test checks how far the observed class distribution deviates from this. Once the Hosmer-Lemeshow statistic is computed, one can compare it to a chi-squared distribution

with $Q-2$ degrees of freedom to test the hypothesis that the observed classes in each group are distributed according to the predictions.

Assume we have a model $\mathcal{M} : X \to [0,1]$ mapping each instance on the predicted probability it is positive and a function $\mathcal{Y} : X \to \{0,1\}$ mapping every instance on its true class label, either 0 (negative) or 1 (positive). Let $\mathcal{M}[l,u](x) = \mathcal{M}(x).\mathbb{I}\,[l \leq \mathcal{M}(x) \leq u]$ and $\mathcal{Y}[l,u](x) = \mathcal{Y}(x).\mathbb{I}\,[l \leq \mathcal{M}(x) \leq u]$. Then, H can be computed using only U-statistics following Algorithm 10.

Algorithm 10

1: **function** HL-STAT-DP(X : dataset, Q : number of groups, ϵ : privacy level; L : precision parameter)
2: $t_0 \leftarrow 0$; $t_Q \leftarrow 1$; $n \leftarrow |X|$
3: $\epsilon' \leftarrow \epsilon/(L+9)$
4: **for all** $q \in [Q-1]$: **do**
5: $t_q \leftarrow \hat{F}^{-1}_{\mathcal{M}}(q/Q)$ ▷ $(L+1)\epsilon'$-DP
6: **for all** $q \in [Q]$, $s \in \{0,1\}$ **do**
7: $E_{s,i} \leftarrow n.\hat{U}_{\mathcal{M}[t_{q-1},t_q]}(X, \{(-q,s)\})$ ▷ ϵ'-DP
8: $O_{s,i} \leftarrow n.\hat{U}_{\mathcal{Y}[t_{q-1},t_q]}(X, \{(-q,s)\})$ ▷ ϵ'-DP
9: **return** $\sum_{i=1}^{Q} \left(\frac{(O_{1i}-E_{1i})^2}{E_{1i}} + \frac{(O_{0i}-E_{0i})^2}{E_{0i}} \right)$

Theorem 6.5.1. *Running Algorithm 10 and disclosing any results of U-statistics it invokes is ϵ-DP.*

Proof. The algorithm queries data at lines 5 and at lines 7–8.

First, from Theorem 6.3.1 we know that by using $Lap(1/\epsilon')$ noise variables for evaluating $\hat{F}^{-1}_{\mathcal{M}}$, the resulting $\hat{F}_{\mathcal{M}}(\cdot)$ is $(L+1)\epsilon'$-DP independently of the number of needed evaluations of it during calls to Algorithm 9.

Next, for the evaluation of the statistics in lines 7–8 independent Laplacian random variables are used. However, when we compare two adjacent datasets where only one instance differs, only 2 of the Q groups and the corresponding 8 statistics are affected. Hence, if all $4Q$ statistics in lines 7–8 are ϵ'-DP, together they are $8\epsilon'$-DP.

In summary, applying the classic composition rule for differential privacy we get that the algorithm is $(L+1)\epsilon' + 8\epsilon' = \epsilon$ -differentially private. □

In particular, Algorithm 10 may publish both the ECDF of predicted probabilities $\hat{F}_{\mathcal{M}}$ and the statistics $O_{s,i}$ and $E_{s,i}$, we only assume the aggregation primitives used for computing the U-statistics are secure.

6.6 Experiments

6.6.1 Setup

Our experiments aim at providing illustrations to our approach and at providing more insight in the practical behavior of our proposal. Unless stated otherwise, our experiments average over 100 runs. Experiments were performed on Intel Core i7-4600U CPUs at 2.10GHz with 16Gb of RAM. Code (using Python 3.9) to reproduce all experiments will be downloadable from the website of the authors. For experiments involving ROC curves and the Hosmer-Lemeshow statistic, we first trained a logistic regression model using the Scikit-Learn package. Our goal was not to obtain the most performant classifier, but to illustrate how our methods can assess properties of an arbitrary classifier.

6.6.2 Datasets

6.6.2 (a) Synthetic data

We will use a dataset $X_{pois(\lambda)}$ constructed as follows. Let $N = 2^{15}$. For all values $i \in [N]$, the number of instances in $X_{pois(\lambda)}$ such that $\phi(x_j) = i$ follows a Poisson distribution $Pois(\lambda)$. At a high level this dataset has a steadily increasing ECDF, however locally there is quite some variance in how quickly it increases.

6.6.2 (b) Real-world data

We use the following datasets:

- Heart disease prediction : This data set aims to pinpoint the most relevant/risk factors of heart disease as well as predict the 10-year risk of coronary heart disease using information about the individuals (e.g. age, gender, smoking habits and blood pressure). (https://www.kaggle.com/dileep070/heart-disease-prediction-usin
- Bank full : This data set is related to a marketing campaign of a Portuguese banking institution. It intends to know if the client would subscribe or not to the product (bank term deposit), using features about the clients (e.g. age, job, education level, marital status, owning a house, having loans). (https://www.kaggle.com/krantiswalke/bankfullcsv)
- Diabetes data set: It studies the effects of diabetes to the readmission of patients to the hospital (https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008#)

Table 6.1 summarizes some relevant characteristics.

Table 6.1: Dataset summary giving the number of instances, the fraction of positive instances and the number of features.

Data set	#inst	pos.frac	#feature
Heart disease	4328	0.152	15
Bank-full	45211	0.117	16
Diabetes	101766	0.460	49

6.6.3 Smoothing

To investigate the effect of smoothing on the error induced by the DP noise, we start from the $X_{Pois(\lambda)}$ dataset. Figure 6.6.1 shows for $p = 1, 2$ curves plotting as a function of ϵ the effect of smoothing $\hat{F}_\phi$ on the error made by the DP noise as

$$\frac{\left\| F_\phi(X_{pois(\lambda)}, \tau) - \acute{F}_\phi(X_{pois(\lambda)}, \tau) \right\|_2^2}{\left\| F_\phi(X_{pois(\lambda)}, \tau) - \hat{F}_\phi(X_{pois(\lambda)}, \tau) \right\|_2^2}$$

One can see that, as expected for a 2-norm evaluation, the 2-norm smoothing outperforms the 1-norm smoothing, especially for small ϵ values. For roughly $\epsilon \geq 0.2$ its curve is below 1, meaning 2-norm smoothing in this range reduces the 2-norm error induced by the DP noise. The higher ϵ becomes, the less likely it is the small amount of DP noise will make the ECDF non-increasing, hence the effect of smoothing on the DP error goes to zero. Figure 6.6.2 shows the same information for fixed ϵ and varying λ. For large λ, the ECDF is strongly increasing and the relevance of smoothing is limited as adding DP noise rarely makes it non-decreasing. For moderate values of λ, smoothing improves the DP noise induced error.

