ABSTRACT

Title of Dissertation: STUDIES IN DIFFERENTIAL PRIVACY AND FEDERATED LEARNING

Christopher Cameron Zawacki
Doctor of Philosophy, 2024

Dissertation Directed by: Professor Eyad H. Abed
Department of Electrical and Computer Engineering

In the late 20^{th} century, Machine Learning underwent a paradigm shift from model-driven to data-driven design. Rather than field specific models, advances in sensors, data storage, and computing power enabled the collection of increasing amounts of data. The abundance of new data allowed researchers to fit flexible models directly to observed data. The influx of information made possible numerous advances, including the development of novel medicines, increases in efficiency of markets, and the proliferation of vast sensor networks.

However, not all data should be freely accessible. Sensitive medical records, personal finances, and private IDs are all currently stored on digital devices across the world with the expectation that they remain private. However, at the same time, such data is frequently instrumental in the development of predictive models. Since the beginning of the 21st century, researchers have recognized that traditional methods of anonymizing data are inadequate for protecting client identities. This dissertation's primary focus is the advancement of two fields of data privacy: Differential Privacy and Federated Learning.

Differential Privacy is one of the most successful modern privacy methods. By injecting carefully structured noise into a dataset, Differential Privacy obscures individual contributions while allowing researchers to extract meaningful information from the aggregate. Within this methodology, the Gaussian mechanism is one of the most common privacy mechanisms due to its favorable properties such as the ability of each client to apply noise locally before transmission to a server. However, the use of this mechanism yields only an *approximate* form of Differential Privacy. This dissertation introduces the first in-depth analysis of the Symmetric alpha-Stable (SaS) privacy mechanism, demonstrating its ability to achieve pure-Differential Privacy while retaining local applicability. Based on these findings, the dissertation advocates for using the SaS privacy mechanism in protecting the privacy of client data.

Federated Learning is a sub-field of Machine Learning, which trains Machine Learning models across a collection (*federation*) of client devices. This approach aims to protect client privacy by limiting the type of information that clients transmit to the server. However, this distributed environment poses challenges such as non-uniform data distributions and inconsistent client update rates, which reduces the accuracy of trained models. To overcome these challenges, we introduce Federated Inference, a novel algorithm that we show is consistent in federated environments. That is, even when the data is unevenly distributed and the clients' responses to the server are staggered in time (asynchronous), the algorithm is able to converge to the global optimum.

We also present a novel result in system identification in which we extend a method known as Dynamic Mode Decomposition to accommodate input delayed systems. This advancement enhances the accuracy of identifying and controlling systems relevant to privacy-sensitive applications such as smart grids and autonomous vehicles.

Privacy is increasingly pertinent, especially as investments in computer infrastructure constantly grow in order to cater to larger client bases. Privacy failures impact an ever-growing number of individuals. This dissertation reports on our efforts to advance the toolkit of data privacy tools through novel methods and analysis while navigating the challenges of the field.

STUDIES IN DIFFERENTIAL PRIVACY AND FEDERATED LEARNING

by

Christopher Cameron Zawacki

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2024

Advisory Committee:
Professor Eyad H. Abed, Chair/Advisor
Professor John S. Baras
Professor André L. Tits
Professor Richard J. La
Professor Radu V. Balan

Acknowledgments

First and foremost, I thank my research advisor, Eyad Abed, for aiding me through the PhD process. He has been instrumental in helping me develop my ideas into concise works that clearly communicate the advancements that have been made.

I would especially like to thank my soon-to-be wife, Mika (3.5 weeks and counting!), for always listening when I needed to talk through a problem (even if it was at 6am before coffee), for understanding why "When are you graduating?" is not an easy question to answer, and for planning the majority of our wedding. I don't know how I'll ever make it up to you, but I look forward to trying ♡.

To the rest of my family, thank you for getting me to where I am today. I would not have made it this far without your unconditional support. But also, thank you for reminding me that I do, in fact, need to graduate at some point.

And thank you to all my friends who carried me through the harder challenges of this dissertation. I could always count on someone to be there when I needed to step away from a problem so that I could approach it again with fresh eyes. Whether that be through taking a break with a quick game or listening to my presentations and providing feedback; this work is better because of them all the same.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
AMSGrad	Adaptive Moment Estimation with Gradient Descent
BO	Bayesian Optimization
CCPA	California Consumer Privacy Act
CDP	Concentrated Differential Privacy
CIFAR	Canadian Institute For Advanced Research
CLT	Central Limit Theorem
CNN	Convolutions Neural Network
CPRA	California Privacy Rights Act
DL	Deep Learning
DMD	Dynamic Mode Decomposition
DMDc	Dynamic Mode Decomposition with Control
DMDidc	Dynamic Mode Decomposition with Input Delayed Control
DNN	Deep Neural Network
DP	Differential Privacy
DP-SGD	Differentially Private Stochastic Gradient Descent
EEA	European Economic Area
ESN	Echo State Network
EU	European Union
FedAvg	Federated Averaging
FedInfer	Federated Inference
FedLin	Linear Federated Learning
FedProx	Federated Proximal Learning
FedSGD	Federated Stochastic Gradient Descent
FL	Federated Learning
FNA	Fine Needle Aspirates
FTS	Federated Thompson Sampling
GDPR	General Data Protection Regulation
HE	Homomorphic Encryption
HIPAA	Health Insurance Portability and Accountability Act
IID	Independent and Identically Distributed
IoT	Internet of Things
LDP	Local Differential Privacy
LGPD	Brazilian General Data Protection Law
LLN	Law of Large Numbers
LTI	Linear Time-Invariant

ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology database
MOON	Model-based Constrastive Learning
MPC	Multiparty Computation
MSE	Mean Squared Error
P2P	Peer-to-Peer
PRV	Privacy Random Variable
SaS	Symmetric alpha-Stable
SCAFFOLD	Stochastic Controlled Averaging for Federated Learning
SNR	Signal-to-Noise Ratio
SOTA	State-of-the-Art
SVM	Support Vector Machine
RBF	Radial Basis Function
RC	Reservoir Computing
RDP	Rényi Differential Privacy
ReLU	Rectified Linear Unit
RFF	Random Fourier Features
WBC	Wisconsin Breast Cancer dataset

Chapter 1

Introduction

The works collected in this dissertation primarily focus on the field of data privacy. The results constitute an effort to increase the effectiveness of existing methods through new insights and to present novel methods that mitigate the challenges of the field. In this section, we discuss the need for data privacy and outline the structure of the thesis.

1.1 The Need for Data Privacy

Data privacy stands as a cornerstone of individual autonomy, safeguarding rights and personal safety, while nurturing trust in institutions and fostering innovation. In today's digital landscape, the sheer volume of data we generate is staggering. Current estimates suggest that the average internet user generated 51.13Tb in the year 2023 [1]. The migration of services to online platforms enhances accessibility but heightens the risk of privacy breaches. In 2021, according to data from the Identity Theft Resource Center, the US experienced over 1800 known data breaches [2]. On a more global scale, a security researcher found a database that contained the information of more than 500 million world wide Facebook users [3]. This database was publicly available for anyone to access. As early as 2007, studies have shown that attackers attempt

to breach internet connected devices as many as $2,805$ times per day, which equates to an attack roughly every 39 seconds [4]. The number of attacks per day is likely much higher today. Both public and private sectors have initiated responses: political leaders are enacting policies to safeguard citizen privacy [5, 6], and consumers are advocating for privacy-centric practices among companies [7, 8].

The push for data privacy was sparked in the 1990s by an event in Massachusetts. Motivated by advances in Machine Learning, the Massachusetts Group Insurance Commission made the decision to release anonymized state employee medical records with the aim of supporting the discovery of novel medicines. Names, addresses, Social Security numbers— were all scrubbed to prevent re-identification, or so they thought; however these precautions would prove to be inadequate.

In 1996, Governor Bill Weld collapsed at a public event. In the heat of the moment, no one realized that this event, along with the policy on health data, would thrust the issue of data privacy into the spotlight. Seeing an opportunity, Latanya Sweeney, then a graduate student at the Massachusetts Institute of Technology (MIT), capitalized on the vulnerability of the ostensibly anonymized data. By cross-referencing public health records with a readily accessible list of voter registrations she had acquired for a mere $20, Sweeney re-identified the governor's entire medical history. In a bold statement on data privacy, she mailed the governor's recent diagnoses and prescription directly to his office, dramatically highlighting the peril of presumed anonymity [9]. Sweeney's actions directly led to important changes in data privacy laws and helped influence the Health Insurance Portability and Accountability Act (HIPAA) [10].

In more recent years, both governmental and private entities have responded to increased public pressure by enacting legislation and privacy policies to address these concerns and provide

individuals with greater control over their personal information. The California Consumer Privacy Act (CCPA), passed in 2018, gives consumers more control over the personal information and regulates how businesses can collect data [11]. The law stipulates the rights that citizens have regarding data collected about them, including knowing the content of the collection, the ability to request the deletion of personal information, and the ability to opt out of the sale or sharing of their personal data. The bill was recently amended, in November of 2020, with the California Privacy Rights Act (CPRA) that added the right to correct inaccurate personal information and to limit the use of specific sensitive data [12]. In 2018, the European Union (EU) enacted the most wide-ranging privacy regulation to date, the General Data Protection Regulation (GDPR) [6]. The bill regulates the processing of personal data of individuals within the EU, European Economic Area (EEA) and applies to businesses outside the EU/EEA that offer goods or services to individuals within the EU/EEA. In 2020, Brazil took inspiration from the GDPR and passed the Brazilian General Data Protection Law (LGPD) [13]. These laws introduce comprehensive rules for the processing of personal data, establish privacy rights for individuals, and impose obligations on businesses to ensure compliance.

In the private sector, the Brave and Firefox browsers have distinguished themselves as leaders of user privacy, offering built-in features that prioritize security and anonymity [8, 14]. Both browsers come with ad, cookie, and tracking blockers by default. Additionally, the technology company Apple has championed privacy policies, particularly with its introduction of app tracking transparency measures [15]. The impact of these measures have reverberated throughout the tech ecosystem, profoundly impacting data collection practices, particularly for companies like Meta, which has lost an estimated $12 billion in annual revenue from the change [16]. The implementation of these policies has compelled companies to reevaluate their data collection practices

and provide users with greater transparency and control over their personal data.

The academic sector has led the way in the design of privacy systems. Techniques like Differential Privacy (DP) obfuscate database queries to safeguard individual data points from adversarial agents [17]. Federated Learning (FL) is a technique that trains Machine Learning models without centralizing client data, preventing a single breach from compromising the entire database of personal information [18]. DP and FL are among the more prominent state-of-the-art privacy protection methods being actively developed in research.

1.2 The Significance of Privacy in Applications

As Machine Learning (ML) progresses and finds increasing use in commercial contexts, the importance of prioritizing privacy-conscious approaches becomes ever clearer. In this section, we delve into several applications where protecting privacy is paramount.

1.2.1 Medical collaboration and patient privacy

In the medical domain, sharing data across institutions is essential for research and improving patient outcomes. However, this practice raises legitimate concerns regarding patient privacy. Patients are understandably cautious about sharing sensitive medical information, underscoring the need for robust privacy-preserving mechanisms. By guaranteeing certain levels of privacy, patients may be more willing to participate in medical studies.

1.2.2 Fraud detection in banking

Financial institutions possess a wealth of sensitive client information that they use to detect fraudulent activities. While leveraging artificial intelligence (AI) can bolster fraud detection, collaborative efforts to improve detection models are often hampered by privacy concerns. By taking client privacy into consideration, financial institutions can more readily train multi-institutional models that are more accurate at detection fraudulent transactions.

1.2.3 Cyber-physical systems

Cyber-physical systems (CPS) are the integration of computation, communication and control with physical components, serving to monitor and control various real-world processes. These systems are integral to numerous technologies, such as smart grids, transportation systems, and industrial automation. Privacy should play a crucial role in the design and implementation of these systems. Collected data from CPS can be used identify and track individuals without their permission. Ultimately, prioritizing privacy in CPS design and deployment is vital for preserving both individual rights and societal trust in modern technologies.

1.2.4 Recommendation systems

Recommendation systems power personalized advertising, necessitating the collection and analysis of user data. However, the pervasive collection of personal data raises significant privacy concerns among users. As seen from patterns in the use of Apple's new tracking controls [15], consumers often do not consent to being tracked. As a result, in order to convince consumers to allow their data to be used in recommendation systems, companies must reassure their user base

that their data will be private and secure.

1.3 Overview of Contributions

The main objective of this research is to provide a better understanding of the complex interplay between Machine Learning techniques and data privacy measures and to improve the accuracy and efficiency of these methods. We examine state-of-the-art data privacy methods and extend them, achieving significant performance improvements. This section outlines our principal contributions. In the associated chapters, each contribution is supported by analysis, numerical evaluations, and accompanying simulations. We emphasize the distinctive insights and solutions introduced, illuminating their broader implications and potential applications within the field of data privacy.

1.3.1 The SaS mechanism

In Part I of the dissertation, we introduce the first in depth analysis of the Symmetric alpha-Stable (SaS) privacy mechanism, demonstrating its ability to achieve pure-Differential Privacy while retaining locally applicability. Based on these finding, the dissertation advocates for using the SaS privacy mechanism in protecting the privacy of client data. Differential Privacy is one of the major prevailing methods for protecting client information. The method achieves privacy through injecting carefully calibrated noise into a dataset to obscure specific client details while leaving general trends recoverable. Within this context, the choice of probability density function used to generate the noise significantly influences the attained level of privacy. Traditional approaches often opt for the Gaussian density due to its well-known and favorable characteristics. In

the chapters of Part I, we introduce, analyze, and empirically assess the novel Symmetric alpha-Stable privacy mechanism. The alpha-Stable densities are a family of probability densities that encompass all densities that are closed under convolution, the Gaussian density included. This family of densities retains the desirable attributes of the Gaussian distribution while enhancing overall privacy protection due to a heavier tail. First, we prove that the SaS privacy mechanism achieves pure-Differential Privacy. Then, through numerical evaluations, we compare the outcomes obtained using the Gaussian and SaS densities when training Deep Neural Networks and showcase substantial privacy improvements that can be achieved by using heavier tails.

1.3.2 Federated Inference

Aside from Differential Privacy, another method to enhance user privacy is through decentralized networks of clients, known as federations, which circumvent the inherent privacy loss that occurs by sending sensitive data to a central server. A notable example is the social media platform Mastodon, which enables users to create and connect to personal forums that are not hosted by large technology firms [19]. The firms that host centralized applications typically offset hosting costs by inserting ads into clients' feeds and selling user data to online data brokers [20, 21]. Federated Learning, a subarea of Machine Learning, involves training advanced models on a distributed set of clients, which maintain ownership over their own data. We propose a novel method for Federated Learning, which we call Federated Inference (FedInfer). FedInfer draws upon insights from Differential Privacy to offer clients measurable privacy guarantees. The method mitigates the challenges traditionally associated with FL approaches that arise from the distributed nature of the federated environment. We provide convergence analysis for the method

and present sample simulations using test datasets.

1.3.3 Dynamic Mode Decomposition for Input Delayed Systems

Data privacy methods can additionally be deployed to data streams utilized for system identification. Many applications now require users to share private data with third-party apps for processing and decision-making. Some example include smart grids, health monitoring, recommendation systems, traffic monitoring, fuel optimization, and industrial control systems in cloud computing [22–24]. Recommendation systems, such as the Netflix Competition, have been shown to allow re-identification of individuals from anonymized data [25]. In a smart grid, customers might share power consumption data for better rates, inadvertently revealing appliance usage and daily activities [26, 27]. It is for these reasons that robust privacy-preserving mechanisms are vital to address user concerns and increase participation, enhancing system efficiency.

The results of this section pertain to the types of systems that can be accurately identified and used with existing privacy techniques for linear systems. In this work, we extend the capabilities of a method of system identification known as Dynamic Mode Decomposition (DMD) [28]. The aim of Dynamic Mode Decomposition is to identify the higher energy spatio-temporal frequencies from data. The method can further be used to obtain a reduced-order model that characterizes the dominant elements of a system's dynamics. However, when input delay is present in a control system, whether it be a private or a non-private system, the stability of the controller can be impacted. We extend Dynamic Mode Decomposition, to simultaneously identify the plant, control dynamics, and the input delay present in the control scheme. While DMD inherently discretizes continuous plants, the method we develop returns a continuous input delay estimate.

1.4 Thesis Organization

While the results collected in this work generally relate to the field of data privacy, they target different frameworks to accomplish the task. As such, the thesis is split into three parts: Part I: Differential Privacy: Part II Federated Learning, and Part III: System Identification.

Part I reports our work on Differential Privacy. Chapter 2 provides an overview of the mathematical foundation that underpins DP and discusses the advancements that have been proposed since its inception. Chapter 3 introduces our novel privacy mechanism, the Symmetric alpha-Stable Privacy (SaS) mechanism. This chapter offers the first in-depth study of the method and discovery of associated privacy guarantees. Supported by rigorous proofs and analyses, we compare the method's performance against existing state-of-the-art mechanisms. Chapter 4 follows with a thorough examination of numerical results and simulations, which build on Chapter 3 while extending the results to Machine Learning applications.

Part II collects our results in the field of Federated Learning. In Chapter 5, we discuss the how FL aims to enhance user privacy through decentralization. We present a selection of existing FL methods and discuss the challenges and limitations faced by the field. Following this review, we present, in Chapter 6, a case study on how federated methods can fall short regarding the protection of sensitive data. To address these issues, Chapter 7 introduces our proposed Federated Learning approach, Federated Inference (FedInfer), which aims to mitigate the challenges of Federated Learning. Chapter 8 we compare FedInfer against a selection of existing Federated Learning solutions and discuss the benefits and limitations of our method.

Part III is devoted to a result in the field of Control Theory. In Chapter 9, we present work on system identification that generalizes an identification method known as Dynamic Mode

Decomposition (DMD) to systems with input delay.

We conclude the thesis with Chapter 10, where we summarize the results of our work and suggest avenues for future research.

Part I

Differential Privacy

Chapter 2

Background for Differential Privacy

Differential Privacy (DP), introduced by Cynthia Dwork in 2006, has emerged as one of the primary methods of preserving privacy during data collection and release, as well as in training machine learning models [17, 29, 30]. This section recalls the mathematical framework underpinning Differential Privacy and provides an overview of the advances that have been developed since its inception.

2.1 Overview of Differential Privacy

Differential Privacy is a framework designed to protect individuals' privacy during data analysis by ensuring that their participation in a dataset does not unduly impact their privacy, especially when the dataset is publicly released [17, 31]. To understand the need for Differential Privacy, one must first consider what is meant by the *infringement of one's privacy*. Naively, one may first assume that if the public release of a study negatively impacts the participants, then their privacy has been breached. Consider an individual who is a known smoker who willingly chooses to participate in the study that resulted in the first published correlation linking smoking to lung cancer. This result may inadvertently lead to increased insurance premiums. However,

because it was known before the study that the individual smoked, the negative result of the study would have impacted them regardless of their choice to partake in the study. As a result, this is not an example of a breach in privacy.

Instead, Differential Privacy seeks to prevent an adversarial agent from re-identifying individuals within a dataset. In the example from Chapter 1, Sweeney was able to cross-correlate unrelated datasets to extract the health records of the then sitting Massachusetts governor. To this end, DP introduces noise into the dataset, which limits the amount of information an adversary may gain about a particular target, ensuring that individuals' privacy remains protected. The objective of the injected noise is to prevent an adversary from learning information about the specific individuals that make up the dataset while allowing general trends in the dataset to remain apparent. This approach provides a principled balance to data utility and individual privacy, ensuring valuable insights can be gleaned from collective datasets without compromising confidentiality.

DP finds applications across diverse domains, including large-scale data analytics, machine learning, and statistical databases, as well as in online advertising platforms, social network analysis, and genomic data analysis [32–34]. Recently, it has garnered attention in Federated Learning (FL), where it enhances privacy during process of training Machine Learning models collaboratively across decentralized devices or servers. For instance, Li et al. [35] utilize Differentially Private Federated Learning for brain tumor image segmentation, while Song et al. [36] apply it in location-based services for privacy preservation. Differential Privacy techniques have additionally been applied to the systems such as smart grids and autonomous vehicles, which require end-user participants to continuously send information to external aggregators [37]. The incorporation of Differential Privacy into machine learning promotes a more ethical and secure

environment for handling sensitive data, particularly in fields such as healthcare, finance, and social sciences. As machine learning continues to influence society, the integration of Differential Privacy with advanced modeling techniques offers promise in effectively managing the trade-off between data utility and individual privacy.

The objective of Differential Privacy is to create datasets (and queries on datasets) that produce nearly indistinguishable results if an individual's data were to be included *or* excluded from the dataset, i.e, the overall analysis remains largely unaffected regardless by the data contributed by a single individual. Moreover, DP can foster more truthful responses in sensitive inquiries. For instance, during the Vietnam War, researchers seeking to determine soldiers' marijuana usage employed a clever DP-inspired technique known as randomized response [38]. Soldiers were randomly given one of two cards, one asking if they used marijuana and the other asking if the sky was blue. By masking the true nature of the question, DP ensured that if a soldier's response were to be disclosed, they could always claim to have answered a benign question, thus reducing the risk of privacy breaches. This example highlights the critical role of DP in fostering trust and confidentiality in data collection and analysis processes, underscoring its importance in modern privacy-conscious society.

2.2 Applications of Differential Privacy

Differential Privacy has emerged as a cornerstone in the protection of individual privacy in data analysis, finding applications across a diverse array of fields. From enhancing the confidentiality of census data [39–41] to securing user data in machine learning models [42–44], Differential Privacy offers a robust framework for sharing insights gleaned from data without

compromising the privacy of the individuals represented in that data. Its versatility is evident in sectors such as healthcare, where it protects sensitive patient information, in finance, where it safeguards transaction data, and in social science research, where it enables the study of human behavior without exposing personal details.

2.2.1 Census data

Differential Privacy has revolutionized the way census data is managed and utilized, offering a novel approach to preserving the privacy of individuals while still allowing for the rich analysis of population trends. In the context of national censuses, which collect sensitive information from millions of individuals, ensuring privacy is paramount. DP provides a mathematical framework that guarantees the confidentiality of respondents' data, ensuring that the information released for public and research use cannot be exploited to identify any individual participant.

The U.S. Census Bureau's adoption of DP for the 2020 Census marked a significant milestone, underscoring the growing importance of privacy-preserving techniques in the digital age [41]. By applying DP techniques, statistical agencies can add carefully calibrated noise to the census data, masking the contributions of individual respondents without significantly distorting the overall accuracy of the data. This process allows for the publication of detailed demographic, socio-economic, and geographic information that is essential for policy making, urban planning, and community services development, among other applications [40, 45]. DP enables the benefits of census data to be harnessed in a manner that respects and safeguards individual privacy, ensuring that the invaluable insights derived from such data do not come at the cost of personal confidentiality. This careful balancing act facilitates the continued trust and participation of the

public in census surveys, which are critical for the democratic process and the effective allocation of resources.

2.2.2 Medical and healthcare models

In the field of medical research and healthcare services, the protection of patient data is not just a legal requirement but a cornerstone of ethical practice. Differential Privacy offers a robust solution to maintianing patient privacy by ensuring that the medical data released for research or public health analysis cannot be traced back to individual patients. This approach allows researchers to access vital health data for epidemiological studies, drug effectiveness research, and health policy planning while maintaining the confidentiality of patient information [46–48].

The sensitive nature of medical data often limits the extent to which it can be shared among researchers, institutions, and countries [49]. Differential Privacy provides a framework that facilitates safer data sharing and collaboration, enabling the pooling of data resources to tackle global health challenges. By ensuring that the privacy of data contributors is protected, DP encourages the sharing of information, which is crucial for advancing medical research and improving patient care on a global scale.

Utilizing DP in handling healthcare data plays a crucial role in enhancing public trust. When individuals are assured that their personal health information is handled with the utmost care for their privacy, they are more likely to support and participate in medical research projects. This trust is vital for the success of public health initiatives and for the advancement of medical science.

2.2.3 Cyber-Physical Systems

Cyber-physical systems (CPS) are the integration of computational elements with physical processes, permeating various aspects of modern life from smart grids to autonomous vehicles. They rely on interconnected networks and computing to monitor, analyze, and control physical processes in real-time. CPS have the ability to enhance efficiency, productivity, and safety.

CPS often handle sensitive data, such as personal information generated by Internet of Things (IoT) devices monitoring application usage within a smart power grid [50]. Protecting this data from unauthorized access or misuse is essential to uphold individuals' privacy and prevent potential harm. The interconnected nature of CPS exposes clients to a myriad of security risks, including data breaches, malware attacks, and system manipulations. Robust privacy measures are crucial to mitigate these risks, safeguarding both data integrity and individuals' privacy.

Fostering user trust is paramount in CPS adoption and utilization [51]. When individuals feel confident that their data is protected, they are more likely to engage with and rely on these systems. This trust is foundational for the widespread acceptance and integration of CPS into various domains. By prioritizing privacy considerations, CPS can effectively harness the benefits of interconnected technologies while upholding individuals' rights and interests.

2.2.4 Deep Learning

While deep learning is able to generate impressively comples models, it generally requires a massive amount of information for training. Differentially Private Stochastic Gradient Descent (DP-SGD) represents a significant advancement in the realm of privacy-centric algorithms, specifically tailored for the training of machine learning models with a focus on maintaining the

confidentiality of client training data. Originating from the foundational work detailed in [43], DP-SGD extends the traditional stochastic gradient descent optimization to incorporate Differentially Private update steps. Their modifications enhances model parameters based on gradients of the loss function calculated from a randomly chosen subset of the training dataset, while simultaneously injecting noise to protect the subset of selected individuals.

Traditional stochastic gradient descent's straightforward approach, while effective for numerous machine learning tasks, does not inherently protect the privacy of individual data points. In contrast, DP-SGD introduces a critical layer of privacy protection by integrating principles of Differential Privacy directly into the optimization process. DP-SGD works by first modifying the gradient calculation step. It introduces a clipping mechanism that limits the magnitude of each gradient update, ensuring that each step remains within a predefined ℓ_p norm, thus preventing any single data point from having a disproportionately large influence on the model's updates. Next, the algorithm injects noise to the clipped gradient updates. The noise is carefully drawn from a pre-selected probability distribution, the characteristics of which are aligned with the desired privacy guarantees. The amplitude and nature of this injected noise are controlled by key privacy parameters, including the privacy budget, denoted by ε, and the accepted error parameter, δ (defined later in this chapter). These parameters play a pivotal role in balancing the trade-off between privacy protection and the model's learning efficacy.

Training a model with DP-SGD makes it significantly more challenging to infer characteristics of individual training samples from the model's parameters or outputs. This ensures that the training of machine learning models can proceed without compromising the privacy of the data subjects represented in the training set. As a result, DP-SGD has become a cornerstone technique for developers and researchers seeking to harness the power of machine learning while adhering

to stringent privacy requirements.

2.3 Mathematical Foundations of Differential Privacy

Differential Privacy operates on a collaboratively constructed dataset, which we denote by $\mathcal{D}$. Conceptually, we can think of such a dataset as a table of records, where each row represents a set of client data. Denote by f a function that operates on this dataset and produces a vector of m numerical values. For instance:

- How many clients have blue eyes?
- What is the average income of all clients?
- What are the optimized parameters of a given machine learning model across all clients?

Differential Privacy primarily prevents *passive* adversaries from obtaining information about a target client. A passive adversary is one that observes communications or interactions without actively tampering with them. For example, a passive adversary may eavesdrop on communication between a client and the server or combine publicly available datasets in a linking attack to re-identify anonymized client data. This is in contrast with an active adversary which seeks to directly disrupt model training.

By a slight abuse of notation, we use the symbol f for the query, despite the composition of the dataset,

Definition 1. *(Query) A function f is termed a query if it takes a dataset $\mathcal{D}$ as input and outputs a vector in $\mathbb{R}^m$:*

$$f : \mathcal{D} \to \mathbb{R}^m. \tag{2.1}$$

◀

The types of queries commonly employed in Differential Privacy methods are those exhibiting finite ℓ_p-sensitivity [17, 31]:

Definition 2. *(ℓ_p-Sensitivity of Query) The ℓ_p-sensitivity of a query f, denoted $\Delta_p f$, is defined to be a maximum of a $p-$norm over the domain of f, $dom(f)$:*

$$\Delta_p f := \max_{\mathcal{D}_1 \simeq \mathcal{D}_2} ||f(\mathcal{D}_1) - f(\mathcal{D}_2)||_p, \tag{2.2}$$

for all $\mathcal{D}_1, \mathcal{D}_2 \in dom(f)$. ◀

From Definitions 1 and 2 above, it is clear that when the sensitivity of f is bounded, the range of the query is also bounded. While focusing on finite queries here, ongoing research aims to extend Differentially Private methods to handle queries with unbounded ranges [52].

Now, we recall the definition of a privacy mechanism, which introduces stochastic noise the result of a query.

Definition 3. *(Privacy mechanism) A privacy mechanism for the query f, denoted $\mathcal{M}_f$, is defined to be a randomized algorithm that returns the result of the query perturbed by a vector of i.i.d. noise sampled from pre-selected densities Y_i,*

$$\mathcal{M}_f(\mathcal{D}) = f(\mathcal{D}) + (Y_1, Y_2, \ldots, Y_m)^T. \tag{2.3}$$

◀

To simplify notation, we denote the resulting vector, $\mathcal{M}_f(\mathcal{D})$, as $\mathbf{x} \in \mathbb{R}^m$. Note that the

noise variables, Y_i, induce a density, which we occasionally denote $p = p(\mathbf{x})$ for $\mathcal{M}_f$, on a given dataset $\mathcal{D}$. Although not strictly necessary, we assume that the injected density is symmetric about the origin, simplifying the analysis.

The privacy mechanism aims to hinder an adversary from conclusively ascertaining the presence of a specific client within the dataset.

Definition 4. *(Neighboring Datasets) Two datasets, denoted $\mathcal{D}_1$ and $\mathcal{D}_2$, are known as neighboring datasets if they differ in the presence or absence of exactly one client record. We denote this relation as $\mathcal{D}_1 \simeq \mathcal{D}_2$.* ◀

This concept is visualized in Figure 2.1 , which depicts two datasets, one that contains the red client, and one that does not. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ represent these two scenarios respectively. To proceed, let us assume the red client has allowed their data to be included in the set and that $\mathcal{D}_1$ is the *true* dataset. Denote a realization of a mechanism as $x \sim \mathcal{M}_f(\mathcal{D}_a)$. Informally, the mechanism $\mathcal{M}_f$ is said to be Differentially Private if the inclusion or exclusion of a single individual in the dataset, illustrated in red in the figure, results in *essentially* the same distribution over the realized outputs x,

$$\Pr[\mathcal{M}_f(\mathcal{D}_1) = x] \approx \Pr[\mathcal{M}_f(\mathcal{D}_2) = x]. \tag{2.4}$$

Differential Privacy then quantifies what *essentially* means mathematically:

Definition 5. *(Pure Differential Privacy) Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be any neighboring datasets. Given a query f that operates on $\mathcal{D}_1$ and $\mathcal{D}_2$, a privacy mechanism $\mathcal{M}_f$ is said to be ε-Differentially*

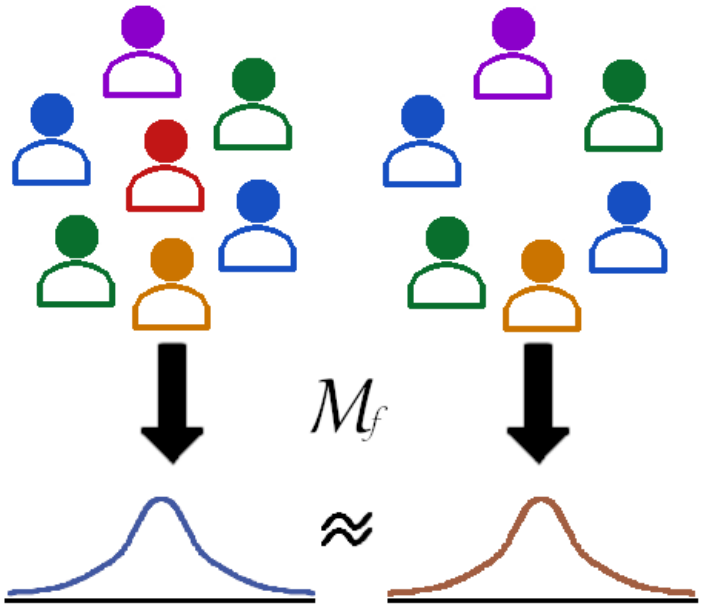

Figure 2.1: In order to protect client identity, Differential Privacy injects noise into the output of a query f on a dataset. This induces a probability density over possible outcomes. A mechanism, $\mathcal{M}_f$, is considered private, if the resulting distributions are *essentially* the same regardless of the inclusion or exclusion of a single client, shown here in red. Differential Privacy quantifies how much information an adversary is able to gain about the red client.

Private (ε-DP) if it satisfies

$$\Pr[\mathcal{M}_f(\mathcal{D}_1) \in \mathcal{X}] \leq e^{\varepsilon} \Pr[\mathcal{M}_f(\mathcal{D}_2) \in \mathcal{X}] \tag{2.5}$$

for some $\varepsilon > 0$ and any subset of outputs $\mathcal{X} \subseteq \mathcal{R}(\mathcal{M}_f(\mathcal{D}_1))$. The mechanism is defined to have no privacy ($\epsilon = \infty$) if, upon its application to each dataset, the supports of the resulting densities are not equal, viz. $\mathcal{R}(\mathcal{M}_f(\mathcal{D}_1)) \neq \mathcal{R}(\mathcal{M}_f(\mathcal{D}_2))$.

◀

The parameter ε is also referred to as the privacy budget. Smaller values of ε are, in general, associated with stronger privacy. We remark that when $\varepsilon = 0$, the definition yields perfect privacy. However, in that case, adding more client data results in no new information.

Note that Eq. 2.5 holds for each element when the density of the distributions is considered [31]:

Theorem 2.3.1. *(Privacy as Densities) Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be neighboring datasets and f be a query that operates on them. Denote by p_1 and p_2 the densities of the privacy mechanism $\mathcal{M}_f$ when applied to $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. Then, a privacy mechanism $\mathcal{M}_f$ is ε-Differentially Private if*

$$p_1(x) \leq e^{\varepsilon} p_2(x), \quad \forall x \in \mathcal{R}(\mathcal{M}_f(\mathcal{D}_1)) \tag{2.6}$$

for all $\mathcal{D}_1 \simeq \mathcal{D}_2$.

Next, we give a brief proof for this known fact.

Proof. Begin by writing condition (2.5) in terms of the generated densities,

$$\int_{\mathcal{X}} p_1(x)dx \leq \int_{\mathcal{X}} e^{\epsilon} p_2(x)dx. \tag{2.7}$$

Equation (2.7) can be rewritten as

$$0 \leq \int_{\mathcal{X}} e^{\epsilon} p_2(x) - p_1(x)dx. \tag{2.8}$$

Noting that (2.6) enforces the integrand in (2.8) to be non-negative, implying that (2.5) is satisfied.

□

Some privacy mechanisms fail to satisfy condition (2.5), but the condition can be relaxed through the inclusion of an additive constant $\delta > 0$, as in the following definition:

Definition 6. *(Approximate Differential Privacy) Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be any neighboring datasets. Given a query f that operates on $\mathcal{D}_1$ and $\mathcal{D}_2$, a privacy mechanism $\mathcal{M}_f$ is said to be (ε, δ)-Differentially Private if it satisfies*

$$\Pr[\mathcal{M}_f(\mathcal{D}_1) \in \mathcal{X}] \leq e^{\varepsilon} \Pr[\mathcal{M}_f(\mathcal{D}_2) \in \mathcal{X}] + \delta. \tag{2.9}$$

This is known as approximate-Differential Privacy.

◀

The accepted error term δ is the probability that the result of the query provides more information to the adversary than expected from the bound ε.

One common modification relates to who applies the privacy mechanism. Up to this point, we have considered a mechanism in relation to a query over the entire dataset $\mathcal{D}$. It is then understood that the mechanism is applied by a *trusted aggregator*, who collects the clients' data prior to obfuscation. However, there does not always exist such a trusted central authority. For example, in a Federated Learning framework, the server is assumed untrustworthy by default. Another situation where clients may with to apply noise locally is if they lack secure communication protocols. In this case, the clients' sensitive information could be leaked to an adversary during the transmission between the clients and server. To this end, a mechanism $\mathcal{M}_f$ is said to be Locally Differentially (LDP) Private if the mechanism can be applied locally by the clients prior to transmission to the server.

Definition 7. *(Local Differential Privacy) Let a client apply the privacy mechanism $\mathcal{M}_f^{loc}$ to their local dataset $\mathcal{D}$. The mechanism $\mathcal{M}_f^{loc}$ is said to be locally differentially privacy if, for any pair*

of datapoints $v_1, v_2 \in \mathcal{D}$, it satisfies the following [53]:

$$\Pr[\mathcal{M}_f^{loc}(v_1) \in \mathcal{X}] \leq e^{\varepsilon} \Pr[\mathcal{M}_f^{loc}(v_2) \in \mathcal{X}] + \delta, \tag{2.10}$$

for all $\mathcal{X} \in \mathcal{R}(\mathcal{M}_f^{loc})$.

◀

The mechanism is called ε-LDP if $\delta = 0$ and (ε, δ)-LDP otherwise.

2.4 Connection to Detection Theory

A common condition when it comes to estimating privacy bounds is to assume the worst case scenario. As a result, any bounds calculated hold regardless of the adversary's capabilities. Consider an adversary that has nearly complete knowledge of all participants in a dataset $\mathcal{D}$ except for their target. The objective of the adversary is to determine if the target is or is not in the dataset, which we denote by $\mathcal{D}_1$ and $\mathcal{D}_0$ respectively. We formalize the choice by the following hypotheses:

- H_0 (The null hypothesis): the true dataset does not contain the target, $\mathcal{D} = \mathcal{D}_0$.
- H_1 (The alternative hypothesis): the true dataset contains the target, $\mathcal{D} = \mathcal{D}_1$.

Let p_0 be the initial probability that the adversary assumes the null hypothesis is true, $p_0 = \Pr[\mathcal{D} = \mathcal{D}_0]$. Likewise, the initial suspicion that the alternative hypothesis is correct is given by $p_1 = 1 - p_0 = \Pr[\mathcal{D} = \mathcal{D}_1]$. Let the adversary observe the output of a query f on $\mathcal{D}$ that has been passed through a privacy mechanism $\mathcal{M}_f(\mathcal{D})$. Let the decision rule of the adversary be

that they report H_0 is true if the result $\mathcal{M}_f(\mathcal{D})$ lies within a set S, otherwise they report H_1 is true.

We calculate bounds on the adversary's updated probability using Bayes' rule,

$$\Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S] = \frac{\Pr[\mathcal{D} = \mathcal{D}_1] \cdot \Pr[\mathcal{M}_f(\mathcal{D}) \in S | \mathcal{D} = \mathcal{D}_1]}{\Pr[\mathcal{M}_f(\mathcal{D}) \in S]}. \tag{2.11}$$

We defined the first term on the right hand side, $\Pr[\mathcal{D} = \mathcal{D}_1]$, to be the adversary's initial estimation of the alternative hypothesis p_1. The next term can be simplified by

$$\Pr[\mathcal{M}_f(\mathcal{D}) \in S | \mathcal{D} = \mathcal{D}_1] = \Pr[\mathcal{M}_f(\mathcal{D}_1) \in S)]. \tag{2.12}$$

The final term, $\Pr[\mathcal{M}_f(\mathcal{D}) \in S]$, is the probability that the output of the mechanism falls in the adversary's decision set S. Without more specifics as to the decision boundary of the set, this term is unknown. However, we can eliminate it by instead considering the ratio between probability updates,

$$\frac{\Pr[\mathcal{M}_f(\mathcal{D}) \in S | \mathcal{D} = \mathcal{D}_1]}{\Pr[\mathcal{M}_f(\mathcal{D}) \in S | \mathcal{D} = \mathcal{D}_0]} = \frac{\Pr[\mathcal{D} = \mathcal{D}_1] \cdot \Pr[\mathcal{M}_f(\mathcal{D}) \in S | \mathcal{D} = \mathcal{D}_1]}{\Pr[\mathcal{D} = \mathcal{D}_0] \cdot \Pr[\mathcal{M}_f(\mathcal{D}) \in S | \mathcal{D} = \mathcal{D}_0]} \tag{2.13}$$

We then substitute Equation (2.13) into the definition of Differential Privacy from (2.5) to get the following relation

$$e^{-\varepsilon} \cdot \frac{\Pr[\mathcal{D} = \mathcal{D}_1]}{\Pr[\mathcal{D} = \mathcal{D}_0]} \leq \frac{\Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S]}{\Pr[\mathcal{D} = \mathcal{D}_0 | \mathcal{M}_f(\mathcal{D}) \in S]} \leq e^{\varepsilon} \cdot \frac{\Pr[\mathcal{D} = \mathcal{D}_1]}{\Pr[\mathcal{D} = \mathcal{D}_0]} \tag{2.14}$$

Substitute $1 - p_1$ and p_1 for $\Pr[\mathcal{D} = \mathcal{D}_0]$ and $\Pr[\mathcal{D} = \mathcal{D}_1]$ respectively,

$$e^{-\varepsilon} \cdot \frac{p_1}{1 - p_1} \leq \frac{\Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S]}{\Pr[\mathcal{D} = \mathcal{D}_0 | \mathcal{M}_f(\mathcal{D}) \in S]} \leq e^{\varepsilon} \cdot \frac{p_1}{1 - p_1}. \tag{2.15}$$

Additionally, note that the conditional probability of the central term can be substituted as follows,

$$\frac{\Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S]}{\Pr[\mathcal{D} = \mathcal{D}_0 | \mathcal{M}_f(\mathcal{D}) \in S]} = \frac{\Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S]}{1 - \Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S]}. \tag{2.16}$$

Equations (2.15) and (2.16) give the following relation,

$$\Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S] \leq \frac{e^{\varepsilon} p_1}{1 + (e^{\varepsilon} - 1) p_1}. \tag{2.17}$$

The lower bound can be solved for similarly,

$$\frac{p_1}{e^{\varepsilon} + (1 - e^{\varepsilon}) p_1} \leq \Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S]. \tag{2.18}$$

Inequalities (2.17) and (2.18) together give the following bounds on the adversary's updated probability

$$\frac{p_1}{e^{\varepsilon} + (1 - e^{\varepsilon}) p_1} \leq \Pr[\mathcal{D} = \mathcal{D}_1 | \mathcal{M}_f(\mathcal{D}) \in S] \leq \frac{e^{\varepsilon} p_1}{1 + (e^{\varepsilon} - 1) p_1}. \tag{2.19}$$

Figure 2.2 depicts the upper and lower bounds for a given privacy budget ε and initial probability p_1. For example, let the adversary assume equal probability between the null and alternative hypotheses, $p_0 = p_1 = 0.5$ and assume the privacy budget is $\varepsilon = 1$. Then, given the outcome of the privacy mechanism applied to the true dataset $\mathcal{M}_f(\mathcal{D})$, it is possible that the adversary gains enough information to update their estimation of the likelihood of the alternative

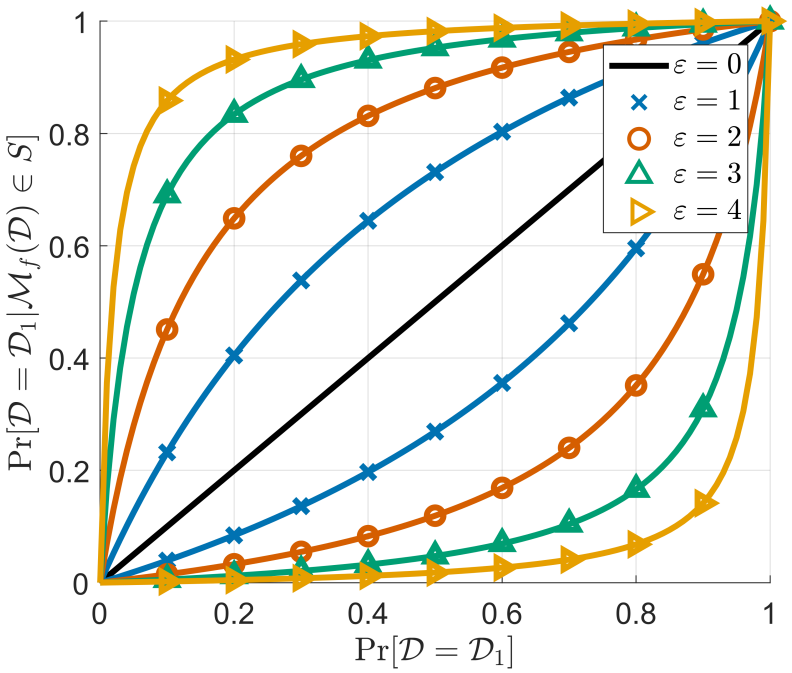

Figure 2.2: Pure-Differential Privacy limits the amount of information an adversary can gain from the outcome of private query. Based on the adversary's initial estimate of the alternative hypothesis, $\Pr[\mathcal{D} = \mathcal{D}_1]$, a Differentially Private mechanism bounds the conditional probability given the outcome of the query. Each pair of matching curves represents the lower and upper bound for an adversary's estimate of the alternative hypothesis after observing the outcome of the privacy mechanism. As the privacy budget ε is increased, the bound of the adversary's updated estimate is increased.

hypothesis to as high as 73% or as low as 27%. Recall that in this scenario, we additionally assumed that the adversary had access to all data in the dataset except for the targets, making these bounds the absolute worst case scenario.