6.6.4 Evaluating an ECDF or its inverse

Starting from the $X_{Pois(\lambda)}$ dataset, Figure 6.6.3 plots as a function of ϵ the mean square errors $(F_\phi(x) - \hat{F}_\phi(x))^2$, ${F_\phi}^{-1}(x) - \hat{F}_\phi^{-1}(x)$ and ${F_\phi}^{-1}(x) - \acute{F}_\phi^{-1}(x)$ when evaluating on points x uniformly distributed over the relevant domains. The inverse ECDF evaluations are performed using Algorithm 9.

The fact that $\hat{F}.$ is not guaranteed to be non-decreasing and this could confuse the binary search algorithm does not seem to have a strong impact on the accuracy of the evaluation. In fact, both the mean squared errors of $\hat{F}_\phi$ and the inverses of its smoothed and non-smoothed versions are of the same order of magnitude.

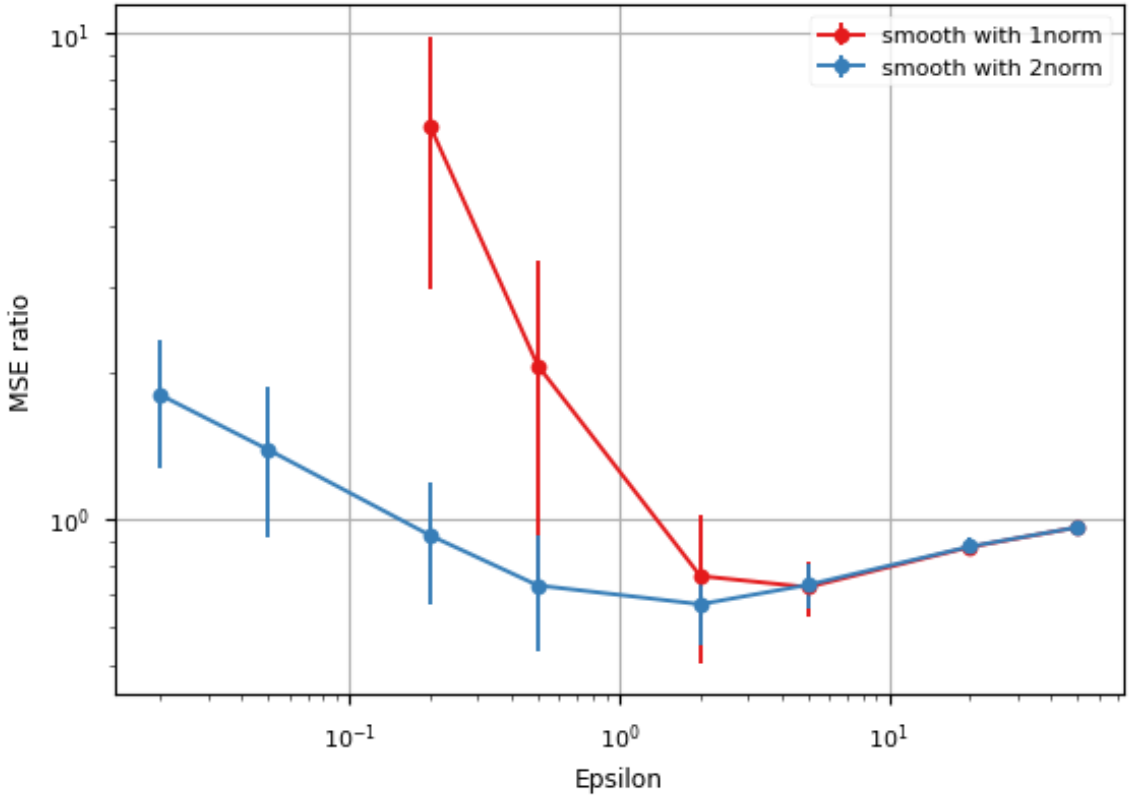


Figure 6.6.1: Effect of smoothing on DP error - fixed $\lambda = 3$

6.6.5 ROC curve estimation

Figure 6.6.4 plots, as a function of ϵ, the 1-norm difference between the true ROC curve and the smoothed differentially private ROC curve for the Bank dataset. This corresponds to the area of the symmetric difference of the area under the true ROC curve and the area under the differentially private ROC curve. Even when the area under both curves would be the same, this value can be non-zero as the curves themselves differ. As expected, error decreases with increasing ϵ. We see that the 1-norm and 2-norm smoothing perform about equally well. As the unsmoothed differentially private ROC curves cross themselves, it is hard to compare their area with the area of the true ROC curve.