An alternative method to determining a sufficient privacy budget was established in [54,55] which relates the privacy budget ε to the false positive and false negative rates of the adversary.

A false positive indicates that the adversary chose H_1 when the null hypothesis was true,

$$p_{FP} := \Pr[\mathcal{M}(D_0) \in S]. \tag{2.20}$$

A false negative occurs when the adversary reports the null hypothesis when the alternative hy-

pothesis is correct,

$$p_{FN} := \text{Pr}[\mathcal{M}(D_1) \notin S]. \tag{2.21}$$

By construction $\mathcal{D}_0$ and $\mathcal{D}_1$ are neighbors and so by the definition of Differential Privacy given in Equation (2.5) we have

$$\begin{aligned} \text{Pr}[\mathcal{M}_f(\mathcal{D}_1) \in S] &\leq e^{\varepsilon}\text{Pr}[\mathcal{M}_f(\mathcal{D}_0) \in S] \\ \text{Pr}[\mathcal{M}_f(\mathcal{D}_0) \notin S] &\leq e^{\varepsilon}\text{Pr}[\mathcal{M}_f(\mathcal{D}_1) \notin S]. \end{aligned} \tag{2.22}$$

Substituting (2.20) and (2.21) then provides the following relations

$$\begin{aligned} e^{\varepsilon}p_{FP} + p_{FN} &\geq 1 \\ p_{FP} + e^{\varepsilon}p_{FN} &\geq 1. \end{aligned} \tag{2.23}$$

Combining the inequalities in (2.23) yields

$$p_{FP} + p_{FN} \geq \frac{2}{1 + e^{\varepsilon}}. \tag{2.24}$$

For example, if high privacy is required, $\varepsilon = 0.1$, and we assume a 1% false positive rate, then by (2.24) the false negative rate is bounded below by 94%. Often, it is easier to specify false positive and negative rates to determine a sufficient privacy budget.

2.5 Common Privacy Mechanisms

The choice of probability density from which the noise is drawn significantly impacts the level of privacy achieved. To prevent bias from being introduced into the query, all mechanisms

considered here are chosen to have zero mean.

One of the most common mechanisms is the Laplace mechanism, which samples noise from the Laplace density [31, 56]:

$$p_{\text{Lap}}(x) = \frac{1}{2b} e^{\frac{-|x|}{b}}. \tag{2.25}$$

Here, b determines the spread of the distribution. The Laplace mechanism has been shown to satisfy pure Differential Privacy, equation (2.5) [31, 56]. Unfortunately, the Laplace density does not trivially extend to local Differential Privacy, limiting its application in methods such as Federated Learning. This property additionally makes the Laplace mechanism challenging to use for training Neural Networks, which rely heavily on repeated compositions of mechanism, so it is not commonly employed for deep learning .

Another frequently employed mechanism is the Gaussian mechanism [31, 56]. This mechanism injects noise, drawn from a normal density with a mean of zero, into the output of a query:

$$p_{\text{Gaus}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x}{\sigma})^2}. \tag{2.26}$$

Since the Gaussian density is closed under convolution, the mechanism naturally extends to environments that require local application of the mechanism. However, the Gaussian mechanism is only approximately Differentially Private, i.e., it requires $\delta > 0$ in equation (2.9). Traditionally, this has not been seen as an issue because adaptive composition scales better than simple composition when considering the error introduced by the injected noise [31].

The Exponential mechanism is another noteworthy approach. This mechanism selects out-

puts from a set probabilistically, weighting them according to their utility scores [57]. By carefully choosing the scoring function, it provides a way to balance privacy and utility effectively. The noise added by the Exponential mechanism is drawn from the exponential density:

$$p_{\text{Exp}}(x) = \lambda e^{-\lambda x}. \tag{2.27}$$

Here, λ controls the rate of decay of the distribution. However, designing appropriate scoring functions that accurately capture utility while ensuring privacy remains a significant challenge in practical implementations.

In the work, we present novel analysis of a privacy mechanism that can be applied locally like the Gaussian, but that provides stronger privacy guarantees like the Laplace.

2.6 Composition

One notable advantage of the Differential Privacy framework is its ability to maintain a quantifiable level of privacy even when multiple private queries are sequentially combined [56, 58]. This property, known as closure under composition, ensures that the privacy guarantees remain intact regardless of the number of queries performed on the dataset. As such, DP mechanisms provide a reliable means of protecting sensitive information while enabling meaningful data analysis. In this section, we recall a selection of composition methods that have progressively improved the estimation of the privacy bound over the years since Differential Privacy's introduction.

2.6.1 Simple composition

The simplest method of composition is derived directly from sequential applications of pure-Differential Privacy (2.5), which provides a loose upper bound for the achievable level of privacy:

Theorem 2.6.1. *(Simple Composition of ε-DP mechanisms [56, 58]) Let $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_k$ be a set of, possibly distinct, Differentially Private mechanisms, with privacy $\varepsilon_1, \varepsilon_2, ..., \varepsilon_k$ respectively. Then their composition $\mathcal{M}(D) := (\mathcal{M}_1(D), \mathcal{M}_2(D), ..., \mathcal{M}_k)$ has a privacy budget of $\sum_{i=i}^{k} \varepsilon_i$.*

Proof. The result follows directly from Eq. (2.5):

$$\begin{aligned}
\frac{\Pr[\mathcal{M}(\mathcal{D}_0) \in \mathcal{X}]}{\Pr[\mathcal{M}(\mathcal{D}_1) \in \mathcal{X}]} &= \frac{\Pi_{i=1}^{k} \Pr[\mathcal{M}_i(\mathcal{D}_0) \in \mathcal{X}_i]}{\Pi_{i=1}^{k} \Pr[\mathcal{M}_j(\mathcal{D}_1) \in \mathcal{X}_i]} \\
&= \Pi_{i=1}^{k} \frac{\Pr[\mathcal{M}_i(\mathcal{D}_0) \in \mathcal{X}_i]}{\Pr[\mathcal{M}_j(\mathcal{D}_1) \in \mathcal{X}_i]} \\
&\leq \Pi_{i=1}^{k} e^{\varepsilon_j} \\
&= e^{\sum_{i=1}^{k} \varepsilon_j}.
\end{aligned} \tag{2.28}$$

□

The statement remains true even if the second mechanism is chosen given the result of the first mechanism. This is known as adaptive composition, denoted by

$$\mathcal{M}_{\text{adapt}}(D) = \Big(\mathcal{M}_1(D), \mathcal{M}_2\big(D, \mathcal{M}_1(D)\big)\Big).$$

A similar result has been shown for approximately private mechanisms, i.e., if two mechanisms are $(\varepsilon_1, \delta_1)$ and $(\varepsilon_2, \delta_2)$ respectively, then their composition is $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP [31, 59]. If a privacy mechanism is adaptively composed with itself k times, denoted by $\mathcal{M}^{\circ k}$, Theorem

2.6.1 gives a privacy bound of $(k\varepsilon, k\delta)$. While simple composition applies to both pure and approximate-DP mechanisms, the relaxation to (ε, δ)-DP allows for a significant reduction in the estimated privacy bound through a composition technique known as advanced composition [31, 59].

Theorem 2.6.2. *(Advanced Composition [31, 59]) Let $\mathcal{M}_f$ be an (ε, δ)-Differentially Private mechanism. Then the k-fold adaptive composition of $\mathcal{M}_f$ is $(\varepsilon', k\delta + \delta')$-DP where*

$$\varepsilon' = \varepsilon\sqrt{2k\log\left(\frac{1}{\delta'}\right)} + k\varepsilon(e^{\varepsilon} - 1). \tag{2.29}$$

◀

Advanced composition results in a $\sqrt{k}$-factor improvement in estimated privacy over simple composition.

One of the main issues with approximate-Differential Privacy is that the definition allows for inherently non-private algorithms to be considered private. For example, consider a dataset $\mathcal{D}$ that contains data on n individuals. Let $\mathcal{M}$ be a mechanism such that for some query request, it returns 0 with probability $(n-1)/n$ and returns the entire dataset with probability $1/n$. By definition, this mechanisms is $(0, 1/n)-$-DP. In order to improve the composition results further, modifications to the definition of approximate-Differential Privacy are required.

Recent extensions to Differential Privacy seek to remedy this issue by weighting the portion of the density induced by the privacy mechanism that surpasses the ε bound. One method to address this issue is Concentrated Differential Privacy.

2.6.2 Concentrated Differential Privacy

To begin, we recall the concept of privacy loss:

Definition 8. *(Privacy Loss) Denote by $\mathcal{D}_1$ and $\mathcal{D}_2$ two neighboring datasets. For some privacy mechanism $\mathcal{M}_f$, let p_1 and p_2 be the induced density when applied to $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. The privacy loss of an outcome x is defined to be the log-ratio of the densities at x [31]:*

$$\mathcal{L}_{\mathcal{D}_1||\mathcal{D}_2}(x) := \ln \frac{p_1(x)}{p_2(x)}. \tag{2.30}$$

◀

By (2.6), it is evident that ε-Differential Privacy (2.5) is equivalent to a strict bound on the privacy loss:

$$|\mathcal{L}_{\mathcal{D}_1||\mathcal{D}_2}(x)| \leq \varepsilon, \quad \forall x \in \mathcal{R}(\mathcal{M}_f(\mathcal{D}_1)) \tag{2.31}$$

for all neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. For mechanisms that are purely-Differentially Private, the privacy budget ε is the maximum over all observations x,

$$\varepsilon = \max_{x \in \mathbb{R}} \mathcal{L}_{\mathcal{D}_1||D_2}(x). \tag{2.32}$$

Using privacy loss as a basis for composition, the authors of [60] introduced a variant of Differential Privacy, named Concentrated Differential Privacy (CDP), which greatly simplified the composition of privacy mechanisms.

Definition 9. *(Concentrated Differential Privacy [60]) Let $\mathcal{M}_f$ be a privacy mechanism that*

operates on all neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$. We say $\mathcal{M}_f$ satisfies ρ-concentrated Differential Privacy (ρ-CDP) if the privacy loss distribution $\mathcal{L}_{\mathcal{D}_1||\mathcal{D}_2}$ is well defined and

$$\mathbb{E}_{Z\leftarrow\mathcal{L}_{\mathcal{D}_1||\mathcal{D}_2}(x)}[\exp(tZ)] \leq \exp(t(t+1)\cdot\rho), \quad \forall t \geq 0. \tag{2.33}$$

◀

The main advantage of Concentrated Differential Privacy is the ease of composition:

Theorem 2.6.3. *(Composition for Concentrated Differential Privacy [60]) Let $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_k$ be a set of privacy mechanisms where each mechanism is ρ_i-CDP respectively. Let $\mathcal{M} = (\mathcal{M}_1(\mathcal{D}), \mathcal{M}_2(\mathcal{D}), ..., \mathcal{M}_k(\mathcal{D}))$ represent the composition of each mechanisms applied independently to a query on a dataset $\mathcal{D}$. Then $\mathcal{M}$ is $\sum_{i=1}^{k} \rho_i$-CDP.*

◀

2.6.3 Rényi Differential Privacy

To prevent inherently non-private algorithms from being considered private, more recent modifications to Differential Privacy have sought to remedy by weighting the portion of the density induced by the privacy mechanism that surpasses the ε bound.

One major contribution was the introduction of Rényi Differential Privacy (RDP), which both a simple composition rule and prevented obviously non-private algorithms. RDP arose from the application of Differential Privacy techniques applied to the training of deep neural networks [43]. RDP is formulated in terms of the Rényi divergence:

Definition 10. *(Rényi Divergence [61]) Let P and Q be distributions over Ω with P being ab-*

solutely continuous with respect to Q. The Rényi divergence between P and Q is then given by

$$D_\alpha(P||Q) = \frac{1}{\alpha - 1} \log \mathbb{E}_{Y \leftarrow Q}\left[\left(\frac{P(Y)}{Q(Y)}\right)^\alpha\right], \quad \alpha \in (1, \infty). \tag{2.34}$$

◀

Rényi Differential Privacy is then given as

Definition 11. *(Rényi Differential Privacy [62]) Let $\mathcal{M}_f$ be a privacy mechanism that operates on all neighboring datasets $\mathcal{D}_0$ and $\mathcal{D}_1$. We say $\mathcal{M}_f$ satisfies ρ-Concentrated Differential Privacy (rho-CDP) if the privacy loss distribution $\mathcal{L}_{\mathcal{D}_0||\mathcal{D}_1}$ is well defined and*

$$D_\alpha(\mathcal{M}_f(\mathcal{D}_0)||\mathcal{M}_f(\mathcal{D}_1)) \leq \varepsilon. \tag{2.35}$$

◀

Similar to Concentrated Differential Privacy, RDP allows for a simple composition rule:

Theorem 2.6.4. *(Composition of RDP [62]) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two privacy mechanisms that are $(\alpha_1, \varepsilon_1)$-RDP and $(\alpha_2, \varepsilon_2)$-RDP respectively. Let $\mathcal{M} = \mathcal{M}_2(\mathcal{D}, \mathcal{M}_1(\mathcal{D}))$ represent the adaptive composition of the mechanisms. Then $\mathcal{M}$ is $(\min\{\alpha_1, \alpha_2\}, \varepsilon_1 + \varepsilon_2)$-RDP.*

◀

RDP is closely related to CDP; specifically, ρ-CDP is equivalent to $(\alpha, \alpha \cdot \rho)$-RDP for all $\alpha \in (1, \infty)$. In practice, RDP has proven quite useful in the training of differnetially private Deep Neural Networks (DNN).

2.6.4 Privacy Loss Random Variables

Recently, methods have been proposed that approximate tight privacy bounds rather than estimating upper bounds. To this end, optimal guarantees for the k-fold composition of an (ε, δ)-mechanism were first given by Kairouz et al. in [63]. These results were extended to the composition of possibly distinct mechanisms by Murtagh et al. in [64]. The calculation is computationally intensive, so recent efforts have sought to develop algorithms to calculate the privacy to an arbitrary accuracy with improved computation time [60, 63, 65–70].

Frequently, a privacy mechanism is associated with a family of (ε, δ)-DP guarantees. That is, for each privacy budget $\varepsilon \in \mathbb{R}_+$, there exists a $\delta(\varepsilon) \in [0, 1]$ that satisfies (2.9). The function $\delta(\varepsilon)$ is known as the privacy curve:

Definition 12. *(Privacy Curve [65]) Let $\mathcal{M}_f$ be a privacy mechanism for a given query f and let $\mathcal{D}_1 \simeq \mathcal{D}_2$ be two neighboring datasets. Denote the range of $\mathcal{M}_f$ by Ω. The privacy curve $\delta(\mathcal{M}_f(\mathcal{D}_1)||\mathcal{M}_f(\mathcal{D}_2)) : \mathbb{R} \to [0, 1]$ is given by*

$$\begin{aligned} &\delta\big(\mathcal{M}_f(\mathcal{D}_1)||\mathcal{M}_f(\mathcal{D}_2)\big)\big(\varepsilon\big) := \\ &\sup_{S \subset \Omega} \Pr[\mathcal{M}_f(\mathcal{D}_2) \in S] - e^{\varepsilon} \Pr[\mathcal{M}_f(\mathcal{D}_1) \in S]. \end{aligned} \tag{2.36}$$

◀

Like DP-mechanisms, privacy curves can be composed as in the next definition.

Definition 13. *(Composition of Privacy Curves [65]) Let δ_1 and δ_2 be any two privacy curves.*

The composition of the privacy curves, denoted by $\delta_1 \otimes \delta_2$ is given as:

$$\begin{aligned} &\delta_1 \otimes \delta_2 \doteq \\ &\delta\Big(\big(\mathcal{M}_1(\mathcal{D}_1), \mathcal{M}_2(\mathcal{D}_1)\big) || \big(\mathcal{M}_1(\mathcal{D}_2), \mathcal{M}_2(\mathcal{D}_2)\big)\Big), \end{aligned} \tag{2.37}$$

where $\mathcal{M}_1(\mathcal{D}_1)$ and $\mathcal{M}_2(\mathcal{D}_1)$ are independently sampled and $\mathcal{M}_1(\mathcal{D}_2)$ and $\mathcal{M}_2(\mathcal{D}_2)$ are independently sampled. ◀

In 2019, Dong et al. proposed f-DP, a dual method to privacy curves that provides tight bounds using a lossless composition theorem [65]. Using this tool, they found that Differential Privacy achieves significantly better privacy than prior bounds had indicated. Their results allow the necessary level of noise to be reduced for a given privacy budget, improving the accuracy of DP-models. The composition of a set of privacy mechanisms is given by the composition of their corresponding privacy curves:

Theorem 2.6.5. *(Composition Theorem [65]) Let $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_k$ be privacy mechanisms for a given query f (with the notation for f suppressed). Denote by $\delta_1(\varepsilon), \delta_2(\varepsilon), ..., \delta_k(\varepsilon)$ their respective privacy curves. Then, the adaptive composition $\mathcal{M}_k \circ \mathcal{M}_{k-1} \circ ... \circ \mathcal{M}_1$ is given by $\delta_1(\varepsilon) \otimes \delta_2(\varepsilon) \otimes ... \otimes \delta_k(\varepsilon)$.* ◀

Because training a Deep Neural Network requires a large number of compositions, the exact calculation is often computationally intractable. To address this issue, Meiser et al. developed an algorithm called *Privacy Buckets* which approximates the composition of privacy curves through their relation with Privacy Loss Random Variables (PRV) [71].

Definition 14. *(Privacy Loss Random Variable [70]) Let $\mathcal{M}_f$ be a privacy mechanism and de-*

note by δ the associated privacy curve. Let X and Y be two random variables with probability densities p_X and p_y respectively. We call (X, Y) Privacy Loss Random Variables for δ if the privacy curve constructed from X and Y, denoted by $\delta(X||Y)$ from (2.36), is equivalent to δ and

- *p_X and p_Y are supported on $\mathbb{R} \cup \{-\infty, \infty\}$*
- *$p_Y(t) = e^t p_X(t)\ \forall t \in \mathbb{R}$*
- *$\lim_{t\to\infty} p_Y(-t) = \lim_{t\to\infty} p_X(t) = 0$.*

◀

Reference [70] additionally provides an approach to construct PRVs that satisfy the conditions of definition 14 from the densities induced over the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ by the privacy mechanism $\mathcal{M}_f$. The advantage of using PRVs is that the composition is given by their convolution:

Theorem 2.6.6. *(Composition as Convolution [70]) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two privacy mechanisms with privacy curves δ_1 and δ_2 respectively and denote by (X_1, Y_1) and (X_2, Y_2) their associated Privacy Loss Random Variables as constructed in [70]. Then the composition of the associated privacy curves $\delta_1 \otimes \delta_2$ is given by*

$$\delta_1 \otimes \delta_2 = \delta(X_1 + X_2||Y_1 + Y_2), \tag{2.38}$$

i.e., the privacy curve constructed from the convolution of the PRVs.

◀

In Chapter 4, we utilize the algorithm designed by Gopi et al. [70] to efficiently computes privacy curves to an arbitrary accuracy. Their method approximates the privacy loss random

variables by truncating and discretizing to facilitate application of the Fast Fourier Transform to convolve the densities efficiently.

2.7 Subsampling

We next recall a method of privacy amplification known as Subsampling, first shown for Differentially Private mechanisms in [53]. Subsampling entails computing the query on a subset sampled from the original dataset. For Neural Networks, this is often related to the selection of mini-batches, which are subsampled datasets used to approximate the the gradient step. Clearly, an individual receives perfect privacy if their data is not included in the batch. Different types of subsampling have been used to increase the privacy of different methods [60, 72–75]. In our work, we use the subsampling described in Appendix B of [70].

Let P and Q be the densities induced by applying a privacy mechanism $\mathcal{M}_f$ to neighboring datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively. Denote by $p \cdot P + (1-p) \cdot Q$ the mixture where the set with denisty P is sampled with probability p and the set with density Q is sampled with probability $(1-p)$. Then, the Privacy Loss Random Variables adjusted for subsampling are given by

Proposition 1. *(PRV Subsampling [70]) Let $\mathcal{M}_f$ be a privacy mechanism and denote by δ the associated privacy curve. Denote by X and Y the PRVs as in definition (14). Denote by X_p and Y_p the PRVs for $\delta_p = \delta(P||p \cdot P + (1-p) \cdot Q)$ where p is the sampling probability. Then*

$$\begin{aligned} X_p &= \log(1 + p(e^X - 1)) \\ Y_p &= \begin{cases} \log(1 + p(e^Y - 1)) & w.p.\ \ p \\ \log(1 + p(e^X - 1)) & w.p.\ \ (1-p). \end{cases} \end{aligned} \tag{2.39}$$

The cumulative distribution functions of X_p and Y_p are given by

$$
\begin{aligned}
CDF_{X_p}(t) &= \begin{cases} CDF_X\Big(\log(\frac{e^t-(1-p)}{p})\Big) & t \geq \log(1-p) \\ 0 & t < \log(1-p) \end{cases} \\
&CDF_{Y_p}(t) \\
&= \begin{cases} p \cdot CDF_Y\Big(\log(\frac{e^t-(1-p)}{p})\Big) + \\ (1-p) \cdot CDF_Y\Big(\log(\frac{e^t-(1-p)}{p})\Big) & t \geq \log(1-p) \\ 0 & t < \log(1-p). \end{cases}
\end{aligned} \tag{2.40}
$$

◀

2.8 Limitations and Alternative Privacy Methods

One of the most fundamental limitations faced by Differentially Private methods is balance between privacy protection and data utility [76]. The introduction of noise to protect privacy can degrade the quality of the output, making it less useful for analysis. Striking the right balance requires careful tuning of privacy parameters, which can be complex and context-dependent. Too much noise can render the data useless, while too little may not offer sufficient privacy protection. On this note, the effectiveness of DP relies heavily on the choice of privacy parameters, such as the privacy budget ε and the accepted error δ. This frequent decision to use the Gaussian mechanism for deep learning algorithms necessarily require a $\delta > 0$. Selecting these parameters involves a trade-off between privacy and accuracy, and there is no one-size-fits-all solution. Additionally, selecting too high a privacy budget may mislead clients into assuming the under-

lying model is private when in reality, the training offers little to no protection if the amount of noise is too low. While DP provides strong theoretical guarantees, in practice, the privacy it offers can still be undermined by side-channel attacks, auxiliary information, or advances in data analysis techniques. Adversaries with access to additional information might be able to infer protected attributes, highlighting the need for continuous research into stronger or more adaptive privacy-preserving methods.