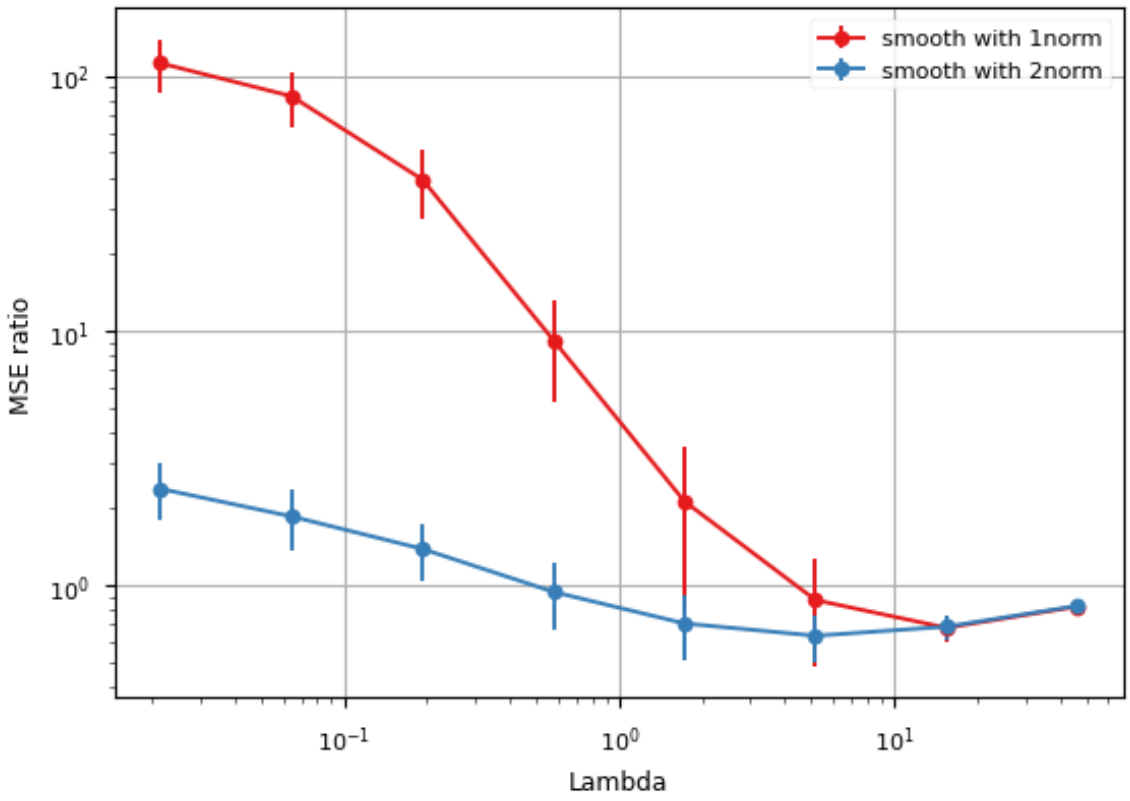


Figure 6.6.2: Effect of smoothing on DP error - fixed $\epsilon = 1$

6.6.6 Hosmer-Lemeshow

As for ROC curves, to understand calibration one can both look at the complete picture of the predicted and observed counts of positive / negative instances in each of the Q groups, and one can look at the HL-statistic which summarizes it as a χ^2 statistic. Here, we show results for the latter approach.

Figure 6.6.5 shows for a logistic regression model on the Bank dataset the mean square error of the Hosmer-Lemeshow test when computed privately for different values of ϵ. Especially for the Bank dataset we observe a quite large variance over the several runs. This can be explained by the fact that the Bank dataset has less balanced classes (see Table 6.1). This causes both predicted and observed counts of positive examples in especially the lowest of the $Q = 10$ groups to be rather small. Even a small error in the small number $E_{1,1}$ (the predicted number of positives in the first group) appearing in the denominator of a term in the H statistic (see Eq 6.5.1) may cause a large error in the final statistic. We can conclude that especially if ϵ is

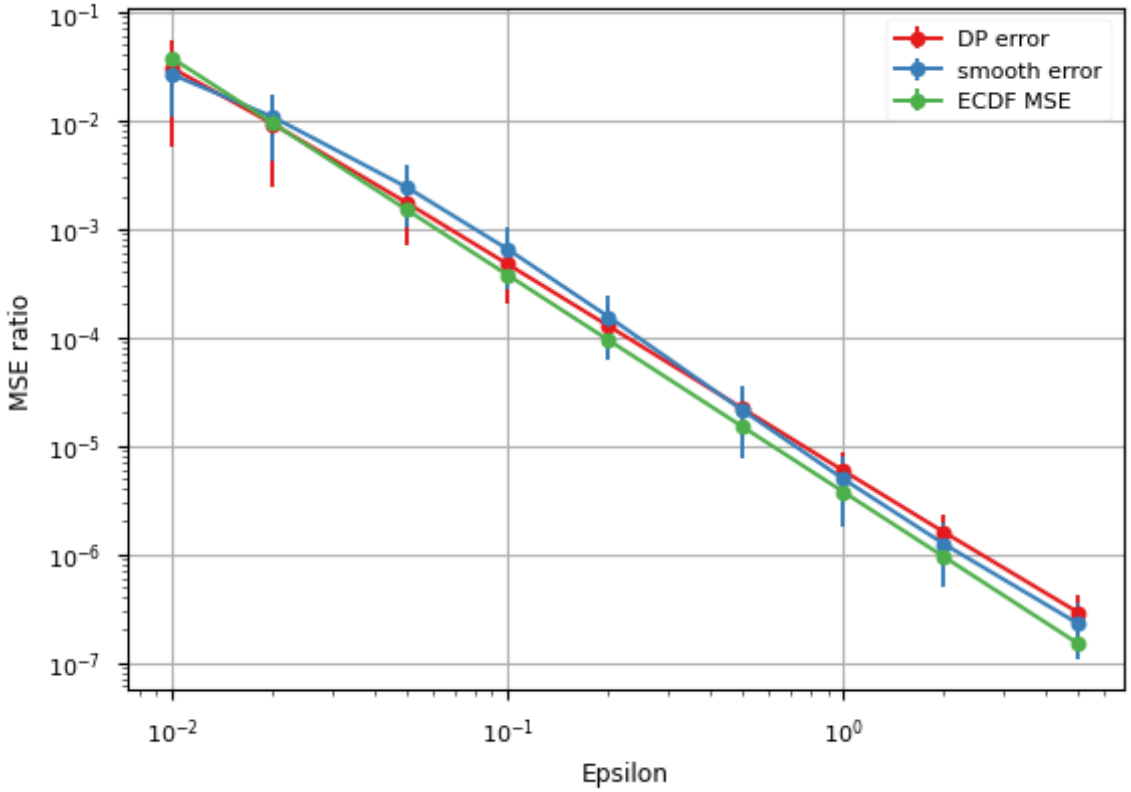


Figure 6.6.3: Inverse ECDF

small, if there is no need to know the individual statistics $E_{\cdot,\cdot}$ and $O_{\cdot,\cdot}$ and multi-party computation is available for other operations than U-statistics, a direct approach may be preferable where one first computes securely the correct statistic and then adds noise at the end proportional to the sensitivity of (only) the HL statistic.

6.6.7 Communication Cost

The cost of a distributed algorithm is often dominated by its communication cost. We compare two DP-ECDF implementations: one using Function Secret Sharing (FSS) and the other based on Secure Aggregation. We compute the total communication cost for each method on computing DP-ECDF applied to display complete ROC curves for the Bank dataset, varying the number of data owners n, the number of servers m, and the number of evaluated thresholds $|B|$. By total communication cost, we mean the total number of bits sent by all parties during the protocol execution which includes the initial key distribution (operation *gen*) and the evaluation phase

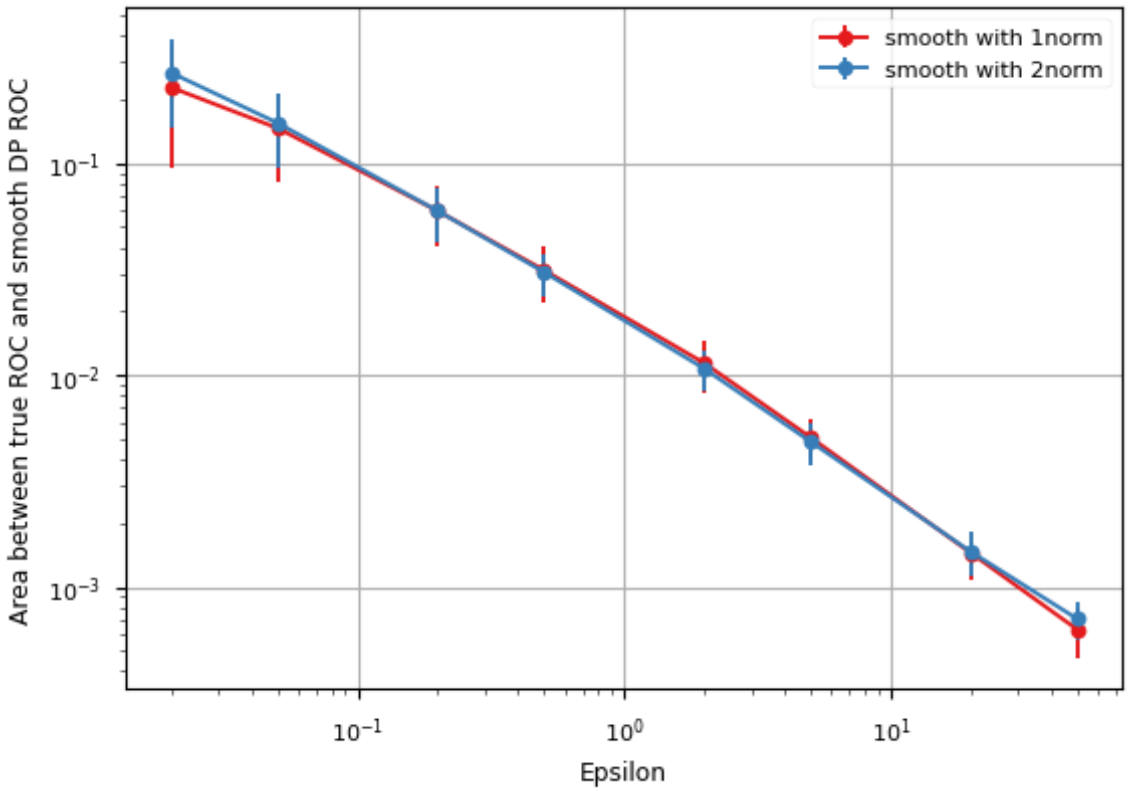


Figure 6.6.4: ROC curve estimation error

(operation *eval*), see Section 6.4.2 for more details.

Based on the analysis detailed in Section 6.4, the theoretical communication complexities of the two protocols are FSS: $O(n + m|B|)$ while secure aggregation is $O(n|B|)$. This indicates FSS is preferable when $m|B| < n|B|$, i.e., for smaller m or larger $|B|$.

Our experimental evaluation confirms the theoretical expectations and reveals trade-offs across different parameter regimes. We show the results in Figures 6.6.6(a) and 6.6.6(b). The x-axis represents the number of evaluated thresholds $|B|$ or the number of servers m, while the y-axis shows the total communication cost in bits. The curves represent the total communication cost for each method, with the red line indicating FSS and the blue line indicating secure aggregation.

These results highlight that FSS offers lower communication costs in scenarios with fewer servers and/or higher evaluation resolution. Conversely, secure aggregation becomes more communication-efficient as the number of servers grows or the number of evaluation points shrinks. Given that modern federated systems may operate

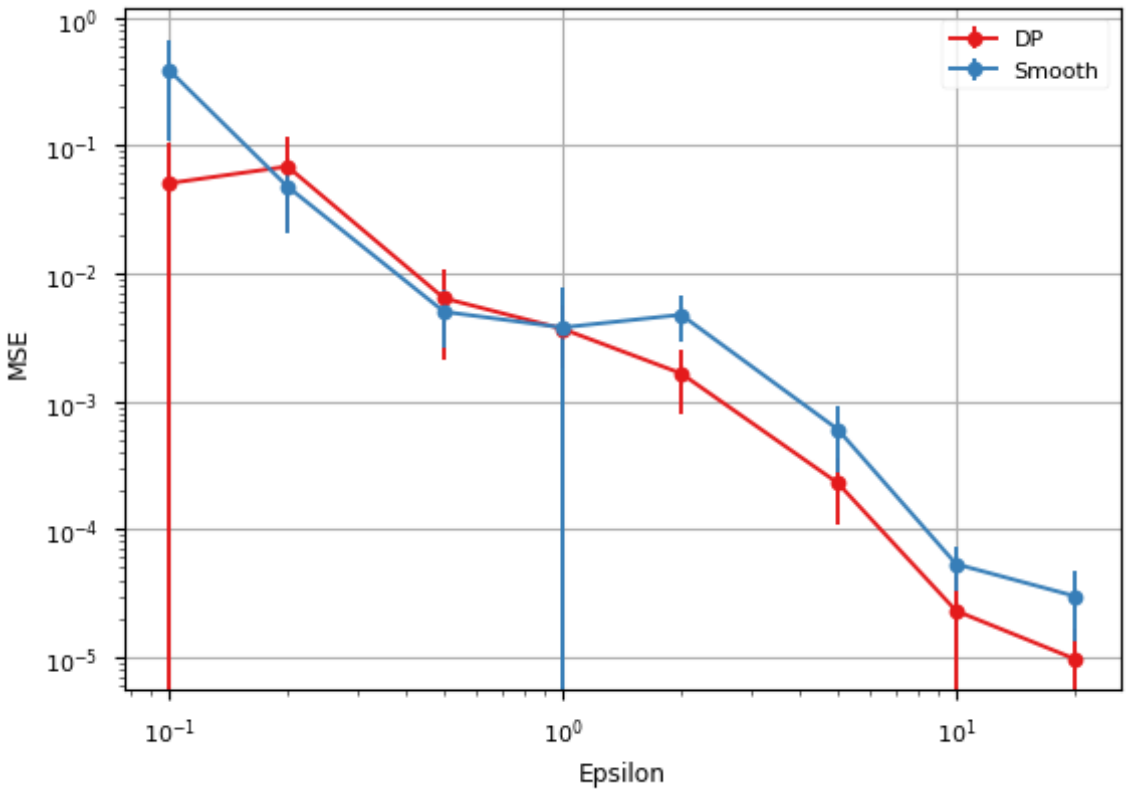
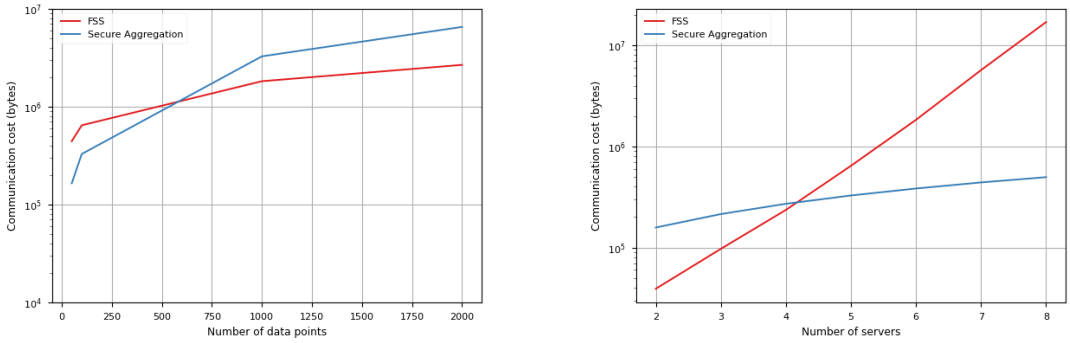


Figure 6.6.5: Hosmer-Lemeshow statistic relative MSE for a logistic regression model on the Bank dataset

under tight bandwidth constraints or in large-scale deployments, understanding these trade-offs is essential.