2.8.1 Secure Multiparty Computation

Secure Multiparty Computation (MPC) is a cryptographic technique that enables multiple parties to jointly compute a function over their private inputs while keeping those inputs confidential [77–79]. Unlike traditional computation methods where data is shared or centralized, MPC allows computations to be performed on encrypted data in a distributed manner, ensuring privacy and security. By leveraging cryptographic protocols such as secret sharing, and zero-knowledge proofs, MPC enables parties to collaborate on computations without revealing sensitive information to each other [80–82].

MPC often incurs a large communication overhead as multiple rounds of communication are required between participating parties. For this reason, multiparty computation is difficult to scale to very large numbers of parties. Moreover, MPC is susceptible to malicious collusion between clients who can use their combined information to eavesdrop on target clients. Unlike Differential Privacy, MPC cannot be used when the output is the dataset, for example the 2020 US Census.

2.8.2 Algebraic Transformation

Similar to MPC, certain types of optimizations allow algebraic manipulation to disguise the original problem while still solving for an equivalent solution [83,84]. Algebraic transformations as a form of privacy are promising because they do not trade accuracy for privacy. Instead, a private transformation function, is used to obfuscate the underlying problem from eavesdropping. For example, this could allow a client to perform an optimization on a cloud service, without the service learning the true parameters of the optimization.

Algebraic transformation works best for problems with a clear modification. However, if an adversary is able to use a linkage attack to estimate what the true dataset might be, it is possible for an adversary to reverse the transformation and directly recover client data. Similarly, as in the MPC case, if multiple parties collude, it is often possible to recover the client information of a specific target. As with MPC, Algebraic transformations cannot do not protect individuals when the output is the dataset.

2.8.3 Homomorphic Encryption

Homomorphic encryption (HE) is a technique in the field of cryptography that allows computations to be performed directly on encrypted data without the need for decryption [85–87]. This means that sensitive information can be kept confidential even while undergoing processing by third parties or in untrusted environments. This capability has profound implications for privacy-preserving data analysis, secure outsourcing of computations to the cloud, and collaborative data sharing while maintaining confidentiality.

Homomorphic Encryption is a cutting edge field of research and currently does not support

many types of operations. Frequently, this means complex computations must be decomposed resulting in increased computation overhead. Moreover, to reduce the computational load, approximation techniques inject noise into the cryptographic process, distorting the result of the calculations [88].

Chapter 3

The Symmetric alpha-Stable Privacy Mechanism

In this chapter, we develop a novel privacy mechanism, which we refer to as the Symmetric alpha-Stable (SaS) privacy mechanism. The motivation for this new mechanism stems from the deficiencies found in two of the most common mechanisms, the Laplace and Gaussian mechanisms. In the following sections, we detail the properties that existing methods lack, namely closure under convolution and pure-Differential Privacy, and prove that the SaS mechanism satisfies both of these conditions.

3.1 Motivation

Differential Privacy mechanisms are the critical component in DP methods. The choice of distribution balances the delicate trade-off between data utility and individual privacy protection. DP mechanisms serve as the tools through which data analysts and researchers can access aggregated insights from sensitive datasets while preserving the confidentiality of individual contributions. However, the stronger the privacy guarantee, the less accuracy the mechanism's results. Moreover, different privacy mechanisms induce different densities over the database queries, which each have unique properties that make them more or less suited to certain situations.

As recalled in Section 2.5 the Laplace and Gaussian densities constitute two of the main privacy mechanisms in Differential Privacy. The Laplace mechanism is the quintessential ε-Differentially Private mechanism. The Laplace mechanism injects zero-mean Laplace noise into the result of a query. The Laplace density extends the exponential density to the whole real line by taking the absolute value of the observation x:

$$p_{Lap}(x; b) = \frac{1}{2b} e^{-\frac{|x|}{b}}. \tag{3.1}$$

The parameter b controls the spread of the density. When b is taken to be $\Delta_1 f/\varepsilon$, its is shown in [17] that the Laplace mechanism satisfies (2.5), pure-Differential Privacy.

There are a few drawbacks to choosing the Laplace mechanism. First, note that the Laplace density is not closed under convolution. Because of this, the Laplace mechanism is not commonly used for Local Differential Privacy applications, where the clients apply the mechanism to their local dataset prior to aggregation on the server. Modifications that extend the method to Local Differential Privacy (see Definition 7) have been shown to be strictly sub-optimal [89, 90]. Additionally, the Laplace density is not smooth. The lack of gradient at the inflection point increases the computational complexity in some settings. Conveniently, the Gaussian mechanism does not suffer from either of these issues.

The Gaussian mechanism, as the name suggests, perturbs the output of the query with noise drawn from a Gaussian density. One major benefit is the ease with which Gaussian perturbations fit into existing Machine Learning analyses. The Gaussian density owes its pervasiveness to its essential role in the Central Limit Theorem (CLT) [91, Thm. 27.1]. One important property is that the density is that it is closed under convolution. This means that two Gaussian estimates can

be combined and the result remains Gaussian. The Gaussian mechanism perturbs a query result with zero-mean Gaussian noise:

$$p_{Gau}(x;\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x}{\sigma}\right)^2} \tag{3.2}$$

When the variance is taken to be $\sigma^2 = 2\ln(1.25/\delta)\Delta_2^2 f/\varepsilon^2$, the Gaussian mechanism satisfies (2.9), approximate Differential Privacy [30, 31]. This fact highlights the main drawback in using the Gaussian mechanism: it is not purely Differentially Private so may release more information to an adversary than intended.

We have recalled the Laplace and Gaussian mechanisms here to underscore deficiencies that are addressed through use of the SaS mechanism. First, we prove in this chapter that the SaS mechanism is purely Differently Private as opposed to the approximate Differential Privacy satisfied by the Gaussian mechanism. Second, due to its use of a stable density, the aggregation of datasets individually perturbed by the SaS mechanism can be described as if the aggregate itself has instead been perturbed, in contrast to the Laplace mechanism.

3.2 Background

To begin, it is essential to note that the Gaussian density belongs to a broader family of distributions called the Lévy alpha-Stable densities, all of which exhibit closure under convolutions [92]. However, in the realm of Differential Privacy, the Gaussian mechanism only adheres to condition (2.9), reflecting approximate Differential Privacy [30, 31]. This section delves into the privacy characteristics of mechanisms stemming from the non-Gaussian stable densities. Specifically, we introduce the Symmetric alpha-Stable mechanism and provide a proof of its adherence

to condition (2.5), achieving ε-DP.

The concept of stable densities, extensively explored by Lévy in 1925 [92], refers to a particular set of probability distributions. These distributions possess a notable property; they remain unchanged under the operation of convolution.

Definition 15. *(The Stable Family) A probability density function Y is termed stable if, for any constants $a, b > 0$, there exist constants $c(a, b) > 0$ and $d(a, b) \in \mathbb{R}$ such that the following holds for two independent and identically distributed random variables Y_1 and Y_2:*

$$aY_1 + bY_2 = cY + d. \tag{3.3}$$

If d equals 0, the distribution is termed strictly stable.

◀

Except for certain special cases, there is no known closed-form expression for the density of a general stable distribution [93]. Nonetheless, several parameterizations of the characteristic function of a stable density are documented in [93]. One common representation of the characteristic function is as follows:

$$\varphi(t; \alpha, \beta, \gamma, \mu) = \exp(it\mu - |\gamma t|^\alpha + i\beta \text{sgn}(t)\Phi(t)), \tag{3.4}$$

where

$$\Phi(t) = \begin{cases} \tan(\frac{\pi\alpha}{2}) & \alpha \neq 1 \\ -\frac{2}{\pi}\log|t| & \alpha = 1. \end{cases} \tag{3.5}$$

The density function is then given by the integral:

$$p(x;\alpha,\beta,\gamma,\mu) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \varphi(t;\alpha,\beta,\gamma,\mu)e^{-ixt}dt. \tag{3.6}$$

In Figure 3.1, we present three example of the symmetric form, $\beta = 0$: $\alpha = 1$ (blue), $\alpha = 1.5$ (orange), and $\alpha = 2$ (green). Each of the three depicted densities has zero for the location parameter ($\mu = 0$) and unit scale ($\gamma = 1$). The Symmetric alpha-Stable densities with

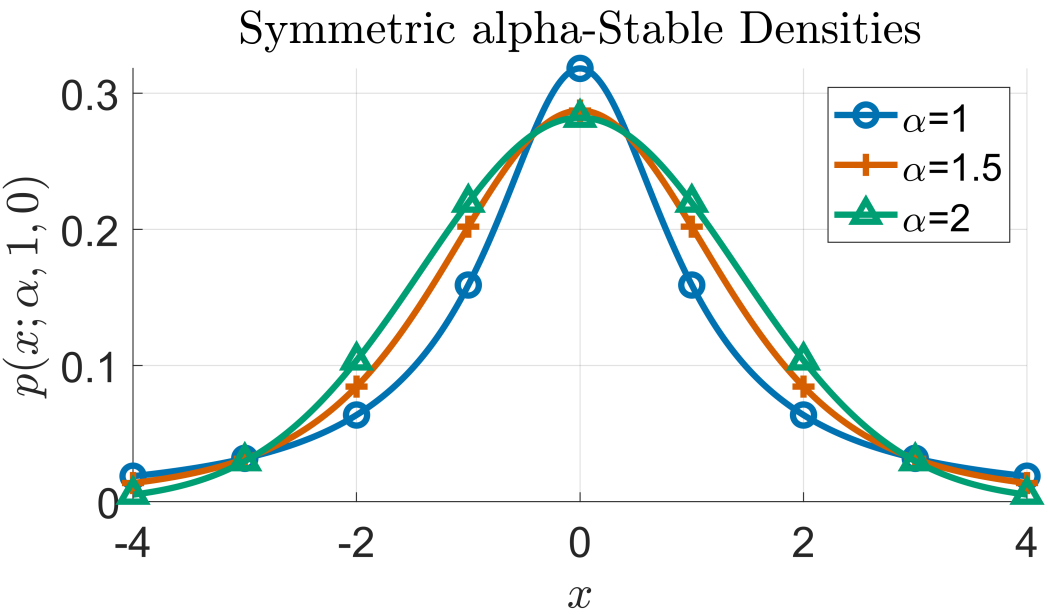

Figure 3.1: The family of Symmetric alpha-Stable densities consists of bell shaped densities with varying tail weights determined by the stability parameter α. This family of densities is unique because it is the only set of densities that are closed under convolution. When $\alpha = 1$, shown in **blue** $\circ$, the density is known as the Cauchy. When $\alpha = 2$, shown in **green** $\triangle$, the density is known as the Gaussian. No other values of alpha (for the symmetric case) have a known closed form solution, for example $\alpha = 1.5$, shown in **orange +**.

$\alpha = 1$ and $\alpha = 2$ are the only two with a known closed form. When $\alpha = 1$, the density is known as the Cauchy density. When $\alpha = 2$, we recover the Gaussian density.

Working with the family of stable distributions presents a significant challenge due to the absence of a closed-form solution for the general density. This difficulty arises because the value at any given point is determined by integrating an infinitely oscillating function.

Denote the real part of the integrand in Equation (3.6) by $q(t; x)$. A visualization of this function with $x = 10$ is illustrated in Figure 3.2.

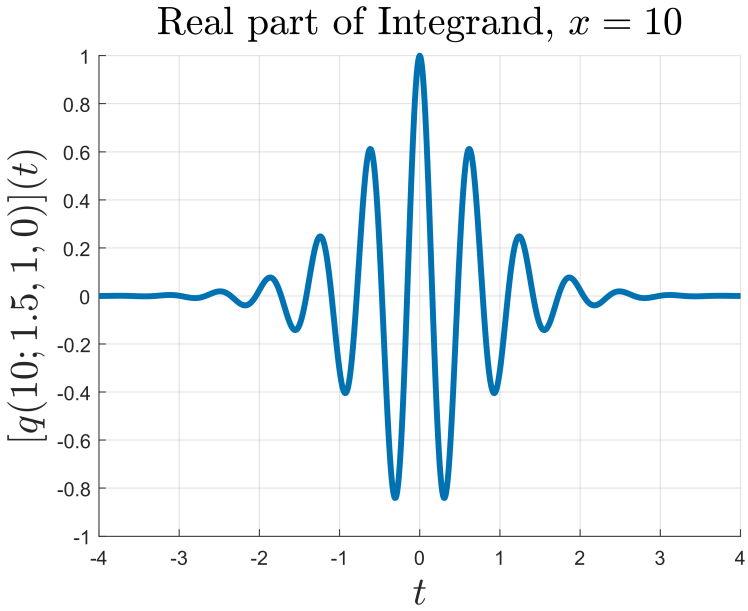

Figure 3.2: The real part of the integrand of (3.6) for $\alpha = 1.5$, $\gamma = 1$, and $\mu = 0$ is an infinitely oscillating function. The value of the stable density with these parameters at the point $x = 10$ is the integral of this function on the real line.

In order for Equation (3.6) to represent a valid probability density, the parameter α must fall within the interval $(0, 2]$. The value of α dictates the rate of decay of the density's tail. The expected value of the density is not defined for $\alpha \leq 1$, but it is defined for $\alpha \in (1, 2]$. Furthermore, the density exhibits infinite variance for $\alpha \in (0, 2)$ and finite variance only when $\alpha = 2$. In this study, we confine α to $(1, 2]$, deferring median or mode estimators for future exploration. The parameter β, normally constrained to $(-1, 1)$, serves as a measure linked to skewness, noting that the strict definition of skewness lacks meaning for $\alpha < 2$. Our focus lies specifically on symmetric alpha-stable (SaS) densities, where $\beta = 0$:

$$\begin{aligned} &p_{SaS}(x; \alpha, \gamma, \mu) := \\ &p(x; \alpha, 0, \gamma, \mu) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-|\gamma t|^{\alpha} - it(x-\mu)} dt. \end{aligned} \tag{3.7}$$

SaS densities have a known closed form for two values of the parameter α: the Cauchy, for $\alpha = 1$, and the Gaussian, for $\alpha = 2$. The last two parameters, $\gamma > 0$ and $\mu \in \mathbb{R}$, are the scale and location parameters respectively.

Remark 1. *For stable densities, it is common for the location parameter to be denoted δ rather than μ, to signify that it is not always equal to the expected value. In our context, we choose to use μ and reserve δ for the definition of approximate Differential Privacy (2.9) as is common in the DP literature. Because we are restricting the domain of interest to $\alpha \in (1, 2]$, where the mean is well defined, we do not believe this notation will be cause for confusion.*

3.3 The Symmetric alpha-Stable mechanism

This leads us to our novel privacy mechanism, the Symmetric alpha-Stable mechanism [94]. While the family of mechanisms is closely related the Gaussian, the Gaussian mechanism only satisfies approximate Differential Privacy. We show in this section that the heaviness of the SaS density's tail allows the privacy mechanism to satisfy pure-Differential Privacy (when $\alpha < 2$).

Definition 16. *(The Symmetric alpha-Stable mechanism) For a given dataset $\mathcal{D}$ and a query function f, we define a privacy mechanism $\mathcal{M}_f$ to be a Symmetric alpha-Stable (SaS) mechanism if each element of the vector of injected values, Y_i for $i \in \{1, ..., m\}$, is drawn independently from a SaS density*

$$\begin{aligned} &p_{SaS}(x; \alpha, \gamma) := \\ &p(x; \alpha, 0, \gamma, 0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-|\gamma t|^\alpha - itx} dt. \end{aligned} \tag{3.8}$$

◀

In this section, we aim to establish that the SaS mechanism, when $\alpha \in [1, 2)$, satisfies (2.5), providing ε-Differential Privacy.

A significant challenge in working with stable densities, excluding the Cauchy and Gaussian, arises from their lack of a closed form. We first introduce the following lemmas before proving that the SaS mechanism achieves ε-DP when α lies in $[1, 2)$.

To ensure that the stable distribution covers the entire real number line, it's crucial to demonstrate that the privacy loss remains finite within a compact set.

Lemma 3.3.1. *(Support of SaS Density) The support of the symmetric alpha-stable density (3.7) is* $\mathbb{R}$.

Proof. See [93, Lemma 1.1]. □

Additionally, we recall a partial sum expansion, as described by Bergström [95], wherein the remainder term possesses a smaller order of magnitude (for large $|x|$) than the final term in the series.

Lemma 3.3.2. *(Finite Series Expansion of SaS Distribution) The symmetric alpha-stable density (3.7), with* $\alpha \in (1, 2]$ *and* $\gamma = 1$*, admits the following finite series expansion:*

$$p_{SaS}(x; \alpha, 1, 0) = \quad -\frac{1}{\pi} \sum_{k=1}^{n} (-1)^k \frac{\Gamma(\alpha k + 1)}{(x)^{\alpha k+1}} \sin\left(\frac{k\alpha\pi}{2}\right) + O\left(x^{-\alpha(n+1)-1}\right), \tag{3.9}$$

as $|x| \to \infty$.

Proof. Bergström's proof [95] offers an expanded form of (3.9) that is valid for the complete

range $\beta \in (-1, 1)$. However, because we have restricted the parameter set to $\beta = 0$, we only require the form provided for our purposes. □

We use the foregoing lemma to argue that the privacy loss remains bounded as the observation $|x|$ tends to infinity. However, Eq. (3.9) is stated for $\gamma = 1$. The next lemma states that the asymptotic behavior of the privacy loss as $|x| \to \infty$ is independent of γ.

Lemma 3.3.3. *(No Scale Dependence in the Limit) Let $\mathcal{D}_1 \simeq \mathcal{D}_2$ be two neighboring datasets. Denote by $\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x;\gamma)$ the privacy loss of observation x for a bounded query f perturbed by a SaS mechanism $\mathcal{M}_f$ with scale parameter γ. In the limit as $|x|$ tends to ∞, the behavior of the privacy loss is indistinguishably asymptotic for distinct choices of γ:*

$$\lim_{|x|\to\infty} \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x;\gamma_1) = \lim_{|x|\to\infty} \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x;\gamma_2), \tag{3.10}$$

for $\gamma_1 \neq \gamma_2$.

Proof.

$$\begin{aligned} p(x;\alpha,\gamma,\mu) &= \frac{1}{2\gamma\pi}\int_{-\infty}^{\infty} e^{-|\hat{t}|^\alpha - i(\hat{x}-\mu_i)\hat{t}}d\hat{t} \\ &= p(\hat{x};\alpha,1,\mu). \end{aligned} \tag{3.11}$$

Substituting (3.11) into the privacy loss function (2.30) gives

$$\begin{aligned} \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x;\gamma) &= \ln \frac{\int_{-\infty}^{\infty} e^{-|\hat{t}|^\alpha - i(\hat{x}-f(\mathcal{D}_1))\hat{t}}d\hat{t}}{\int_{-\infty}^{\infty} e^{-|\hat{t}|^\alpha - i(\hat{x}-f(\mathcal{D}_2))\hat{t}}d\hat{t}} \\ &= \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(\hat{x};1). \end{aligned} \tag{3.12}$$

Observing that $|\hat{x}|$ tends to ∞ as $|x|$ is driven to ∞, we have

$$\lim_{|x|\to\infty} \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x;\gamma) = \lim_{|\hat{x}|\to\infty} \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(\hat{x};1), \quad \forall\gamma. \tag{3.13}$$

In the limit, as $|x|$ tends to infinity, the shift and scale of $\hat{x}_1$ and $\hat{x}_2$ are irrelevant. □

With the above results, we are now in a position to state and prove our main contribution, namely, that for $\alpha \in [1, 2)$, the privacy loss for SaS densities is bounded, i.e. the SaS mechanism is ε-differentially private.

Theorem 3.3.4. *(The SaS mechanism is ϵ-DP) Let $\mathcal{D}_1 \simeq \mathcal{D}_2$ be two neighboring datasets and let f be a bounded query that operates on them. Consider the SaS mechanism, which we denote by $\mathcal{M}_f$, with stability parameter α in the reduced range $\alpha \in [1, 2)$. Then, the mechanism $\mathcal{M}_f$ satisfies (2.5), pure Differential Privacy.*

Proof. Each element of the mechanism's output is the perturbation of the query's response by an independent sample from the uni-variate density in (3.7). Thus, the joint density is equal to the product of the individual densities. As a result, we can write the privacy loss for a given observation vector **x** as

$$\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(\mathbf{x}) = \ln \frac{\prod_{i=1}^{m} p_{SaS}(x_i;\alpha,\gamma,f(\mathcal{D}_1)_i)}{\prod_{i=1}^{m} p_{SaS}(x_i;\alpha,\gamma,f(\mathcal{D}_2)_i)}. \tag{3.14}$$

This can be written as the sum of the log-ratios of the individual elements:

$$\mathcal{L}^{SaS}_{\mathcal{D}_1,\mathcal{D}_2}(\mathbf{x}) = \sum_{i=1}^{m} \ln \frac{p_{SaS}(x_i;\alpha,\gamma,f(\mathcal{D}_1)_i)}{p_{SaS}(x_i;\alpha,\gamma,f(\mathcal{D}_2)_i)}. \tag{3.15}$$

Without loss of generality, let this sum be written in decreasing order of magnitudes of the terms, i.e. the first term, $i = 1$, has the largest magnitude. We now have the following bound:

$$\left|\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(\mathbf{x})\right| \leq m\left|\ln \frac{p_{SaS}(x_1;\alpha,\gamma,f(\mathcal{D}_1)_1)}{p_{SaS}(x_1;\alpha,\gamma,f(\mathcal{D}_2)_1)}\right|. \tag{3.16}$$

Our objective is to prove that the right side of (3.16) is bounded as function of x_1, which will imply, by Theorem 2.3.1, the mechanism is ε-differentially private. We do so by first proving that the privacy loss is bounded on any compact set. Note that this is not immediate, since we are dealing with the log of a ratio and have no assurance that the numerator or denominator ever vanishes. Then, we show that in the limit as $|x|$ tends to infinity, the privacy loss tends to 0, and thus does not diverge.