6.6.8 Runtime

Here, we complement the analysis on communication cost with an experiment on the most expensive local computation: the smoothing of the private ECDF. For solving the optimization problem in Eq 6.3.6, we use the CVXOPT package, in particular a linear program solver for the 1-norm smoothing and a quadratic program solver for the 2-norm smoothing. Figure 6.6.7 shows the runtime as a function of N, which is a good problem size parameter as the number of variables in the optimization problem grows as $O(N \log(N))$. One can observe that the quadratic program is solved more quickly.



(a) Number of thresholds (b) Number of servers

Figure 6.6.6: Empirical comparison of FSS and Secure Aggregation for DP-ECDF communication cost. 6.6.6(a) Varying number of thresholds $|B|$; 6.6.6(b) Varying number of servers m.

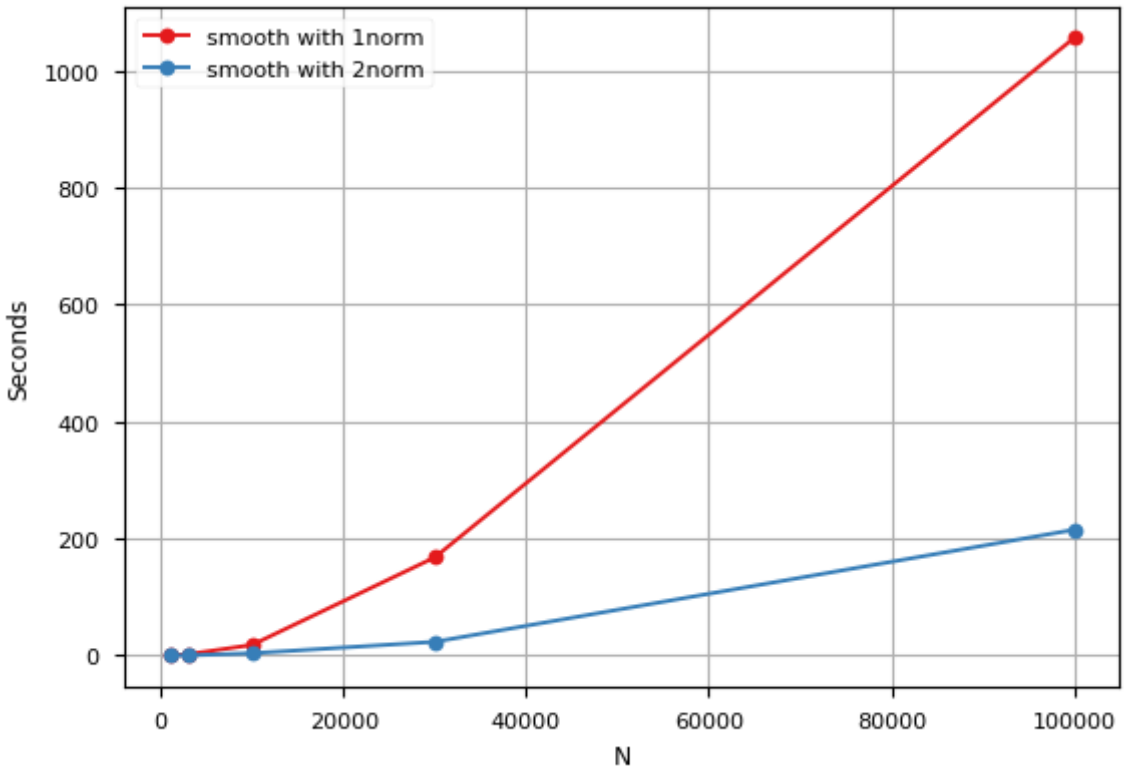


Figure 6.6.7: Runtime of solving Eq 6.3.6

6.7 Discussion

In this chapter we studied differentially private empirical cumulative distribution functions. We proved privacy guarantees and proposed algorithms to securely compute such private ECDFs. We elaborated in more depth two applications of ECDFs: ROC curves and the Hosmer-Lemeshow statistic. Our experimental results suggest the approach can convey the full information of an ECDF at a reasonable precision and privacy level.

Cumulative distribution functions are important in various other areas of machine learning and statistics. One application we didn't elaborate in-depth concerns histograms of (discretized) continuous variables, e.g., a histogram of the yearly income of a set of persons grouped in bins of \$5000. A common strategy [Cardoso and Rogers, 2022] is to add DP noise to the count in each bin independently. An alternative strategy would be to consider the cumulative distribution, which can be made private by noise only logarithmic in the number of bins. The interpretation then is that noise can not only consist of a change in the count in a bin, but also in a shift from one bin to an adjacent one, as depicted in [Alabi et al., 2023].

There are several potentially interesting lines of future work. Among others it would be interesting to elaborate more applications of ECDF, to develop more efficient algorithms to securely compute private ECDF, get a better understanding of the various statistical processes affecting the error DP noise induces and to leverage [Chaudhuri et al., 2024] to broaden the algorithm to all classes of U-statistics.

Chapter 7

Conclusion and Perspectives

7.1 Conclusion

This thesis has addressed critical challenges in privacy-preserving machine learning, with a specific focus on optimizing and analyzing machine learning models in scenarios that involve repeated querying. Repeated queries, such as iterative gradient computations in optimization or multiple evaluations in empirical cumulative distribution function (ECDF) estimation, give significant challenges to maintaining strong privacy guarantees while preserving utility. By designing specific privacy-preserving strategies to the structural properties of these applications, this thesis demonstrates that it is possible to achieve better privacy-utility trade-offs than with generic methods.