Initially, let x_1 be an element in a compact set $[a, b] \subset \mathbb{R}$. The log-ratio of the densities could become unbounded within a finite interval in two ways: the argument vanishes or diverges. Consider first the case where one of the densities vanishes within the interval. By Lemma 3.3.1, an SaS density has support on the entire real line, $\mathbb{R}$. Therefore, the density is strictly positive over all compact sets $[a, b] \subset \mathbb{R}$.

Then, we consider if the numerator or denominator of (3.16) could be unbounded within the interval $[a, b]$. For simplicity, let $\mu = 0$ and apply the substitution $e^{-ix_1} = \cos(tx_1) - i\sin(tx_1)$ to the representation of the SaS density (3.7):

$$p_{SaS}(x_1;\alpha,\gamma,0) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-|\gamma t|^\alpha}(\cos(tx_1) - i\sin(tx_1))dt. \tag{3.17}$$

Splitting the integral we have

$$\begin{aligned} p_{SaS}(x_1;\alpha,\gamma,0) =& \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-|\gamma t|^\alpha}\cos(tx_1)dt \\ & -i\frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-|\gamma t|^\alpha}\sin(tx_1)dt. \end{aligned} \tag{3.18}$$

Since $\sin(tx_1)$ is an odd function the second integral vanishes:

$$p_{SaS}(x_1;\alpha,\gamma,0) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-|\gamma t|^\alpha}\cos(tx_1)dt. \tag{3.19}$$

As $\cos(tx_1)$ is bounded above by 1, the density is bounded above:

$$p_{SaS}(x_1;\alpha,\gamma,0) \leq \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-|\gamma t|^\alpha}dt. \tag{3.20}$$

Observe that the integrand in (3.20) is symmetric about $t = 0$, so we can remove the absolute value by adjusting the bounds of the integration:

$$p_{SaS}(x_1;\alpha,\gamma,0) \leq \frac{1}{\pi}\int_{0}^{\infty} e^{-(\gamma t)^\alpha}dt. \tag{3.21}$$

Letting $\hat{t} = (\gamma t)^\alpha$, substitute $\hat{t}$ into the inequality (3.21):

$$\begin{aligned} p_{SaS}(x_1;\alpha,\gamma,0) &\leq \frac{1}{\alpha\gamma\pi}\int_{0}^{\infty} \hat{t}^{\frac{1}{\alpha}-1}e^{-\hat{t}}d\hat{t} \\ &= \frac{\Gamma(\frac{1}{\alpha})}{\alpha\gamma\pi}, \end{aligned} \tag{3.22}$$

where Γ is the standard Gamma function. Note that the Gamma function is finite on the interval $1/\alpha \in (1/2, 1)$ [96]. Equation (3.22) states that the density p_{SaS} is bounded over the real line.

It is therefore bounded on the compact subset $[a, b]$. We proceed to prove that the privacy loss remains bounded in the limit as $|x_1|$ tends to infinity.

Recall the series expansion presented in Lemma 3.3.2, for scale $\gamma = 1$. Truncate the series to a single term by taking $n = 1$ and consider the privacy loss after substitution in (3.16):

$$\left|\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(\mathbf{x})\right| \leq m\left|\ln \frac{\left(x_1 - f(\mathcal{D}_1)\right)^{-\alpha-1} + O(x_1^{-2\alpha-1})}{\left(x_1 - f(\mathcal{D}_2)\right)^{-\alpha-1} + O(x_1^{-2\alpha-1})}\right|. \tag{3.23}$$

In the limit, as $|x_1|$ tends infinity, the error terms in the numerator and denominator are dominated by the first terms:

$$\begin{aligned}
&\lim_{||\mathbf{x}||\to\infty} \left|\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(\mathbf{x})\right| \leq \\
&\lim_{|x_1|\to\infty} m\left|\ln \frac{\left(x_1 - f(\mathcal{D}_1)\right)^{-\alpha-1} + O(x_1^{-2\alpha-1})}{\left(x_1 - f(\mathcal{D}_2)\right)^{-\alpha-1} + O(x_1^{-2\alpha-1})}\right| = \\
&\lim_{|x_1|\to\infty} m\left|\ln \frac{\left(x_1 - f(\mathcal{D}_1)\right)^{-\alpha-1}}{\left(x_1 - f(\mathcal{D}_2)\right)^{-\alpha-1}}\right| = 0,
\end{aligned} \tag{3.24}$$

and the privacy loss converges to 0. By Lemma 3.3.3, the choice of γ does not impact the asymptotic behavior. Since this result holds for any value of $\mathbf{x} \in \mathcal{R}(\mathcal{M}_f)$, by Theorem 2.3.1, we have proved that the SaS mechanism is ε-differentially private. □

Although Theorem 3.3.4 establishes that the SaS mechanism is purely-Differentially Private, it does not offer a connection between the sensitivity of the query Δf, the scale of the noise distribution γ, and the achieved level of privacy ε. This limitation stems from the absence of a known closed-form expression for the density p_{SaS}. Before pursing further details on these relationships, we revisit the Differential Privacy characteristics of two widely used privacy mechanisms to facilitate the subsequent comparison.

3.4 Privacy Scaling with Noise

In this section, we recall the characteristics of two common privacy mechanisms: the Laplace mechanism and the Gaussian mechanism [17, 31]. After discussing these mechanisms, we proceed to study the relation between the privacy budget ε and the scale γ of the SaS mechanism and to provide related numerical results. We proceed to argue that the privacy budget of the SaS mechanism scales with the same order as the Laplace and Gaussian mechanisms, i.e., we wish to show that

$$\varepsilon_{SaS} \overset{?}{\propto} \frac{\Delta_1 f}{\gamma} \tag{3.25}$$

which is similar to

$$\varepsilon_{Lap} = \frac{\Delta_1 f}{b} \quad \text{and} \quad \varepsilon_{Gau} \propto \frac{\Delta_2 f}{\sigma}. \tag{3.26}$$

3.4.1 Level of privacy afforded by the SaS mechanism

For a given problem, there are three factors to consider when setting the parameters of a mechanism: the sensitivity of the query Δf, the scale of the noise γ, and the privacy budget ε. In this section, we study the relationship between these three quantities for the SaS mechanism.

Theorem 3.3.4 bounds the privacy loss by considering the largest of the m dimensional query responses (see Eq. (3.16)). This motivates us to focus on the case $m = 1$, i.e., we now restrict to real-valued queries, $f(\mathcal{D}) \in [a, b] \subset \mathbb{R}$. Furthermore, in this section, when referring to the sensitivity of query f we exclusively use the ℓ_1-sensitivity and denote it by Δ_1.

We begin by establishing the linear relation between sensitivity and scale. To do so, we first prove that the extremes of the privacy loss, for a given privacy budget ε, occur when the

query over datasets $\mathcal{D}_1 \simeq \mathcal{D}_2$ returns values in the boundary of the range, $\mathcal{R}(f)$. For instance, when $f(\mathcal{D}_1) = b$ and $f(\mathcal{D}_2) = a$ (or vice versa, $f(\mathcal{D}_1) = a$ and $f(\mathcal{D}_2) = b$, by symmetry of the absolute value of the query).

In order to prove that the privacy loss is maximized at the boundary of the query's range, we first establish that the density is monotonic on each semi-infinite interval to the left and right of the location parameter μ. We give a proof for the generic symmetric stable density using the fact that the density is *bell-shaped*, the definition of which is recalled next [97].

Definition 17. *(Bell-Shaped Function) A continuous real-valued function is said to be bell-shaped if the n^{th} derivative, $f^{(n)}$ for each $n \in \mathbb{N}_0$, changes sign exactly n times over its domain.*

◀

Lemma 3.4.1. *(Monotonic First Derivative) The symmetric alpha-stable density (3.7) with location parameter μ is monotonically increasing from $-\infty$ to μ and monotonically decreasing from μ to ∞.*

Proof. See the proof of [97, Cor. 1.3] which states that all stable distributions are bell-shaped densities. Taking $n = 1$ in Def. 17 implies that the first derivative of the density, f', changes sign exactly once. Because the density is symmetric, the change in sign must occur at the axis of symmetry and the density must then decrease monotonically to 0 in the limit as $|x| \to \infty$. □

We now utilize Lemma 3.4.1 to prove that, out of all neighboring datasets $\mathcal{D}_1 \simeq \mathcal{D}_2$, the maximum of the privacy loss occurs at the boundary of the query's range $[a, b]$. Recall that the SaS mechanism involves injecting noise with a location parameter of 0. Thus, the location parameter is the result of the query, $\mu_i = f(\mathcal{D}_i)$, and is itself bounded by the range of the query. By Theorem 3.3.4, the privacy loss of the SaS mechanism is bounded. As a result, we denote

by $x^*(\mu_1, \mu_2)$ the point at which the maximum privacy loss occurs as a function of the location parameters μ_1 and μ_2 generated by datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively.

Theorem 3.4.2. *(Privacy Loss Maximization Occurs at Boundary) Let $\mathcal{D}_1 \simeq \mathcal{D}_2$ be neighboring datasets and denote by f a bounded query that operates on them and returns values in the compact set $[a, b] \subset \mathbb{R}$. Denote the SaS mechanism's privacy loss for an observation x by $\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)$. Let the location parameters of the two densities be $\mu_1 = f(\mathcal{D}_1)$ and $\mu_2 = f(\mathcal{D}_2)$, with $\mu_1 \neq \mu_2$. Then*

$$\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}\Big(x^*(\mu_1, \mu_2)\Big) \leq \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}\Big(x^*(b, a)\Big). \tag{3.27}$$

Proof. Without loss of generality, take $\mu_1 > \mu_2$. Recall that the privacy loss, (2.30), is given by the log-ratio of two densities. Consider Figure 3.3 and let $p(x; \mu_1)$, in blue, and $p(x; \mu_2)$, in orange, represent the numerator and denominator of the privacy loss respectively. Let ϵ be a value in $[0, b - \mu_1]$. First, we show that if the privacy loss achieves a maximum $x^*(\mu_1, \mu_2)$, then

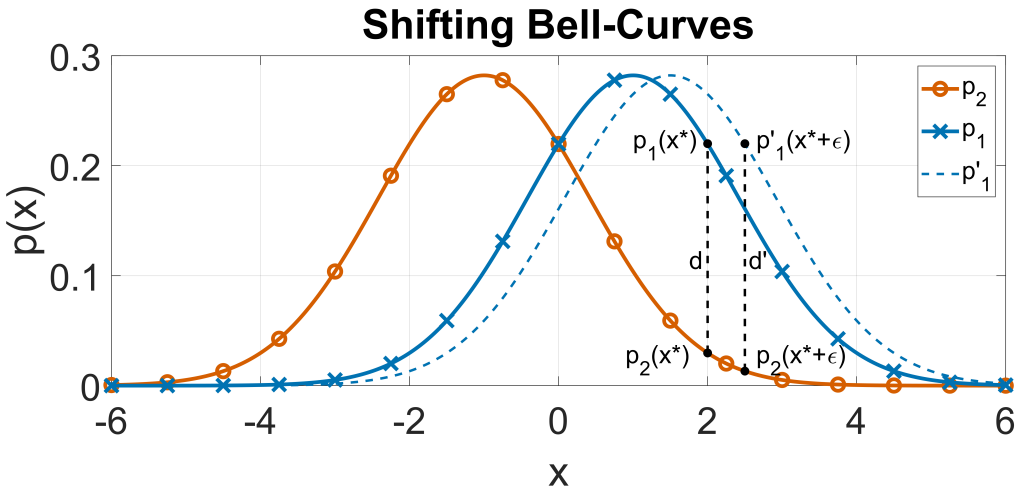

Figure 3.3: Consider two bell curves, shown here as p_1 **in blue** and p_2 **in orange**, with location parameters $\mu_1 > \mu_2$ respectively. Given a point $x^* > \mu_1$, denote by d the distance between the curves at x^*: $d := p_1(x^*) - p_2(x^*)$. Shifting the distribution p_1 to the right by some positive value ϵ, gives the curve p'_1 shown as a dotted line. By Lemma 3.4.1, the distance $d' := p'_1(x^* + \epsilon) - p_2(x^* + \epsilon)$ is necessarily larger than d.

$\mu_1 \leq x^*(\mu_1, \mu_2)$. Observe that, by construction, $p(x = \mu_1; \mu_1) \geq p(x = \mu_1; \mu_2)$. Consider a

point to the left of μ_1. By the symmetry of SaS densities, $p(\mu_1 - \epsilon; \mu_1) = p(\mu_1 + \epsilon; \mu_1)$ and because the first derivative is negative, Lemma 3.4.1, $p(\mu_1 - \epsilon; \mu_2) \geq p(\mu_1 + \epsilon; \mu_2)$. Thus,

$$\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(\mu_1 - \epsilon) \leq \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(\mu_1 + \epsilon). \tag{3.28}$$

Next, let $\mu_1 < b$. Then, observe that $p(x^*(\mu_1, \mu_2); \mu_1) = p(x^*(\mu_1, \mu_2)+\epsilon; \mu_1+\epsilon)$, illustrated by the upper two marked points in Figure 3.3. Similarly, by Lemma 3.4.1, $p(x^*(\mu_1, \mu_2)+\epsilon; \mu_2) \leq p(x^*(\mu_1, \mu_2); \mu_2)$, marked by the two lower points. Thus, $\mathcal{L}$ can only be made larger by increasing μ_1 in the direction of the bound b. Likewise, if $\mu_1 = b$, then shifting the distribution to the left can only decrease the maximum. A similar argument shows that the log-ratio cannot be decreased by shifting $p(x; \mu_2)$ towards $p(x; a)$, which completes the proof. □

Because the maximum of the privacy loss is invariant to translation, Theorem 3.4.2 additionally implies the following corollary.

Corollary 3.4.2.1. *(Relative Location Parameter) Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be any neighboring datasets. Consider the privacy loss of the SaS mechanism, with $\alpha \in (1, 2)$, for a one-dimensional query f, with bounded range $\mathcal{R}(f) = [a, b]$. Denote by Δ_1 the ℓ_1-sensitivity of f. Then, for a given $\alpha \in (1, 2)$ and scale γ,*

$$\max_{\mathcal{D}_1 \simeq \mathcal{D}_2} \max_{x \in \mathbb{R}} \mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x) = \max_{x \in \mathbb{R}} \ln \frac{p_{SaS}(x; \alpha, \gamma, \Delta_1)}{p_{SaS}(x; \alpha, \gamma, 0)}. \tag{3.29}$$

Proof. The result follows directly from Theorem 3.4.2 and that the maximum of the privacy loss is invariant under translation. □

We can now assert that there is a linear relation between the sensitivity of the query Δ_1 and

the scale of the density γ.

Theorem 3.4.3. *(Linearity of Scale and Query's Sensitivity) Let $\mathcal{D}_1 \simeq \mathcal{D}_2$ be neighboring datasets and f be a one-dimensional query with bounded range $\mathcal{R}(f) = [a, b]$. Denote by Δ_1 the ℓ_1-sensitivity of f. Let p_{SaS} be the SaS density as described in equation (3.7). Then, the level of privacy ε remains the same if the sensitivity Δ_1 and the scale γ are both scaled by the same constant $c > 0$:*

$$\max_{x' \in \mathbb{R}} \ln \frac{p_{SaS}(x'; \alpha, c\gamma, c\Delta_1)}{p_{SaS}(x'; \alpha, c\gamma, 0)} = \max_{x \in \mathbb{R}} \ln \frac{p_{SaS}(x; \alpha, \gamma, \Delta_1)}{p_{SaS}(x; \alpha, \gamma, 0)}. \tag{3.30}$$

Proof. We proceed by contradiction. Denote by x^* the location of the optimal value on the right side of (3.30). Consider the left side of (3.30) in terms of the expression (3.7):

$$\max_{x' \in \mathbb{R}} \ln \frac{\int_{-\infty}^{\infty} e^{-|c\gamma t|^\alpha - it(x' - c\Delta_1)} dt}{\int_{-\infty}^{\infty} e^{-|c\gamma t|^\alpha - itx'} dt}. \tag{3.31}$$

The change of variables $\hat{t} = ct$ results in the equivalent expression

$$\max_{x' \in \mathbb{R}} \ln \frac{\int_{-\infty}^{\infty} e^{-|\gamma \hat{t}|^\alpha - i\hat{t}(\frac{x'}{c} - \Delta_1)} d\hat{t}}{\int_{-\infty}^{\infty} e^{-|\gamma \hat{t}|^\alpha - i\hat{t}\frac{x'}{c}} d\hat{t}} \tag{3.32}$$

Denote by x'^* the location of the maximum in (3.32) and assume that it is not equal to cx^*. This

leads to the following contradiction

$$\max_{cx' \in \mathbb{R}} \ln \frac{\int_{-\infty}^{\infty} e^{-|\gamma \hat{t}|^{\alpha} - i\hat{t}(x' - \Delta_1)} d\hat{t}}{\int_{-\infty}^{\infty} e^{-|\gamma \hat{t}|^{\alpha} - i\hat{t}x'} d\hat{t}} \neq \max_{x \in \mathbb{R}} \ln \frac{\int_{-\infty}^{\infty} e^{-|\gamma \hat{t}|^{\alpha} - i\hat{t}(x - \Delta_1)} d\hat{t}}{\int_{-\infty}^{\infty} e^{-|\gamma \hat{t}|^{\alpha} - i\hat{t}x} d\hat{t}} \tag{3.33}$$

which is equivalent to

$$\max_{cx' \in \mathbb{R}} \ln \frac{p_{SaS}(x'; \alpha, \gamma, \Delta_1)}{p_{SaS}(x'; \alpha, \gamma, 0)} \neq \max_{x \in \mathbb{R}} \ln \frac{p_{SaS}(x; \alpha, \gamma, \Delta_1)}{p_{SaS}(x; \alpha, \gamma, 0)}. \tag{3.34}$$

□

Remark 2. *(Normalized Form) Because the scale and sensitivity are related linearly, we can combine γ and Δ_1 into a single parameter $\hat{\gamma} = \gamma/\Delta_1$ by taking $c = 1/\Delta_1$:*

$$\max_{x \in \mathbb{R}} \ln \frac{p_{SaS}(x; \alpha, \gamma, \Delta_1)}{p_{SaS}(x; \alpha, \gamma, 0)} = \max_{x' \in \mathbb{R}} \ln \frac{p_{SaS}(x'; \alpha, \hat{\gamma}, 1)}{p_{SaS}(x'; \alpha, \hat{\gamma}, 0)}. \tag{3.35}$$

We use the normalized form on the right side of Eq. (3.35) to gain an intuitive understanding of how the maximum of the privacy loss behaves as α and γ are allowed to vary. Figure 3.4 fixes $\gamma = \Delta_1 = 1$ and illustrates how the privacy loss approaches a straight line as α tends to 2. Note that when $\alpha = 2$, corresponding to the privacy loss of the Gaussian mechanism, the loss is unbounded, illustrating that the Gaussian mechanism is not purely differentially private. In Figure 3.5, with α fixed at $3/2$, we see that as the scale of the density (noise level), γ, increases, the level of privacy also increases (seen in the decreasing maximum ε value; recall from 2.32 that

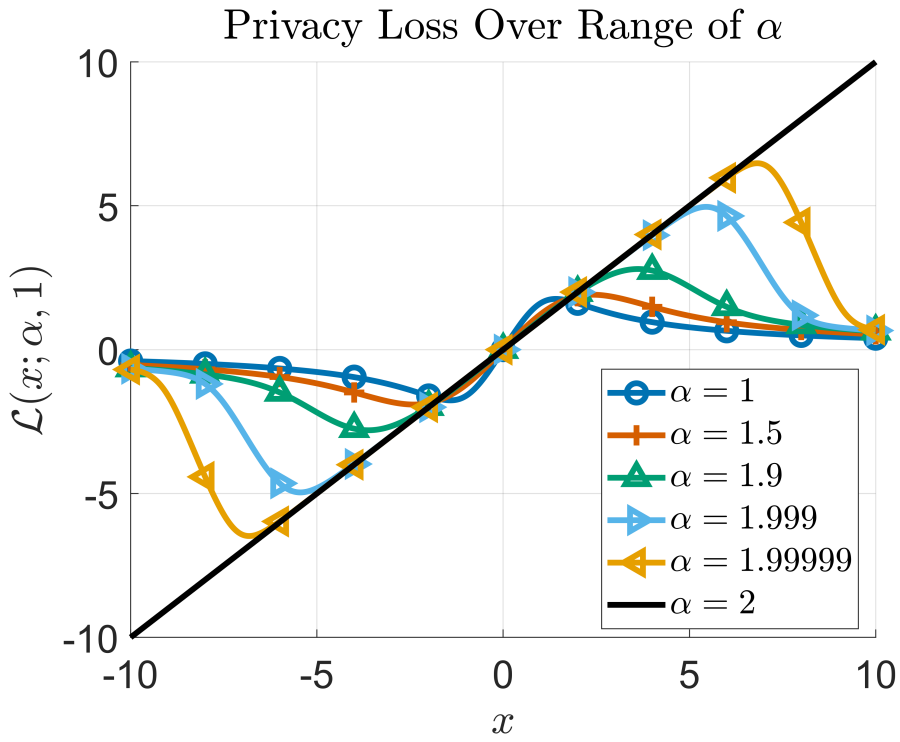

Figure 3.4: Denote by $\mathcal{L}(x;\alpha,\gamma=1)$ the privacy loss of the SaS mechanism with unit scale over observations x. Without loss of generality, let $\mathcal{D}_1$ and $\mathcal{D}_2$ be neighboring datasets such that the privacy loss of the Gaussian mechanism is linear: $\mathcal{L}(x;2,1)=x$ shown in **black**. As the stability parameter α is reduced, we observe that the privacy loss becomes bounded, reaching a peak before converging to the x-axis.

$\varepsilon = \max_x \mathcal{L}(x)$).

Next, to derive the behavior of the privacy loss at observation x in terms of the scale, we use a special case of the second partial sum expansion discussed by Bergström in [95].

Lemma 3.4.4. *(A Second Finite Series Expansion) The symmetric alpha-stable density (3.7), with $\alpha \in (1,2)$ and $\gamma = 1$, has the following finite series expansion:*

$$p_{SaS}(x;\alpha,1,0) = \frac{1}{\pi}\sum_{k=0}^{n}(-1)^k \frac{\Gamma(\frac{k+1}{\alpha})}{k!\alpha}(x)^k \cos\left(\frac{k\pi}{2}\right) + O\big(|x|^{n+1}\big), \tag{3.36}$$

as $|x| \to 0$.

Proof. The full form provided by Bergström [95] states the result for the full range $\beta \in (-1,1)$.

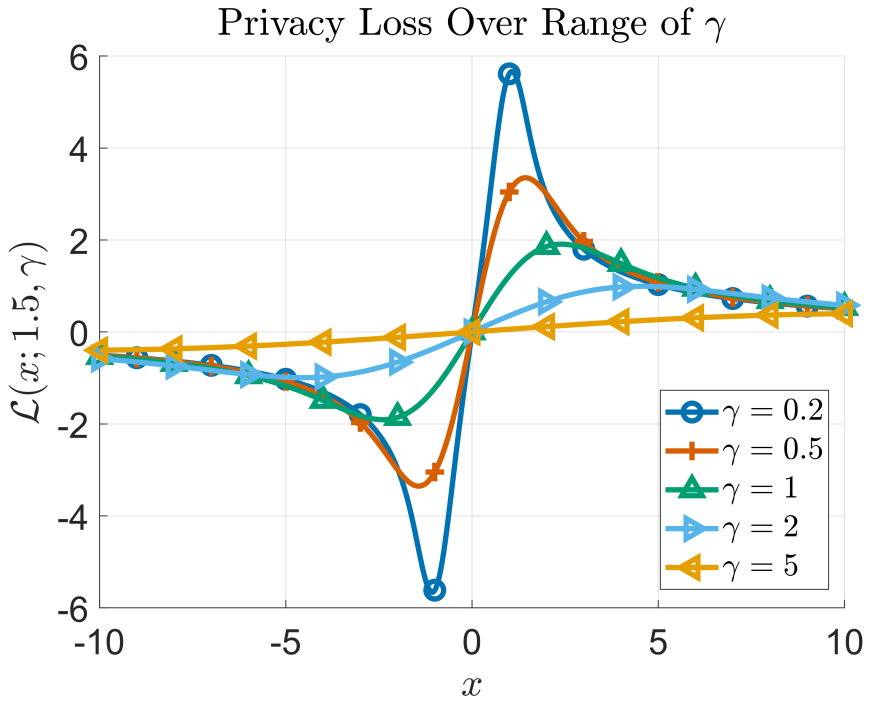

Figure 3.5: Denote by $\mathcal{L}(x; \alpha = 1.5, \gamma)$ the privacy loss of the SaS mechanism with stability parameter $\alpha = 1.5$ over observations x. Without loss of generality, let $\mathcal{D}_1$ and $\mathcal{D}_2$ be neighboring datasets such that the privacy loss is symmetric about the origin. As the scale γ of the density is increased, we observe that the increase in noise decreases the maximum possible privacy loss, compressing the curve toward the x-axis.

In our work, we only require (3.36), so for brevity, we leave out the full form of the series. □

Below we make use of the following two elementary Taylor series expansions. For any $c \neq 0$:

$$\frac{1}{c+x} = \frac{1}{c} - \frac{x}{c^2} + \frac{x^2}{c^3} - \frac{x^3}{c^4} + O(x^4), \tag{3.37}$$

and

$$\ln \frac{c+x}{c} = \frac{x}{c} - \frac{x^2}{2c^2} + \frac{x^3}{3c^3} + \frac{x^4}{4c^4} + O(x^5). \tag{3.38}$$

Using Lemma 3.4.4, we now assert a relationship between the privacy loss for a given observation x and the scale of the SaS mechanism γ.

Theorem 3.4.5. *Let $\mathcal{D}_1 \simeq \mathcal{D}_2$ be neighboring datasets and f a bounded query that operates on them. Denote by Δ_1 the ℓ_1-sensitivity of the query f. Let $\mathcal{M}_f$ be a SaS mechanism with*

$\alpha \in (1,2)$. *Let the observation x be fixed and take γ to be the independent variable. Then*

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)](\gamma) = O\Big(\frac{\Delta_1}{\gamma}\Big) \text{ as } \gamma \to \infty. \tag{3.39}$$

(Δ_1 is included in (3.39) in order to highlight the analogy with (3.26).)

Proof. Fix the observation x, then, by Lemma 3.4.2, the maximum privacy loss for x over the datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ is

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)](\gamma) = \ln \frac{\int_{-\infty}^{\infty} e^{-|\gamma t|^\alpha - it(x-\Delta_1)} dt}{\int_{-\infty}^{\infty} e^{-|\gamma t|^\alpha - itx} dt}. \tag{3.40}$$

Let $\hat{t} = \gamma t$, $\hat{x} = x\Delta_1$, and $\hat{\gamma} = \gamma/\Delta_1$ and denote $(\hat{x}-1)/\hat{\gamma}$ and $\hat{x}/\hat{\gamma}$ by x_1 and x_2. The Eq. (3.40) becomes

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)](\gamma) = \ln \frac{\int_{-\infty}^{\infty} e^{-|\hat{t}|^\alpha - i\hat{t}x_1} d\hat{t}}{\int_{-\infty}^{\infty} e^{-|\hat{t}|^\alpha - i\hat{t}x_2} d\hat{t}}. \tag{3.41}$$

We consider the numerator first, followed by the denominator. Expand the numerator in (3.41) using the partial sum expansion given in Lemma 3.4.4 with $n = 0$:

$$p_{SaS}(x_1; \alpha, 1, 0) = \frac{\Gamma\left(\frac{1}{\alpha}\right)}{\alpha} + O(|x_1|), \quad |x_1| \to 0. \tag{3.42}$$

For simplicity we denote

$$a(\alpha) = \frac{\Gamma\left(\frac{1}{\alpha}\right)}{\alpha}, \tag{3.43}$$

which gives

$$p_{SaS}(x_1; \alpha, 1, 0) = a(\alpha) + O(|x_1|), \quad |x_1| \to 0. \tag{3.44}$$

Thus, there exist positive constants C and x_0 such that

$$|p_{SaS}(x_1;\alpha,1,0)| \le a + C|x_1|, \quad \forall |x_1| \le x_0. \tag{3.45}$$

Replace x_1 by its definition in (3.45), first noting that the translations in x are described by the last parameter in the notation for the SaS density:

$$p_{SaS}\Big(\frac{\hat{x}-1}{\hat{\gamma}};\alpha,1,0\Big) = p_{SaS}\Big(\frac{\hat{x}}{\hat{\gamma}};\alpha,1,\frac{1}{\hat{\gamma}}\Big). \tag{3.46}$$

Then

$$\Big|p_{SaS}\Big(\frac{\hat{x}}{\hat{\gamma}};\alpha,1,\frac{1}{\hat{\gamma}}\Big)\Big| \le a + C\Big|\frac{\hat{x}-1}{\hat{\gamma}}\Big|, \ \ \forall\Big|\frac{\hat{x}-1}{\hat{\gamma}}\Big| \le x_0. \tag{3.47}$$

Note that the range restriction in (3.47) is equivalent to $\hat{\gamma} \ge |\hat{x}-1|/x_0$. Thus, denote $|\hat{x}-1|/x_0$ and $C|\hat{x}-1|$ by γ_0 and $\hat{C}$ respectively. Then

$$\Big|p_{SaS}(\frac{\hat{x}}{\hat{\gamma}};\alpha,1,\frac{1}{\hat{\gamma}})\Big| \le a + \hat{C}\cdot\frac{1}{\hat{\gamma}}, \quad \forall\hat{\gamma} \ge \gamma_0, \tag{3.48}$$

which can be represented in big O notation as

$$p_{SaS}\Big(\frac{\hat{x}}{\hat{\gamma}};\alpha,1,\frac{1}{\hat{\gamma}}\Big) = a + O\Big(\frac{1}{\hat{\gamma}}\Big), \quad \hat{\gamma}\to\infty. \tag{3.49}$$

Using the same logic, the denominator in (3.40) can be represented as

$$p_{SaS}\Big(\frac{\hat{x}}{\hat{\gamma}};\alpha,1,0\Big) = a + O\Big(\frac{1}{\hat{\gamma}}\Big), \quad \hat{\gamma}\to\infty. \tag{3.50}$$

Combining (3.49) and (3.50), (3.41) can now be expressed for large $\hat{\gamma}$ in the form

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)](\gamma) = \ln \frac{a + O(\frac{1}{\hat{\gamma}})}{a + O(\frac{1}{\hat{\gamma}})}, \quad \hat{\gamma} \to \infty. \tag{3.51}$$

Using the elementary Taylor series (3.37), we rewrite the denominator as

$$\frac{1}{a + O(\frac{1}{\hat{\gamma}})} = \frac{1}{a} + O\Big(\frac{1}{\hat{\gamma}}\Big) = \frac{1 + O(\frac{1}{\hat{\gamma}})}{a}, \tag{3.52}$$

as $\hat{\gamma} \to \infty$. Plugging (3.52) into (3.51) gives

$$\begin{aligned}[\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)](\gamma) &= \ln \frac{a + O(\frac{1}{\hat{\gamma}})}{1} \cdot \frac{1 + O(\frac{1}{\hat{\gamma}})}{a} \\ &= \ln \frac{a + O(\frac{1}{\hat{\gamma}}) + O(\frac{1}{\gamma^2})}{a},\end{aligned} \tag{3.53}$$

as $\hat{\gamma} \to \infty$. The squared term is dominated in the limit and leaves

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)](\gamma) = \ln \frac{a + O(\frac{1}{\hat{\gamma}})}{a}, \quad \hat{\gamma} \to \infty. \tag{3.54}$$

Next, we use the elementary Taylor series (3.38) and expand to

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)](\gamma) = \frac{O(\frac{1}{\hat{\gamma}})}{a} + O\Big(\frac{1}{\hat{\gamma}^2}\Big), \quad \hat{\gamma} \to \infty. \tag{3.55}$$

Recalling that $\hat{\gamma} = \gamma/\Delta_1$, we complete the proof:

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1||\mathcal{D}_2}(x)](\gamma) = O\Big(\frac{\Delta_1}{\gamma}\Big), \quad \gamma \to \infty. \tag{3.56}$$

□

Theorem 3.4.5 only guarantees that the privacy of a specific observation x scales as $O(\Delta_1/\gamma)$ for large γ. Without additional information about the location of the maximum, which is difficult to attain due to the lack of a known closed form for the general SaS density, Theorem 3.4.5 does not allow us to conclude that the maximum over all observations scales in the same manor. Because of this, in Figure 3.6 we provide numerical results graphing the max privacy loss ε over a range of scale values γ (with $\Delta_1 = 1$) for a selection of α values. Note that because this is a

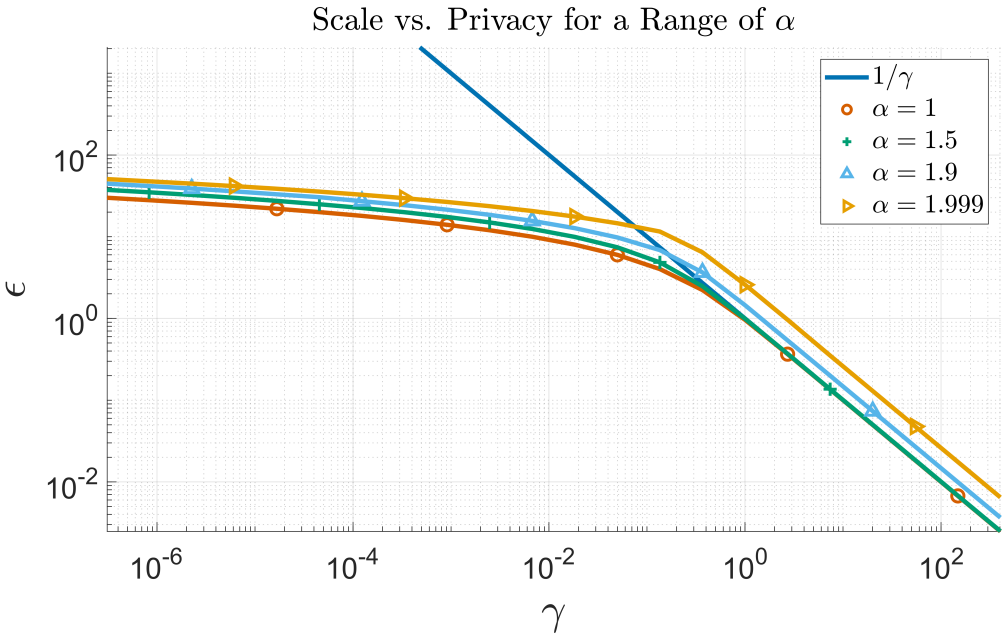

Figure 3.6: The maximum privacy loss of the Laplace mechanism is inversely related to the scale of the injected noise γ, show linearly on the log-log plot in **blue**. For large values of γ, the privacy loss of the SaS mechanism falls off at the same rate, see Corollary 3.4.6.1. However, for small scales, as γ is decreased, the SaS mechanism increases at a rate of $O(\log(1/\gamma))$ as shown in Corollary 3.4.6.2 (as opposed to $O(1/\gamma)$ for the Laplace). The equation for the Cauchy's privacy loss, shown here in **orange**, is explicitly given in Equation (3.57) with $\Delta_1 = 1$.

log-log plot, a vertical shift, as seen with $\alpha = 1.9$, corresponds to a multiplicative scalar in the limiting behavior. We observe that as γ increases, the maximum privacy loss falls off at a rate similar to that for $\alpha = 1$ (at least for practically useful values of ε and γ). To this end, we take advantage of the closed form of the density when $\alpha = 1$ to provide more concrete results for that

case.

Theorem 3.4.6. *Let $\mathcal{D}_1 \simeq \mathcal{D}_2$ be neighboring datasets and f a bounded query with ℓ_1-sensitivity Δ_1 that operates on them. Take the stability parameter $\alpha = 1$ for the SaS mechanism. Then, the privacy budget ε as a function of the scale γ is given by*

$$\varepsilon(\gamma) = \ln \frac{\sqrt{4(\frac{\gamma}{\Delta_1})^2 + 1} + 1}{\sqrt{4(\frac{\gamma}{\Delta_1})^2 + 1} - 1}. \tag{3.57}$$

Proof. Consider the privacy budget ε for the SaS mechanism when $\alpha = 1$:

$$\varepsilon := \max_{x \in \mathbb{R}} \mathcal{L}^{SaS}_{\mathcal{D}_1 || \mathcal{D}_2}(x). \tag{3.58}$$

As in the proof of the foregoing theorem, let $\hat{t} = \gamma t$, $\hat{x} = x\Delta_1$, and $\hat{\gamma} = \gamma/\Delta_1$ in (3.40):

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1 || \mathcal{D}_2}(x)](\gamma) = \ln \frac{\int_{-\infty}^{\infty} e^{-|\hat{t}|^\alpha - i\hat{t}\frac{\hat{x}-1}{\hat{\gamma}}} d\hat{t}}{\int_{-\infty}^{\infty} e^{-|\hat{t}|^\alpha - i\hat{t}\frac{\hat{x}}{\hat{\gamma}}} d\hat{t}}. \tag{3.59}$$

Note that the SaS density, when $\alpha = 1$, takes the closed form

$$p_{SaS}(x; 1, \gamma, \mu) = \frac{1}{\pi\gamma(1 + (\frac{x-\mu}{\gamma})^2)}. \tag{3.60}$$

Substituting (3.60) into (3.59) gives

$$[\mathcal{L}^{SaS}_{\mathcal{D}_1 || \mathcal{D}_2}(x)](\gamma) = \ln \frac{1 + (\frac{\hat{x}}{\hat{\gamma}})^2}{1 + (\frac{\hat{x}-1}{\hat{\gamma}})^2}. \tag{3.61}$$

To find the maximum, we take the derivative of the right side with respect to $\hat{x}$,

$$\frac{d}{d\hat{x}} \ln \frac{1 + (\frac{\hat{x}}{\hat{\gamma}})^2}{1 + (\frac{\hat{x}-1}{\hat{\gamma}})^2} = \frac{-2(\hat{x}^2 - \hat{x} - \hat{\gamma}^2)}{(\hat{\gamma}^2 + (\hat{x} - 1)^2)(\hat{\gamma}^2 + \hat{x}^2)}. \tag{3.62}$$

This equates to 0 when

$$\hat{x}^2 - \hat{x} - \hat{\gamma}^2 = 0. \tag{3.63}$$

There are two solutions:

$$\hat{x}^* = \frac{1}{2}(1 \pm \sqrt{1 + 4\hat{\gamma}^2}). \tag{3.64}$$

Since the privacy loss is symmetric, we take the positive solution without loss of generality. Plugging the positive maximum location into (3.61) gives

$$\varepsilon(\hat{\gamma}) = \ln \frac{1 + \frac{1+\sqrt{1+4\hat{\gamma}^2}}{4\hat{\gamma}^2}}{1 + \frac{1+4\hat{\gamma}^2}{4\hat{\gamma}^2}}. \tag{3.65}$$

Recalling that $\hat{\gamma} = \gamma/\Delta_1$, equation (3.65) is equivalent to the following expression after simplification,

$$\varepsilon\big|_{\alpha=1}(\gamma) = \ln \frac{\sqrt{4(\frac{\gamma}{\Delta_1})^2 + 1} + 1}{\sqrt{4(\frac{\gamma}{\Delta_1})^2 + 1} - 1}. \tag{3.66}$$

□

To make claims about the limiting behavior of the privacy loss, we again invoke three elementary Taylor series:

$$\sqrt{1 + x^2} \pm x = 1 \pm x + \frac{x^2}{2} - \frac{x^4}{8} + O(x^6), \tag{3.67}$$

$$\ln \frac{(1+x)}{(1-x)} = 1 + 2x + \frac{2x^3}{3} + \frac{2x^5}{5} + O(x^7), \tag{3.68}$$

and

$$\sqrt{4x^2+1} + c = (c+1) + 2x^2 - 2x^4 + O(x^5). \tag{3.69}$$

Corollary 3.4.6.1. *(Large scale approximation) In the limit as γ grows without bound, when $\alpha = 1$ the privacy budget ε, falls off at the following rate:*

$$\varepsilon(\gamma)\big|_{\alpha=1} \approx \frac{\Delta_1}{\gamma}, \;\; \textit{as } \gamma \to \infty. \tag{3.70}$$

Proof. The change of variables $x = \Delta_1/(2\gamma)$ applied to Eq. (3.66) gives

$$\varepsilon(x)\big|_{\alpha=1} = \ln \frac{\sqrt{\frac{1}{x^2}+1}+1}{\sqrt{\frac{1}{x^2}+1}-1}. \tag{3.71}$$

Because x only equates to 0 in the limit of $\gamma \to \infty$, and we seek the dynamics when γ is large but finite, we can safely multiply the argument of the logarithm in (3.71) by x/x giving

$$\varepsilon(x)\big|_{\alpha=1} = \ln \frac{\sqrt{1+x^2}+x}{\sqrt{1+x^2}-x}. \tag{3.72}$$

Expand the numerator and denominator of (3.72) using the elementary Taylor series (3.67), giving the following expression for small x after eliminating the higher order terms:

$$\varepsilon(x)\big|_{\alpha=1} \approx \ln \frac{1+x}{1-x}, \quad \text{as } x \to 0. \tag{3.73}$$

This can be further simplified by appealing to the Taylor series expansion (3.68) yielding a first

order approximation

$$\varepsilon(x)\big|_{\alpha=1} \approx 2x, \quad \text{as } x \to 0. \tag{3.74}$$

Recalling that $x = \Delta_1/(2\gamma)$ now gives

$$\varepsilon(\gamma)\big|_{\alpha=1} \approx \frac{\Delta_1}{\gamma}, \quad \text{as } \gamma \to \infty. \tag{3.75}$$

□

Corollary 3.4.6.2. *(Small scale approximation) In the limit as γ becomes vanishing small, for $\alpha = 1$ the privacy budget ε,, increases at the following rate:*

$$\varepsilon(\gamma)\big|_{\alpha=1} \approx 2\ln\frac{\sqrt{2}\Delta_1}{\gamma}, \text{ as } \gamma \to 0. \tag{3.76}$$

Proof. Begin by expanding the argument of the logarithm in (3.66) using the elementary Taylor series (3.68),

$$\ln\frac{\sqrt{4(\frac{\gamma}{\Delta_1})^2+1}+1}{\sqrt{4(\frac{\gamma}{\Delta_1})^2+1}-1} = \ln\frac{2+2(\frac{\gamma}{\Delta_1})^2+O(\gamma^3)}{2(\frac{\gamma}{\Delta_1})^2+O(\gamma^3)}, \quad \gamma \to 0. \tag{3.77}$$

As γ tends to 0, the higher order behavior is dominated by γ^2 and we have

$$\varepsilon(\gamma)\big|_{\alpha=1} \approx \ln\frac{2}{(\frac{\gamma}{\Delta_1})^2}, \quad \gamma \to 0. \tag{3.78}$$

Equivalently, the expression in (3.78) gives

$$\varepsilon(\gamma)\big|_{\alpha=1} \approx 2\ln\frac{\sqrt{2}\Delta_1}{\gamma}, \text{ as } \gamma \to 0. \tag{3.79}$$

□

In Figure 3.7, we supplement the numerical results with graphs of the limiting behavior derived in Corollaries 3.4.6.1 and 3.4.6.2. The figure numerically confirms that, for small γ, the

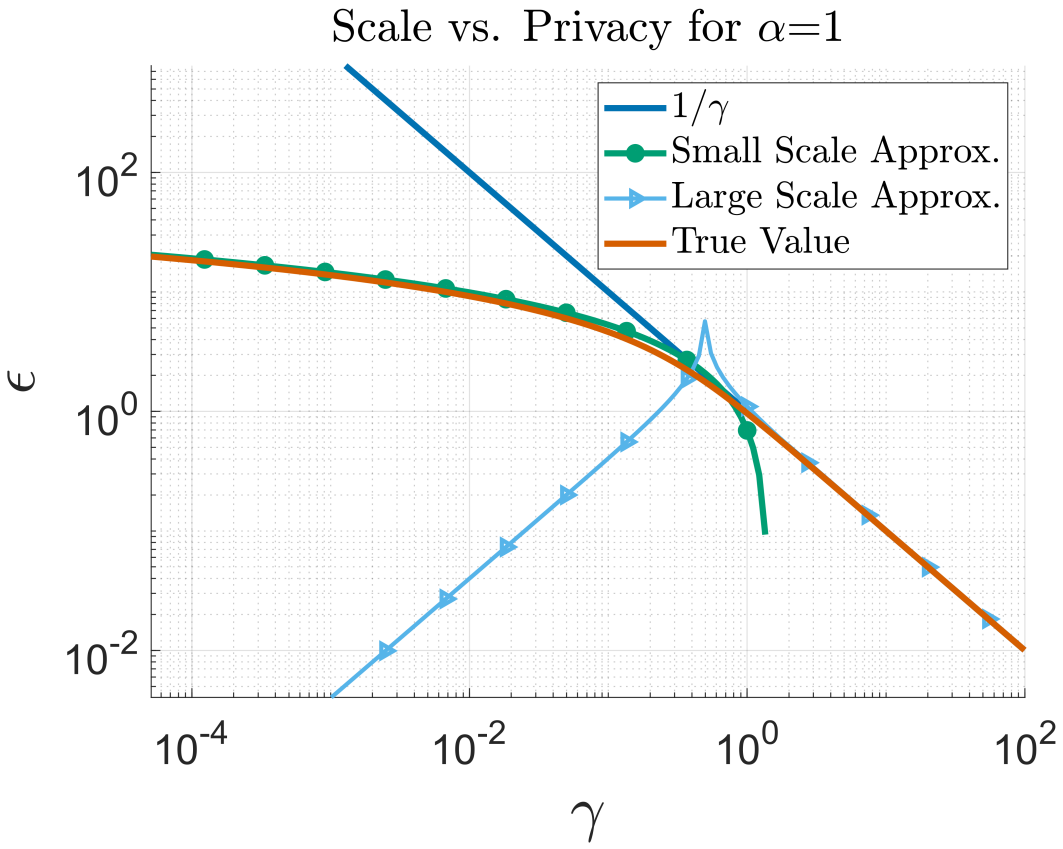

Figure 3.7: The privacy loss ε for the SaS mechanism, with $\alpha = 1$ and $\Delta_1 = 1$, over a range of scale values γ described by (3.57) and shown here in **orange**. For small γ, the privacy loss is approximated by Eq. (3.79) in **green**, and, for large γ, the privacy loss is approaximated by Eq. (3.75), shown in **light blue**. For comparison, the privacy loss of the Laplace mechanism is shown in **blue**.