In the context of optimization (Chapter 5), we demonstrated that constraining models to a subclass of Lipschitz constrained functions allows for improved privacy-utility trade-offs. By leveraging gradient sensitivity properties, we showed that the bias introduced by selecting models from this subclass is less harmful to utility compared to the bias induced by gradient clipping in popular differentially private optimization methods. This insight led to the development of Lip-DP-SGD, a novel optimization algorithm that reduces the privacy budget and improves both accuracy and runtime performance compared to state-of-the-art, as demonstrated experimentally on a diverse range of datasets.

For ECDF computation (Chapter 6), we explored its unique structural properties to design a novel algorithm that provides strong privacy guarantees while preserving utility. Unlike traditional methods, our approach is compatible with a wide range of security protocols, including function secret sharing, enabling its use in more complex scenarios such as multi-party computation. This contribution highlights how structural insights into ECDFs can be leveraged to achieve enhanced privacy and enable practical deployment in collaborative environments.

Through these two contributions, this thesis has shown that tailoring privacy-preserving techniques to specific applications can uncover new opportunities for improving privacy-utility trade-offs while enabling broader practical setups. By addressing optimization and ECDF estimation in particular, this work lays a foundation for the development of more efficient, accurate, and scalable privacy-preserving techniques in machine learning and data analysis.

7.2 Perspectives

Looking forward, the findings and methods presented in this thesis suggest promising directions for future research.

Extending these insights to other applications, exploring tighter privacy bounds for specific tasks, and enhancing the scalability of privacy-preserving protocols in distributed and collaborative settings are just a few avenues for continued exploration. Ultimately, this work contributes to the broader effort to balance the competing demands of privacy and utility in modern machine learning systems, paving the way for more secure and practical data-driven solutions.

We list bellow a few of these future research topics.

Theoretical analysis of Lip-DP-SGD. In Chapter 5 we prove empirically the the benefit one can take from Lip-DP-SGD. However, further theoretical analysis of Lip-DP-SGD, especially the interaction between sensitivity, learning rate and number of iterations, remains an interesting area of research, similar to the work of [Song et al., 2020] on DP-SGD. An analysis on the interactions between hyperparameters would provide valuable insights into the optimal use of our method and its potential combination with other regularization techniques.

Deep Learning architectures. In Chapter 5, we demonstrated that Lip-DP-SGD is applicable to various feed-forward neural networks. Specifically, we achieved state-of-the-art results with MLPs, CNNs, and ResNets. While this work primarily focused on these architectures, there is significant potential to extend our approach to other classes of models, as well as to additional instances of feed-forward neural networks, such as transformers. Recent studies, such as [Qi et al., 2023], have proposed methods to efficiently enforce Lipschitz constraints on transformers. However, we have not yet explored whether Lip-DP-SGD can be directly applied to these models without encountering issues like DP noise amplification due to the large number of parameters inherent to transformers. This raises the possibility that specific refinements may be required to adapt our method effectively in such cases.

Exponential Mechanism. The exponential mechanism is particularly well-suited for optimization scenarios; however, there is limited literature on leveraging this mechanism for differentially private model optimization. While the Gaussian mechanism has become a de facto choice due to its favorable composition properties, it also presents significant drawbacks in our context, such as the reliance on the ℓ_2 norm for estimating the Lipschitz value. Incorporating the exponential mechanism into Lip-DP-SGD could overcome these limitations, enabling the use of ℓ_1 or ℓ_∞ norms to provide faster and more reliable estimates of Lipschitz values.

Display continuous metrics with DP. The DP ECDF allows for the publication of the complete ROC curve while ensuring privacy guarantees. It is based on a discrete count function, which we leveraged to design a binary mechanism. However, sharing continuous metrics—such as those in healthcare, finance (e.g., stock prices), or location tracking—introduces distinct challenges and promising research directions. One key difficulty stems from temporal correlations in continuous data. When data points are interdependent over time, a single modification can influence multiple points, potentially heightening the risk of privacy breaches. This phenomenon, known as "temporal privacy leakage," can weaken the expected privacy protection of DP mechanisms [Cao et al., 2017]. Some approaches attempt to exploit the autocorrelation inherent in certain metrics [Katsomallos et al., 2019], but they struggle to achieve a satisfactory balance between privacy and utility.

Bibliography

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Abadi et al., 2016] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep Learning with Differential Privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. arXiv: 1607.00133.

[Alabi et al., 2023] Alabi, D., Ben-Eliezer, O., and Chaturvedi, A. (2023). Bounded space differentially private quantiles. *Transactions on Machine Learning Research*.

[Andrew et al., 2022] Andrew, G., Thakkar, O., McMahan, H. B., and Ramaswamy, S. (2022). Differentially Private Learning with Adaptive Clipping. In *Advances in Neural Information Processing Systems*. arXiv. arXiv:1905.03871 [cs, stat].

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 214–223. JMLR.org.

[Balandat et al., 2020] Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. (2020). BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. arXiv:1910.06403 [cs, math, stat].

[Balle et al., 2018] Balle, B., Barthe, G., and Gaboardi, M. (2018). Privacy amplification by subsampling: Tight analyses via couplings and divergences. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Balle et al., 2020] Balle, B., Bell, J., Gascón, A., and Nissim, K. (2020). Private Summation in the Multi-Message Shuffle Model. In *CCS*.

[Bassily et al., 2019] Bassily, R., Feldman, V., Talwar, K., and Guha Thakurta, A. (2019). Private Stochastic Convex Optimization with Optimal Rates. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[Bassily et al., 2014] Bassily, R., Smith, A., and Thakurta, A. (2014). Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473.

[Becker et al., 1996] Becker, Barry, Kohavi, and Ronny (1996). Adult. UCI Machine Learning Repository. DOI: 10.24432/C5XW20.

[Bengio et al., 2012] Bengio, Y. et al. (2012). Practical recommendations for gradient-based training of deep architectures. *Neural networks: Tricks of the trade*, pages 437–478.

[Bethune et al., 2023] Bethune, L., Massena, T., Boissin, T., Prudent, Y., Friedrich, C., Mamalet, F., Bellet, A., Serrurier, M., and Vigouroux, D. (2023). Dp-sgd without clipping: The lipschitz neural network way.

[Béthune et al., 2023] Béthune, L., Novello, P., Coiffier, G., Boissin, T., Serrurier, M., Vincenot, Q., and Troya-Galvis, A. (2023). Robust one-class classification with signed distance function using 1-lipschitz neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

[Bonawitz et al., 2017] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017). Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA. Association for Computing Machinery.

[Boyle et al., 2015] Boyle, E., Gilboa, N., and Ishai, Y. (2015). Function secret sharing. In Oswald, E. and Fischlin, M., editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 337–367, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Boyle et al., 2016] Boyle, E., Gilboa, N., and Ishai, Y. (2016). Function Secret Sharing: Improvements and Extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1292–1303, Vienna Austria. ACM.

[Cao et al., 2017] Cao, Y., Yoshikawa, M., Xiao, Y., and Xiong, L. (2017). Quantifying differential privacy under temporal correlations. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 821–832.

[Cape et al., 2017] Cape, J., Tang, M., and Priebe, C. (2017). The two-to-infinity norm and singular subspace geometry with applications to high-dimensional statistics. *The Annals of Statistics*, 47.

[Cardoso and Rogers, 2022] Cardoso, A. R. and Rogers, R. (2022). Differentially Private Histograms under Continual Observation: Streaming Selection into the Unknown. *arXiv:2103.16787 [cs]*. arXiv: 2103.16787.

[Carlini et al., 2021] Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. (2021). Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association.

[Chan et al., 2012] Chan, T.-H. H., Shi, E., and Song, D. (2012). Privacy-preserving stream aggregation with fault tolerance. In *Financial Cryptography*.

[Chaudhuri et al., 2024] Chaudhuri, K., Loh, P.-L., Pandey, S., and Sarkar, P. (2024). On differentially private u statistics. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Information Processing Systems*, volume 37, pages 23078–23122. Curran Associates, Inc.

[Chaudhuri and Monteleoni,] Chaudhuri, K. and Monteleoni, C. Privacy-preserving logistic regression.

[Chaudhuri et al., 2011] Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. (2011). Differentially Private Empirical Risk Minimization. *Journal of Machine Learning Research*, page 41.

[Chen and Lee, 2020] Chen, C. and Lee, J. (2020). Stochastic adaptive line search for differentially private optimization. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1011–1020.

[Chen et al., 2020] Chen, X., Wu, S. Z., and Hong, M. (2020). Understanding gradient clipping in private sgd: A geometric perspective. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13773–13782. Curran Associates, Inc.

[Cheu et al., 2019] Cheu, A., Smith, A. D., Ullman, J., Zeber, D., and Zhilyaev, M. (2019). Distributed Differential Privacy via Shuffling. In *EUROCRYPT*.

[De et al., 2022] De, S., Berrada, L., Hayes, J., Smith, S. L., and Balle, B. (2022). Unlocking High-Accuracy Differentially Private Image Classification through Scale. *arXiv preprint arXiv:2204.13650*.

[Deng, 2012] Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

[Dinur and Nissim, 2003] Dinur, I. and Nissim, K. (2003). Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, page 202–210, New York, NY, USA. Association for Computing Machinery.

[Drechsler et al., 2022] Drechsler, J., Globus-Harris, I., Mcmillan, A., Sarathy, J., and Smith, A. (2022). Nonparametric differentially private confidence intervals for the median. *Journal of Survey Statistics and Methodology*, 10(3):804–829.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of Machine Learning Research*, volume 12, pages 2121–2159.

[Dwork et al., 2006a] Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our Data, Ourselves: Privacy Via Distributed Noise Generation. In Vaudenay, S., editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 486–503, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Dwork et al., 2006b] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating Noise to Sensitivity in Private Data Analysis. In Halevi, S. and Rabin, T., editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Dwork et al., 2010a] Dwork, C., Naor, M., Pitassi, T., and Rothblum, G. N. (2010a). Differential privacy under continual observation. In *Proceedings of the 42nd ACM symposium on Theory of computing - STOC '10*, page 715, Cambridge, Massachusetts, USA. ACM Press.

[Dwork and Roth, 2013] Dwork, C. and Roth, A. (2013). The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407.

[Dwork et al., 2010b] Dwork, C., Rothblum, G. N., and Vadhan, S. (2010b). Boosting and Differential Privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60, Las Vegas, NV, USA. IEEE.

[Erlingsson et al., 2019] Erlingsson, U., Feldman, V., Mironov, I., Raghunathan, A., and Talwar, K. (2019). Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity. In *SODA*.

[Fawcett, 2006] Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874.

[Ghazi et al., 2020] Ghazi, B., Kumar, R., Manurangsi, P., and Pagh, R. (2020). Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead. In *ICML*.

[Goldreich, 2006] Goldreich, O. (2006). *Foundations of Cryptography: Volume 1*. Cambridge University Press, USA.

[Gouk et al., 2020] Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. (2020). Regularisation of neural networks by enforcing lipschitz continuity.

[Hartmann and West, 2019] Hartmann, V. and West, R. (2019). Privacy-Preserving Distributed Learning with Secret Gradient Descent. Technical report, arXiv:1906.11993.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Hofmann, 1994] Hofmann, H. (1994). Statlog German Credit Data. UCI Machine Learning Repository. DOI: 10.24432/C5NC77.

[Hosmer et al., 2013] Hosmer, D. W., Lemeshow, S., and Sturdivant, R. X. (2013). *Applied Logistic Regression*. Wiley, New York.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML*.

[Jayaraman et al., 2018] Jayaraman, B., Wang, L., Evans, D., and Gu, Q. (2018). Distributed learning without distress: Privacy-preserving empirical risk minimization. In *NeurIPS*.

[Kasiviswanathan et al., 2011] Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011). What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826.

[Katsomallos et al., 2019] Katsomallos, M., Tzompanaki, K., and Kotzinos, D. (2019). Privacy, Space and Time: a Survey on Privacy-Preserving Continuous Data Publishing. *Journal of Spatial Information Science*, (19).

[Kifer et al., 2012] Kifer, D., Smith, A., and Thakurta, A. (2012). Private Convex Empirical Risk Minimization and High-dimensional Regression. In *Proceedings of the 25th Annual Conference on Learning Theory*, pages 25.1–25.40. JMLR Workshop and Conference Proceedings. ISSN: 1938-7228.

[Kingma and Ba, 2014] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

[Koskela and Honkela, 2020] Koskela, A. and Honkela, A. (2020). Learning Rate Adaptation for Differentially Private Learning. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 2465–2475. PMLR. ISSN: 2640-3498.

[Krizhevsky et al., 2009] Krizhevsky, A., Nair, V., and Hinton, G. (2009). Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*.

[Lacoste et al., 2019] Lacoste, A., Luccioni, A., Schmidt, V., and Dandres, T. (2019). Quantifying the carbon emissions of machine learning.

[Lagrange, 1788] Lagrange, J.-L. (1788). *Mécanique analytique*. Chez la Veuve Desaint, Paris, France.

[Lee and Kifer, 2018] Lee, J. and Kifer, D. (2018). Concentrated differentially private gradient descent with adaptive per-iteration privacy budget. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 1656–1665, New York, NY, USA. Association for Computing Machinery.

[Lei, 2011] Lei, J. (2011). Differentially private m-estimators. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

[Li et al., 2019] Li, S., Jia, K., Wen, Y., Liu, T., and Tao, D. (2019). Orthogonal deep neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(4):1352–1368.