SaS mechanism, due the appearance of the logarithm in (3.79), scales better than the Laplace and Gaussian mechanisms as recalled in (3.26).

Now that we have shown that the SaS mechanism behaves in a similar manner to other common privacy mechanism, we move on to describe the expected error introduced to the query's result by using any of these mechanisms.

3.5 Error Analysis

It is typical for methods to employ the ℓ_2-norm when defining a measure of error. However, the moments of SaS densities are only defined up to α, and since we consider $\alpha < 2$, the second moment lacks a clear definition [93]. In lieu of the ℓ_2-norm, we opt for the mean absolute deviation (MAD), as used in [31]:

Definition 18. *(Expected Privacy Distortion) Let $\mathcal{D}$ be a dataset and denote by $f(\mathcal{D})$ and $\mathcal{M}_f(\mathcal{D})$ the response of a query and privacy mechanism respectively. Denote the density of the privacy mechanism by Y. The mean absolute deviation is*

$$E\big(f(\mathcal{D}), \mathcal{M}_f(\mathcal{D})\big) := \mathbb{E}|f(\mathcal{D}) - \mathcal{M}_f(\mathcal{D})|, \tag{3.80}$$

which is equivalent to the expectation of the absolute value of the injected noise Y:

$$E\big(f(\mathcal{D}), \mathcal{M}_f(\mathcal{D})\big) = \mathbb{E}|Y|. \tag{3.81}$$

◀

Before beginning the analysis of the error incurred by the SaS mechanism, we establish that the SaS mechanism adheres to strict stability.

Lemma 3.5.1. *(SaS density is Strictly Stable) The SaS density (3.7) with location parameter $\mu = 0$ is strictly stable.*

Proof. Consider three independent and identically distributed SaS densities denoted by Y_1, Y_2,

and Y with $\mu = 0$. Let a and b represent two scalar values. Next, examine the density of the combined random variable $aY_1 + bY_2$. Since SaS densities are determined by their characteristic functions, we establish the following relation:

$$\varphi_{aY_1+bY_2}(t) = \varphi_{aY_1}(t)\varphi_{bY_2}(t). \tag{3.82}$$

Using the definition of a characteristic function, we bring the constants into the argument

$$\begin{aligned} \varphi_{aY_1}(t)\varphi_{bY_2}(t) &= \mathbb{E}[e^{itaY_1}]\mathbb{E}[e^{itbY_2}] \\ &= \varphi_{Y_1}(at)\varphi_{Y_2}(bt). \end{aligned} \tag{3.83}$$

Expand by substituting the expression for the characteristic function of a stable distribution with $\mu = 0$ (3.4) into both functions on the right side,

$$\begin{aligned} \varphi_{Y_1}(at)\varphi_{Y_2}(bt) &= \exp(|\gamma at|^\alpha)\exp(|\gamma bt|^\alpha) \\ &= \exp|(a^\alpha + b^\alpha)^{1/\alpha}\gamma t|^\alpha. \end{aligned} \tag{3.84}$$

Setting $c = (a^\alpha + b^\alpha)^{1/\alpha}$ gives $aY_1 + bY_2 = cY$. □

We are now equipped to determine the expected error introduced in the query by the SaS mechanism.

Theorem 3.5.2. *(Expected Distortion Due to SaS mechanism) Let f be a bounded query that operates on dataset $\mathcal{D}$. Denote by $\mathcal{M}_f$ the SaS mechanism and take the stability parameter α to*

be restricted to the range $\alpha \in (1, 2)$. *Then, the mean absolute distortion is*

$$E\big(f(\mathcal{D}, \mathcal{M}_f(\mathcal{D})\big) = \frac{2\gamma}{\pi}\Gamma\big(1 - \frac{1}{\alpha}\big). \tag{3.85}$$

Proof. Note that, by Lemma 3.5.1, the noise injected by the SaS mechanism is strictly stable. In [93], the proof of Corollary 3.5 includes a statement that if a density Y is strictly stable, then its mean absolute deviation is given by

$$\mathbb{E}[|Y|] = \frac{2\gamma}{\pi}\Gamma\big(1 - \frac{1}{\alpha}\big). \tag{3.86}$$

□

We now provide the expected distortions of the two most common privacy mechanisms: the Laplace and the Gaussian mechanisms [30, 31], to show that each induces an error linear in the scale of the noise. The mean absolute deviation of the Laplace density is

$$\mathbb{E}[|Lap(0, b)|] = \mathbb{E}[Exp(b^{-1})] = b. \tag{3.87}$$

The mean absolute deviation Gaussian density is the expected value of the half-normal random variable

$$\mathbb{E}[|\mathcal{N}(0, \sigma^2)|] = \sqrt{\frac{2}{\pi}}\sigma. \tag{3.88}$$

Note that for each of the three densities, the error is related linearly to the density's respective scale. From (3.86) we recover the distortion of the Gaussian mechanism by taking $\alpha = 2$ and $\gamma = \sigma/\sqrt{2}$. Next, we proceed to prove that the expected distortion is monotonic in α, reaching a

minimum when $\alpha = 2$ and diverging as α tends to 1.

Remark 3. *We note that the relationship between the expected error of the SaS mechanism* $\mathbb{E}[|Y^{SaS}|]$ *and the scale of the density* γ *in (3.86) is linear. Recall that the privacy budget* ε *is inversely proportional to the large values of* γ *(Figure 3.6). This relationship is proven for* $\alpha = 1$ *in Corollary 3.4.6.1. Therefore, small values of* ε*, while enhancing the client's privacy, necessarily increase the expected error induced by the mechanism. In other words, the level of privacy is inversely related to the accuracy of the query. We note that this relationship is shared by other common mechanisms such as the Laplace and Gaussian.*

Corollary 3.5.2.1. *(Error is Monotonic in* α*) The mean absolute distortion injected into a query by the SaS mechanism decreases monotonically as* α *increases from* 1 *to* 2*.*

Proof. Because the stability parameter α is chosen from the bounded set $(1, 2)$, the argument of the Gamma function in (3.86) varies between $(0, 1/2)$. The Gamma function has an asymptote at $x = 0$ and reaches a local minimum in the right plane at $x \approx 1.462$ [96]. Thus, for a given γ, the distortion in Eq. (3.86) is minimized when α tends to 2. □

A naive first thought is that $\alpha = 2$ is the optimal parameter value as the injected error achieves a minimum. This result is proven in Corollary 3.5.2.1 and depicted in Figure 3.8. Table 3.1 provides a list of expected distortions for a selection of stability values α. While choosing α to be 2 minimizes the constant term in the expected distortion, it increases the required scale γ necessary to achieve a given privacy budget ε. By selecting α close to 2 we can achieve an essentially equivalent expected distortion constant but provide better levels of privacy. In particular, note that the expected distortion between $\alpha = 2$ and $\alpha = 1.999$ differ only by only 0.044%. Thus, to achieve similar accuracy results to the Gaussian mechanism, we can choose to

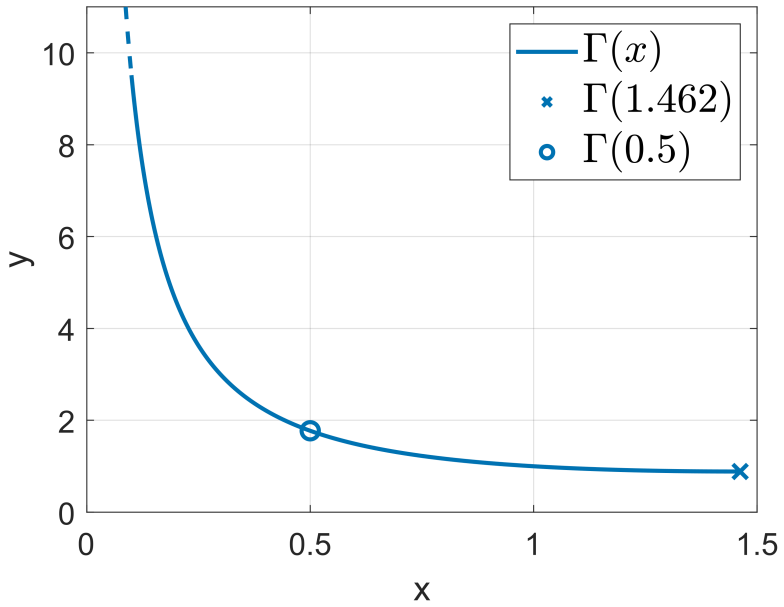

Figure 3.8: The Gamma function $\Gamma(x)$ achieves a minimum value in the right hand plane at $x \approx 1.462$. When α is bounded between $[1, 2)$, the Gamma component of the injected error takes input values in $[0, 0.5)$. The Gamma function is monotonically decreasing in this interval from ∞ to 1.7725.

Table 3.1: The expected distortion for the Gaussian mechanism ($\alpha = 2$) and a selection of SaS mechanisms ($\alpha < 2$). The expected distortion is given as a multiple of the injected noise scale γ.

α	Expected Distortion
2	1.1284γ
1.999	1.1289γ
1.99	1.1340γ
1.95	1.1576γ
1.9	1.1903γ
1.8	1.2687γ
1.0	∞

focus on α close to 2, leaving a further exploration of an optimal choice of α for future work. It should be noted that decreasing α enhances the privacy experienced by the client, thus decreasing ε. Therefore, for a fixed privacy budget ε one can compose an optimization that minimizes a weighted sum of privacy and injected noise. This optimization is problem specific since it deals with the particular sensitivity of the dataset in question, as well as the desired weighting between error and privacy. We leave it for a more application focused examination.

3.6 Concluding Remarks

In conclusion, the SaS mechanism represents an advancement in the field of Differential Privacy. This mechanism not only provides strong guarantees of privacy but also offers distinct advantages when compared to other common privacy mechanisms. We have proven that the SaS mechanism achieves pure Differential Privacy, ensuring that individual data points remain protected even in the face of powerful adversaries. This is starkly contrasted with the Gaussian mechanism, which only achieves approximate Differential Privacy. Additionally, the SaS mechanism utilizes a stable density, allowing it to be used in local applications where the Laplace mechanisms is difficult to analyze.

We showed that the expected distortion introduced by the SaS mechanism into the query result can be made to be essentially equivalent to the Gaussian mechanism. The expected distortion can additionally be formulated as an optimization between injected error and privacy guarantees. Thus, there is little reason to use the Gaussian mechanism over the SaS mechanism. In the next section, we examine the SaS mechanism under modern adaptations of Differential Privacy which are designed specifically to estimate the privacy lost during the training of deep neural networks.

Chapter 4

Analysis of SaS Mechanism Performance

In the era of big data, preserving privacy while extracting valuable insights is paramount. Differential Privacy (DP) has emerged as a promising framework for achieving this delicate balance, offering rigorous privacy guarantees in data analysis. In this chapter, we employ two mechanisms within this framework: the Symmetric alpha-Stable (SaS) mechanism and the Gaussian mechanism. Privacy mechanisms play a crucial role in safeguarding individual privacy while enabling meaningful analysis of sensitive datasets. This chapter delves into an in-depth analysis of the privacy curves associated with both the SaS and Gaussian mechanisms, aiming to provide insights into their respective privacy properties and trade-offs.

Furthermore, this chapter presents the results of training Deep Neural Networks (DNNs) on three widely studied classification datasets: MNIST [98], CIFAR-10 [99], and a dataset related to Pneumonia classification [100]. By employing both the SaS and Gaussian mechanisms, we aim to compare their effectiveness in preserving privacy across different types of datasets and tasks. Through comprehensive experimentation and analysis, we seek to uncover the impact of various factors, such as dataset complexity and privacy parameter settings, on the performance and privacy guarantees offered by these mechanisms.

By investigating the privacy curves of the SaS and Gaussian mechanisms and their implications for training DNNs on diverse datasets, this chapter contributes to a deeper understanding of Differential Privacy in real-world applications. Ultimately, we aim to assist practitioners and researchers in making informed decisions regarding privacy-preserving data analysis techniques, paving the way for the development of more robust and privacy-conscious Machine Learning models.

4.1 Analysis of Privacy Curve

To initially compare the level of privacy achieved by the SaS mechanism with that of Gaussian mechanisms, we analyze their respective privacy curves. In Figure 4.1, we depict the numerical evaluation of the privacy curves for several values of α, recalling that $\alpha = 2$ is equivalent to the Gaussian mechanisms. The privacy curve for the Gaussian mechanism, without subsampling or compositions, is characterized by the following well-known expression:

$$\delta(\varepsilon) = \Phi(-\frac{\varepsilon\sigma}{\mu} + \frac{\mu}{2\sigma}) - e^{\varepsilon}\Phi(-\frac{\varepsilon\sigma}{\mu} - \frac{\mu}{2\sigma}), \tag{4.1}$$

where Φ is the Gaussian Cumulative Distribution Function (CDF) [101]. Notably, this equation diverges, thereby necessitating that the Gaussian mechanism can only fulfill equation (2.9) with $\delta > 0$. Comparatively, the privacy curves for the SaS mechanisms (depicted in orange and green) exhibit plateaus below the curve for the Gaussian mechanism (blue). These plateaus are horizontal asymptotes, stemming from the strict bound on the Privacy Random Variables [94]. Such bounds enable the SaS mechanism to satisfy equation (2.5) for specific ε values with $\delta = 0$. In this context, we also consider the privacy achieved by the SaS mechanism when $\delta > 0$ to

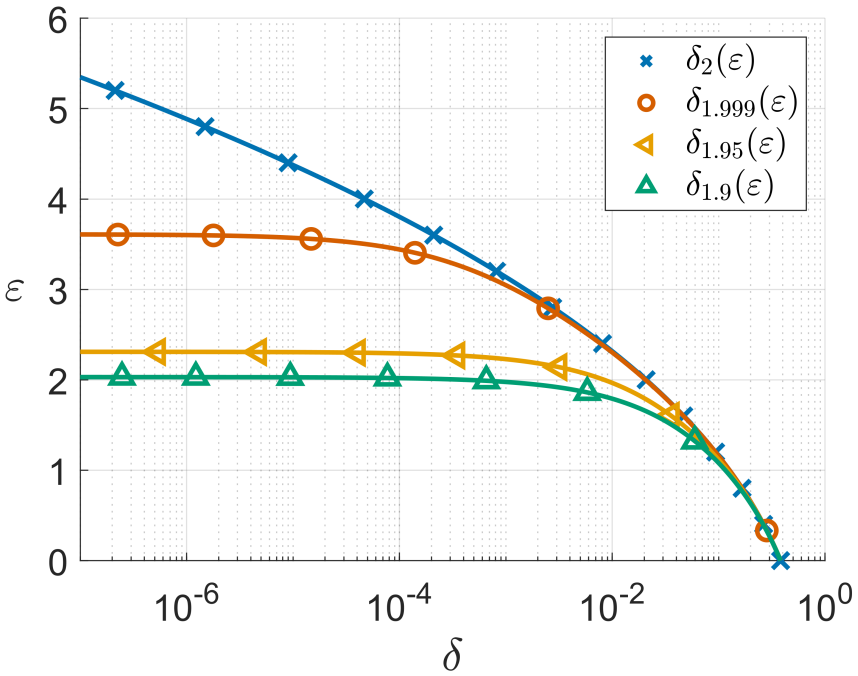

Figure 4.1: The privacy curve for four privacy mechanisms with scale $\gamma = 1$: the SaS mechanism with $\alpha = 1.95$ (**yellow** $\triangleleft$), the SaS mechanism with $\alpha = 1.9$ (**green** $\triangle$), the SaS mechanism with $\alpha = 1.999$ (**orange** $\circ$), and the Gaussian mechanism ($\alpha = 2$, **blue** $\times$). Because the SaS mechanism with α strictly less than 2 is purely differentially rivate, the privacy curve for both $\alpha = 1.9$ and $\alpha = 1.999$ exhibit a horizontal asymptote. While decreasing α improves privacy, it is shown in Table 3.1 that this in turn increases the expected distortion of the query response.

facilitate a more effective comparison with the Gaussian mechanism. The observed divergence in Figure 4.1 underscores a significant privacy enhancement.

When allocated a fixed privacy budget, a single application of a small ε privacy mechanism offers the advantage of accommodating more compositions. This directly correlates with the number of iterations during the Machine Learning training process, thereby enhancing the accuracy of trained models. Typically, an increase in privacy corresponds to a rise in the error introduced into the query through the mechanism. However, for the SaS mechanism with α approximately equal to 2, the expected error introduced is nearly equivalent with that caused by the Gaussian mechanism, see Section 3.5.

4.2 Simulation Setup

In this section, we outline the structure of the simulation that is shared by all trials. This includes the type of learning algorithm and Python framework.

4.2.1 DP-SGD

To train the Machine Learning models, we use Differentially Private Stochastic Gradient Descent (DP-SGD), a privacy-preserving optimization algorithm designed for training Deep Neural Networks while ensuring that client training data remains private [43]. In traditional stochastic gradient descent, updates to model parameters are based on the gradients of the loss function computed from a randomly selected subset of the training data. DP-SGD extends this approach by incorporating Differential Privacy, preventing the inadvertent leakage of sensitive information from individual data samples. This is achieved by clipping the gradient step to ensure a finite ℓ_p norm and by injecting noise drawn from a pre-selected density to the update step during the optimization process. The level of noise is determined by privacy parameters, such as the privacy budget ε and the accepted error parameter δ.

4.2.2 DP programming environment

In this work we use the privacy-preserving deep learning library Opacus [102]. Opacus is a Python package designed to facilitate privacy-preserving Machine Learning workflows by providing a user-friendly interface for implementing Differential Privacy (DP) mechanisms. Developed by researchers at Facebook AI, Opacus serves as a valuable tool for practitioners and researchers aiming to incorporate robust privacy guarantees into their Machine Learning models.

Opacus offers a comprehensive suite of DP mechanisms, including the Laplace mechanism, the Gaussian mechanism, and comes equipped with the Differentially Private Stochastic Gradient Descent (DP-SGD) algorithm.

One of the key features of Opacus is its seamless integration with popular deep learning frameworks such as PyTorch [103]. By leveraging the capabilities of these frameworks, Opacus simplifies the process of implementing DP mechanisms within existing Machine Learning pipelines. This integration allows users to apply DP techniques to a wide range of Machine Learning tasks, including classification, regression, and generative modeling.

Opacus also provides tools for evaluating and analyzing the privacy guarantees offered by DP mechanisms. Users can generate privacy reports, visualize privacy curves, and assess the impact of different privacy parameters on model performance. These capabilities empower users to make informed decisions about the trade-offs between privacy and utility in their Machine Learning workflows. Overall, Opacus represents a valuable resource for anyone interested in incorporating Differential Privacy into their Machine Learning projects.

4.3 MNIST

The MNIST dataset is a collection of 28×28 pixel grayscale images of handwritten single digit numbers [98]. The dataset serves as a popular benchmark in the field of Machine Learning, particularly in the development and evaluation of novel Machine Learning methods. The dataset is comprised of $60,000$ training images and $10,000$ testing images, making it a standard resource for training and assessing the performance of various Machine Learning models. The goal is to accurately classify each digit based on the pixel values in the images.

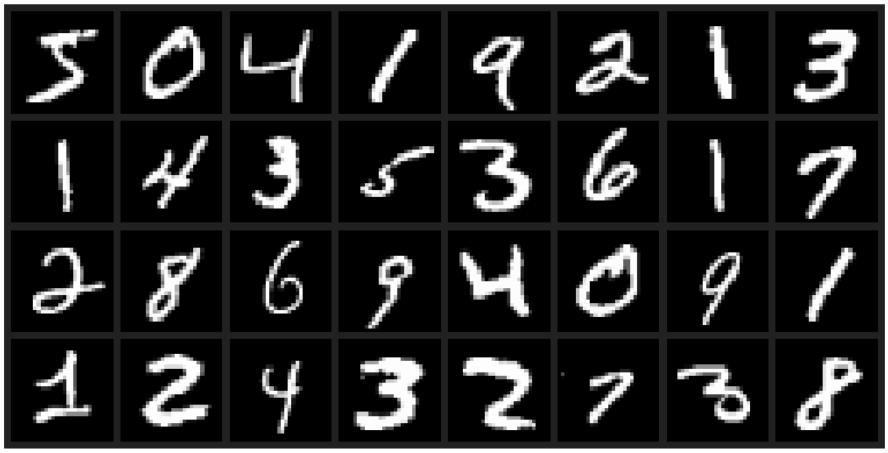

Figure 4.2: A selection of input images from the MNIST dataset. The images are publicly available and are used as a common bench-marking dataset for classification methods. Each input consists of a hand-drawn image of a single digit, 0 through 9. The objective of a learning algorithm is thus to predict the digit in the image.

Researchers often use the MNIST dataset as a starting point to test the effectiveness of different techniques. The dataset's simplicity and small size make it convenient for experimenting with various algorithms, allowing developers to quickly iterate and refine their models.

The dataset has seen a revival within the field of Differential Privacy, where it is used to ensure that privacy techniques do not inhibit the accuracy of the trained model. To prevent the leaking of sensitive data while using advanced composition techniques, it has become a standard to set δ relative to the magnitude of the training dataset, $\delta \approx 1/n$, where n is the size of the training set. For MNIST, we take $\delta = 1 \times 10^{-5}$.