[Mathur et al., 2022] Mathur, Akshay, Mathur, and Akshay (2022). NATICUSdroid (Android Permissions) Dataset. UCI Machine Learning Repository.

[McMahan et al., 2017] McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

[McSherry and Talwar, 2007] McSherry, F. and Talwar, K. (2007). Mechanism design via differential privacy. In *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE.

[Meyfroidt et al., 2011] Meyfroidt, G., Güiza, F., Cottem, D., De Becker, W., Van Loon, K., Aerts, J., Berckmans, D., Ramon, J., Bruynooghe, M., and Van den

Berghe, G. (2011). Computerized prediction of intensive care unit discharge after cardiac surgery: development and validation of a gaussian processes model. *BMC Medical Informatics and Decision Making*, 11(64).

[Mironov, 2017] Mironov, I. (2017). Renyi Differential Privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. arXiv:1702.07476 [cs].

[Mironov et al., 2019] Mironov, I., Talwar, K., and Zhang, L. (2019). Rényi Differential Privacy of the Sampled Gaussian Mechanism. *arXiv e-prints*, page arXiv:1908.10530.

[Miyato et al., 2018] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *Advances in Neural Information Processing Systems*.

[Narayanan and Shmatikov, 2007] Narayanan, A. and Shmatikov, V. (2007). How To Break Anonymity of the Netflix Prize Dataset.

[Papernot and Steinke, 2022] Papernot, N. and Steinke, T. (2022). Hyperparameter Tuning with Renyi Differential Privacy. In *International Conference on Learning Representations*. arXiv. arXiv:2110.03620 [cs].

[Papernot et al., 2021] Papernot, N., Thakurta, A., Song, S., Chien, S., and Erlingsson (2021). Tempered sigmoid activations for deep learning with differential privacy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9312–9321.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Polyak, 1964] Polyak, B. (1964). Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4:1–17.

[Polyak and Juditsky, 1992] Polyak, B. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *Siam Journal on Control and Optimization*, 30:838–855.

[Qi et al., 2023] Qi, X., Wang, J., Chen, Y., Shi, Y., and Zhang, L. (2023). Lipsformer: Introducing lipschitz continuity to vision transformers. In *The Eleventh International Conference on Learning Representations*.

[Qiao et al., 2020] Qiao, S., Wang, H., Liu, C., Shen, W., and Yuille, A. (2020). Micro-batch training with batch-channel normalization and weight standardization.

[Rademacher, 1919] Rademacher, H. (1919). Über partielle und totale differenzierbarkeit von funktionen mehrerer variabeln und über die transformation der doppelintegrale. *Mathematische Annalen*, 79:340–359.

[Rajkovic, 1997] Rajkovic, V. (1997). Nursery. UCI Machine Learning Repository. DOI: 10.24432/C5P88W.

[Realinho et al., 2021] Realinho, V., Vieira Martins, M., Machado, J., and Baptista, L. (2021). Predict students' dropout and academic success. UCI Machine Learning Repository. DOI: 1.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Sabater et al., 2022] Sabater, C., Bellet, A., and Ramon, J. (2022). An accurate, scalable and verifiable protocol for federated differentially private averaging.

[Sabater and Ramon, 2021] Sabater, C. and Ramon, J. (2021). Zero knowledge arguments for verifiable sampling. In *NeurIPS workshop on Privacy in Machine Learning*.

[Scaman and Virmaux, 2019] Scaman, K. and Virmaux, A. (2019). Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems*. arXiv. arXiv:1805.10965 [cs, stat].

[Shamir, 1979] Shamir, A. (1979). How to share a secret. *Commun. ACM*, 22(11):612–613.

[Shi et al., 2011] Shi, E., Chan, T.-H. H., Rieffel, E. G., Chow, R., and Song, D. (2011). Privacy-Preserving Aggregation of Time-Series Data. In *NDSS*.

[Shokri et al., 2016] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2016). Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18.

[Song et al., 2013] Song, S., Chaudhuri, K., and Sarwate, A. D. (2013). Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248, Austin, TX, USA. IEEE.

[Song et al., 2020] Song, S., Steinke, T., Thakkar, O., and Thakurta, A. G. (2020). Evading the curse of dimensionality in unconstrained private generalized linear problems. In *Proceedings of the 23th International Conference on Artificial Intelligence and Statistics*.

[Stewart, 2012] Stewart, J. (2012). *Calculus: Early Transcendentals*. Brooks/Cole Cengage Learning.

[Sweeney, 2000] Sweeney, L. (2000). Simple Demographics Often Identify People Uniquely. . *Pittsburgh*.

[Sweeney, 2002] Sweeney, L. (2002). K-anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

[Triastcyn and Faltings, 2019] Triastcyn, A. and Faltings, B. (2019). Bayesian Differential Privacy for Machine Learning. arXiv:1901.09697 [cs, stat] version: 1.

[UDHR, 1948] UDHR (1948). Universal declaration of human rights (1948). *Refugee Survey Quarterly*, 27(3):149–182.

[Wang et al., 2019] Wang, Y.-X., Balle, B., and Kasiviswanathan, S. P. (2019). Subsampled renyi differential privacy and analytical moments accountant. In Chaudhuri, K. and Sugiyama, M., editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1226–1235. PMLR.

[Wolberg et al., 1995] Wolberg, William, Mangasarian, Olvi, Street, Nick, and Street, W. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. DOI: 10.24432/C5DW2B.

[Wu and He, 2018] Wu, Y. and He, K. (2018). Group normalization. *International Journal of Computer Vision*, 128:742 – 755.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv:1708.07747 [cs, stat].

[Yeh, 2016] Yeh, I.-C. (2016). default of credit card clients. UCI Machine Learning Repository. DOI: 10.24432/C55S3H.

[Yoshida and Miyato, 2017] Yoshida, Y. and Miyato, T. (2017). Spectral Norm Regularization for Improving the Generalizability of Deep Learning. arXiv:1705.10941 [cs, stat].

[Yousefpour et al., 2021] Yousefpour, A., Shilov, I., Sablayrolles, A., Testuggine, D., Prasad, K., Malek, M., Nguyen, J., Ghosh, S., Bharadwaj, A., Zhao, J., Cormode, G., and Mironov, I. (2021). Opacus: User-friendly differential privacy library in PyTorch. *arXiv preprint arXiv:2109.12298*.

[Yu et al., 2019] Yu, L., Liu, L., Pu, C., Gursoy, M. E., and Truex, S. (2019). Differentially Private Model Publishing for Deep Learning. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 332–349. arXiv:1904.02200 [cs].

[Zagoruyko and Komodakis, 2016] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *BMVC*.