4.3.1 Model for MNIST

On the MNIST dataset, we use a small 4-layer convolutional neural network (CNN) architecture for the image classification tasks. Our model comprises two convolutional layers followed

by two fully connected layers. The first convolutional layer takes input images with a single channel and produces 16 output channels using an 8×8 kernel with a stride of 2 and padding of 3. The second convolutional layer then processes the feature maps from the first layer, generating 32 output channels with a 4×4 kernel and a stride of 2. Following each convolutional layer, Rectified Linear Unit (ReLU) activation functions are applied to introduce non-linearity. Max-pooling operations with a kernel size of 2×2 and a stride of 1 are performed after each ReLU activation to downsample the feature maps. Subsequently, the feature maps are flattened and fed into two fully connected layers. The first fully connected layer reduces the dimensionality to 32 units with ReLU activation, and the final layer produces classification scores with 10 output neurons. The network outputs class predictions based on the input images, enabling classification into different categories. We use an initial learning rate of 0.1 with built-in cosine decay and set gradient clipping to 1. The model is trained for 10 epochs with a training batch size of 64. The subsampling ratio is set to 0.001. For all the trials on the MNIST data, the privacy parameter δ is held fixed at a $\delta = 1 \times 10^{-5}$.

4.3.2 Results on MNIST dataset

We summarize the accuracy and privacy results of the model training on the MNIST dataset in Table 4.1. To obtain each result, we train the model over 10 epochs. In the table, the value for test accuracy gives the average and one standard deviation of 10 trials.

In each trial, the scale of the noise is empirical adjusted so that a fixed privacy budget is reached after the 10 epochs. The last column gives the expected distortion given the scale γ

Table 4.1: In each trial, the scale of injected noise γ is tuned to reach a target privacy budget ε after the given number of epochs. The provided accuracy on the MNIST dataset is the average result of 10 trials with one standard deviation. The model is trained over 10 epochs with a privacy budget of $0.5, 1$, and 3. The top row in the table indicates the accuracy of the model with Differential Privacy disabled.

Data	α	ε	γ	Accuracy	$\mathbb{E}(\lvert Y \rvert)$
MNIST	—	∞	—	$98.63 \pm 0.12\%$	0
	2	0.5	1.41	$91.89 \pm 0.46\%$	1.5910
	2	1	1.06	$94.05 \pm 0.36\%$	1.1961
	2	3	0.81	$95.22 \pm 0.35\%$	0.9140
	1.999	0.5	0.93	$\mathbf{94.60 \pm 0.30\%}$	1.0499
	1.999	1	0.81	$\mathbf{94.89 \pm 0.27\%}$	0.9144
	1.999	3	0.67	$\mathbf{95.53 \pm 0.30\%}$	0.7564

Table 4.2: The percentage increase in accuracy achieved by switching from the Gaussian mechanism ($\alpha = 2$) to the SaS mechanisms with $\alpha = 1.999$. These tests indicate that a greater advantage is obtained in the high-privacy regime.

ε	Acc. Improvement
0.5	2.95%
1	0.89%
3	0.33%

and stability α. As seen in Table 4.1, the SaS mechanism is able to achieve a greater accuracy for each given ε than the Gaussian mechanism. This higher accuracy is a result of the increase in privacy awarded by the heavier tail, which allows for the use of less noise. In fact, the SaS mechanism provides the greatest improvement in the high privacy regime. Table 4.2 provides the percentage improvement in accuracy by switching the model from the Gaussian mechanism to the SaS mechanisms with $\alpha = 1.999$.

4.4 CIFAR-10

The CIFAR-10 dataset is another common image classification problem in the field of Machine Learning [99]. The ten classes in the CIFAR-10 dataset include the following common objects: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Each

class contains $6,000$ images, split into $50,000$ training images and $10,000$ testing images. Because the training dataset is on the same order of magnitude as the MNIST's training set, we again take $\delta = 1 \times 10^{-5}$.

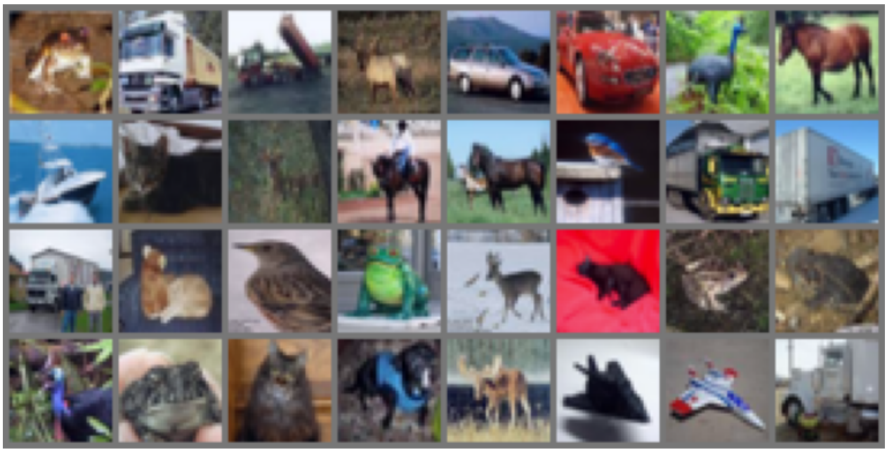

Figure 4.3: A selection of input images from the CIFAR-10 dataset. The images are publicly available and are used as a common benchmark dataset for classification methods. Each image contains one of 10 classes.

The inclusion of color makes CIFAR-10 a more challenging dataset than MNIST. While existing Differential Privacy methods can achieve good accuracy and privacy results on the MNIST dataset, the CIFAR-10 dataset still proves to be a challenging dataset for Differentially Private Machine Learning techniques [104]

4.4.1 Model for CIFAR-10

For the CIFAR-10 data, we used a convolutional neural network with 9 layers. The model is comprised of four convolutional layers with Rectified Linear Unit (ReLU) activation functions, gradually increasing the number of output channels from 32 to 128. Each convolutional layer uses a 3×3 kernel, stride of 1, and padding of 1 to maintain spatial dimensions. Between

convolutional layers, average pooling layers with a 2×2 kernel and stride of 2 are applied to downsample feature maps. The network employs adaptive average pooling to ensure a fixed-sized output before passing through a fully connected linear layer which produces the final output classification. While the initial learning rate and subsampling ratio remained fixed at 0.1 and 0.001 respectively, we trained the CIFAR-10 model for 200 epochs with a batch size of 2048. For all trials on the CIRAF-10 dataset, except where noted, the privacy parameter δ was again held at a $\delta = 1 \times 10^{-5}$.

4.4.2 Results on CIFAR-10 dataset

We summarize the accuracy and privacy results of the model training on the CIFAR-10 dataset in Table 4.3. Each result is the average and standard deviation of 10 trials. In each trial, the model is trained over 200 epochs.

Table 4.3: We present the test accuracy and privacy results of our model after training the CIFAR-10 dataset. In each trial, the scale of injected noise γ is tuned to reach a target privacy budget ε after 200 epochs. The provided accuracy is the average and standard deviation of 10 trials. The results compare the Gaussian ($\alpha = 2$) with the SaS ($\alpha = 1.999$) mechanisms for privacy budgets $1, 3$,and 8. The results marked with a $*$ are state-of-the-art for the privacy budget $\varepsilon = 1$ on this the CIFAR-10 dataset.

Data	α	ε	γ	Accuracy	$\mathbb{E}(\lvert Y \rvert)$
CIFAR-10	—	∞	—	$81.28 \pm 0.50\%$	0
	2	1	15.41	$50.50 \pm 0.48\%$	17.3883
	2	3	5.66	$60.63 \pm 0.63\%$	6.3866
	2	8	2.60	$68.55 \pm 0.48\%$	2.9338
	1.999	1	5.66	$\mathbf{60.52}^{*} \pm \mathbf{0.41}\%$	6.3898
	1.999	3	3.20	$\mathbf{67.53 \pm 0.53}\%$	3.6126
	1.999	8	2.01	$\mathbf{70.79 \pm 0.38}\%$	2.2692

As with the MNIST trials, the scale of the noise is empirical adjusted to reach a fixed privacy budget. The last column gives the expected distortion given the scale γ and stability α. The results on the CIFAR-10 dataset mirror the take-away from the MNIST data where the

Table 4.4: The percentage increase in accuracy achieved by switching from the Gaussian mechanism ($\alpha = 2$) to the SaS mechanisms with $\alpha = 1.999$. These tests indicate that a greater advantage is obtained in the high-privacy regime.

ε	Acc. Improvement
1	19.84%
3	11.38%
8	3.27%

greatest improvement are seen in the high privacy regime. According to a recent survey [104], the results for $\varepsilon = 1$ are state-of-the-art (SOTA). These results are more notable given that the model we trained is much less capable and achieves a significantly reduced accuracy in the DP-free case when compared with previous SOTA results. We used a smaller model to allow fair comparison between the two mechanisms by allowing more statistics to be gathered by the runs afforded by the reduced training time. This signifies that the accuracy gained is from the reduced noise allowed by the SaS mechanism. Table 4.4 provides the percentage improvement in accuracy by switching the model from the Gaussian mechanism to the SaS mechanisms with $\alpha = 1.999$.

4.5 Pediatric Pneumonia Dataset

To test the privacy advantage on a more realistic dataset, we tested our framework using a classification task involving chest radiographs from the Pediatric Pneumonia Chest X-Ray dataset as described originally in [105]. The original task is designed for three-class classification: normal, bacterial pneumonia, or viral pneumonia. However, as in [48], we simplify the problem into a binary classification task by merging the viral and bacterial pneumonia labels. This binary task focuses on predicting whether a radiograph indicates signs of pneumonia or not, identifying individuals who should receive additional examination.

Figure 4.4 depicts a subset of the Pediatric Pneumonia Chest X-Rays training dataset [105].

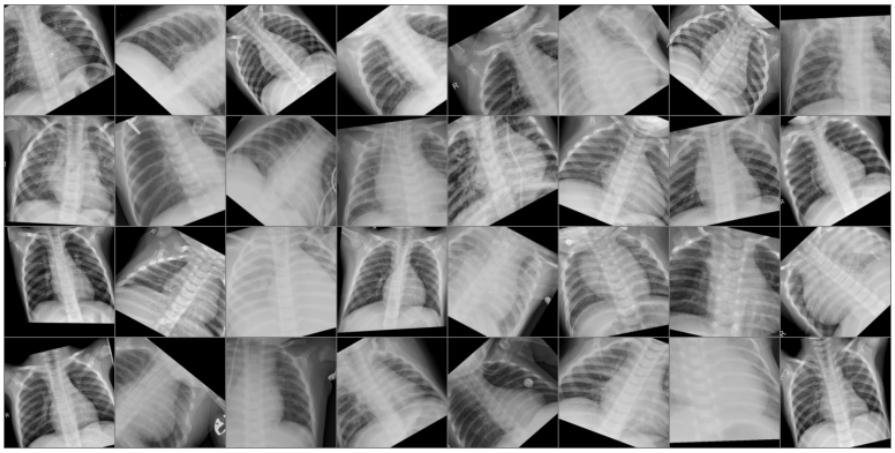

Figure 4.4: The Pediatric Pneumonia Chest X-Ray dataset is composed of x-ray images of patient lungs. The objective of a Machine Learning model trained on this dataset is to classify whether the patient has one of two types of pneumonia or if their lungs are healthy.

The objective of the trained model is to determine the presence of pneumonia in the patient. The dataset is comprised of 5163 chest x-rays, from which 1339 are from healthy patients and 3824 show evidence of pneumonia. We divide the dataset into a training set of 4539 images and a test set of 624 images. We train each model for 20 epochs and adjust the noise scale γ to attain a predetermined list of privacy budgets, $\varepsilon \in \{0.6, 1, 3\}$. The learning rate is determined empirically. Each trial is run 10 times with the average results summarized below.

4.5.1 Model for pediatric pneumonia dataset

For the classification task, we employed a publicly available model that is commonly used in both the private and non-private settings, the VGG-11 architecture with Batch Normalization disabled [106]. The VGG-11 architecture is a Convolutional Neural Network (CNN) model that belongs to the VGG (Visual Geometry Group) family of models. It was introduced by the Visual Geometry Group at the University of Oxford. VGG-11 consists of 11 layers, including convolu-

tional layers with small 3x3 filters, followed by max-pooling layers to downsample the feature maps. Batch Normalization is disabled to ensured that the training procedure satisfied the privacy constraints. The architecture ends with fully connected layers for classification. VGG-11 is known for its simplicity and effectiveness, providing a strong baseline for various computer vision tasks, including image classification and object detection.

4.5.2 Results on pediatric pneumonia dataset

For the Pediatric Pneumonia Chest X-Ray dataset, the model is tasked with the binary classification task of determining if the patient in the input X-Ray has pneumonia. To this end, the model is trained for 20 epochs. As the dataset is smaller than that of MNIST and CIFAR-10, the results exhibit a greater level of variance than both prior datasets and so we increased the number of trials to 40. Table 4.5 summarizes the results of 40 trials per set of parameters. The test accuracy is provided as the average plus-minus one standard deviation.

Table 4.5: The test accuracy and privacy results for the Machine Learning model trained on the Chest X-Ray dataset. In each trial, the scale of injected noise γ is tuned to reach a target privacy budget ε after 20 epochs. The provided accuracy is the average and standard deviation of 40 trials. The results compare the Gaussian ($\alpha = 2$) with the SaS ($\alpha = 1.999$) mechanisms for privacy budgets $0.6, 1$,and 3.

Data	α	ε	γ	Accuracy	$\mathbb{E}(\lvert Y \rvert)$
Chest X-Ray	—	∞	—	$92.2 \pm 0.25\%$	0
	2	0.6	2.441	$85.35 \pm 1.75\%$	2.7544
	2	1	1.365	$89.75 \pm 0.96\%$	1.5403
	2	3	0.869	$90.26 \pm 0.75\%$	0.9806
	1.999	0.6	1.115	$\mathbf{89.74 \pm 0.82\%}$	1.2587
	1.999	1	0.954	$\mathbf{90.08 \pm 0.65\%}$	1.0770
	1.999	3	0.724	$\mathbf{90.41 \pm 0.55\%}$	0.8173

As with the prior two datasets, the SaS mechanisms is able to outperform the Gaussian mechanisms at all privacy budgets tested. Once again, the greatest difference in performance is observed in the high privacy regime. Table 4.6 provides the percentage improvement in accuracy

Table 4.6: The percentage increase in accuracy achieved by switching from the Gaussian mechanism ($\alpha = 2$) to the SaS mechanisms with $\alpha = 1.999$. These tests indicate that a greater advantage is obtained in the high-privacy regime.

ε	Acc. Improvement
0.6	5.14%
1	0.37%
3	0.17%

by switching the model from the Gaussian mechanism to the SaS mechanisms with $\alpha = 1.999$.

4.6 Discussion

To explore how the SaS mechanism increases performance, the next set of results observe changes in the privacy trajectories as hyper-parameters are progressively adjusted. To compare the accuracy of the mechanisms, the reported results focus on the CIFAR-10 dataset as it has more distinct trends than the results of the MNIST and Pediatric Pneumonia datasets. Besause the CIFAR-10 dataset is used, we use the model described in Section 4.4.1. All other results are numerical evaluations of the privacy curves themselves.

The first set of results, shown in Figure 4.5, depicts the average test accuracy attained by the convolutional neural net trained for 200 epochs as a function of the injected noise's scale γ. From this graph we observe two trends as anticipated from the expected distortion, equation (18). First, as the scale γ increases, the accuracy of the model decreases. This is attributed to the linear relation between the expected distortion and the scale γ. Second, as the stability parameter α decreases, we observe a further decrease in accuracy. This too can be explained by the expected distortion (18), which is known to monotonically increase as α decreases [94]. Observe that when α is close to 2, e.g., $\alpha = 1.999$ in Figure 4.5, there is a negligible difference in accuracy between the SaS mechanism and the Gaussian mechanism. This result further aligns with the expected

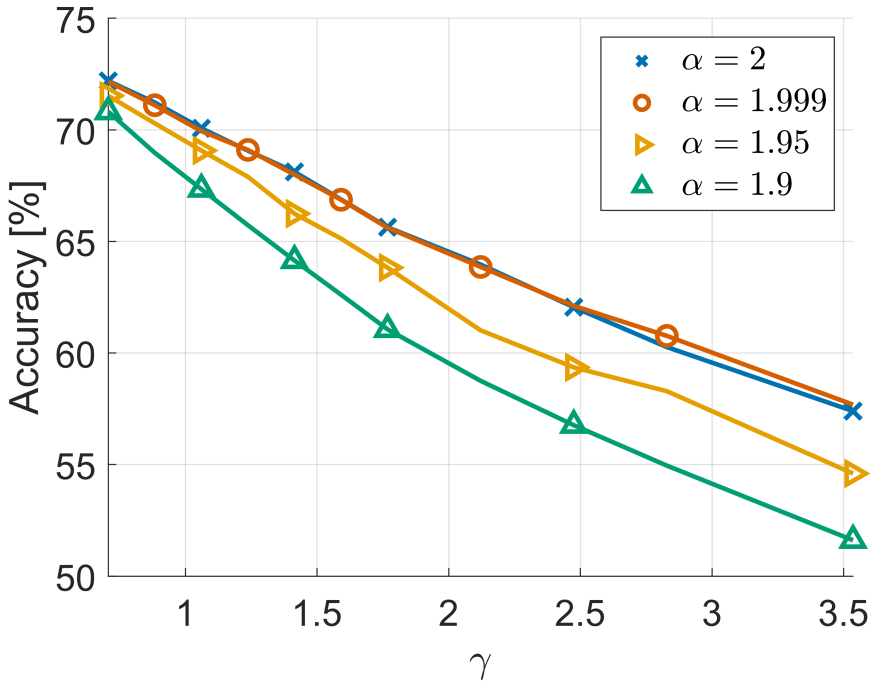

Figure 4.5: The averaged results of ten trials using the CIFAR-10 dataset and Opacus' standard implementation of DP-SGD. The legend provides the stability parameter for different alpha-Stable distributions. When $\alpha = 2$, shown in **blue**, the SaS mechanism is equivalent to the Gaussian mechanism. When the spread of the noise increases there is a diminishing of accuracy for all models. Likewise, as the weight of the density's tail increases, the accuracy of the model decreases. When the stability parameter is close to 2, $\alpha = 1.999$ shown in **orange**, the accuracy is essentially equivalent between the SaS and Gaussian mechanisms.

distortion provided in Table 3.1, which shows a 0.044% difference in expected error from the injected noise. Thus, Figure 4.5 demonstrates that the SaS mechanism is capable of achieving accuracy results comparable to that of the Gaussian mechanism.

We now proceed to demonstrate the superior privacy afforded by the SaS mechanism. Figure 4.6 summarizes the results of the numerical evaluation of the privacy curves shown in Figure 4.1 using 200 compositions. As expected, when the scale of the injected noise increases, so to does the final level of privacy (through the decreasing of ε). However, we observe a gap in performance between the SaS mechanisms and the Gaussian mechanism. While one might initially assume that the level of privacy achieved should vary smoothly between all the results, recall that the SaS mechanism with $\alpha < 2$ has an undefined second moment, i.e. the density has infinite

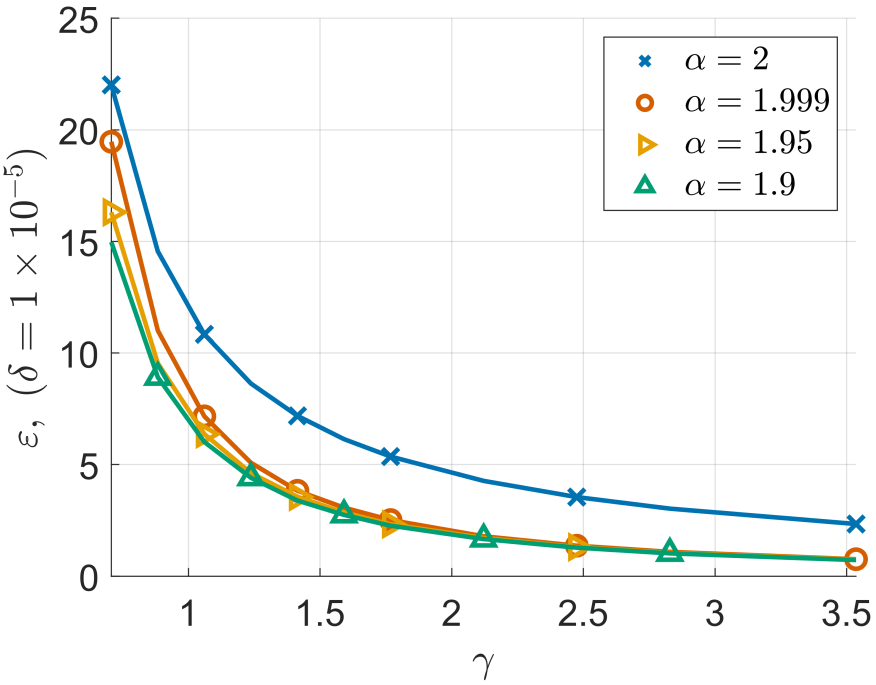

Figure 4.6: The level of privacy, ε, achieved by a selection of privacy mechanisms with their stability parameter α given in the legend. For a fixed α, level of privacy is determined for a given scale γ of the injected noise. For this calculation δ is held fixed at 1×10^{-5} and the sampling ratio p is set to 0.001. Observe that there exists a gap between the Gaussian mechanism ($\alpha = 2$ in **blue**) and all other SaS mechanisms (in **orange**, **yellow**, and **green**). We attribute this gap to the jump from finite to infinite variance in the noise that causes the divergence in privacy curves seen in Figure 4.1.

variance. In comparison, the Gaussian has finite variance. We proceed to demonstrate that this gap stems from the superior privacy curves of the SaS mechanism, Figure 4.1.

There are several factors that affect the privacy curves in Figure 4.1, resulting in the measured privacy seen in Figure 4.6: the acceptable privacy error δ, the subsampling ratio p, and the number of epochs N. First, because the SaS mechanism is purely differentially private, we observe a horizontal asymptote in the privacy curve. In comparison, the privacy curve for the Gaussian mechanism diverges since δ must be strictly greater than 0. Thus, as δ tends to 0, we expect to observe a greater difference in the achieved level of privacy between the SaS and Gaussian mechanisms. Figure 4.7 displays this effect through a sequence of five subplots. Each plot depicts the achieved privacy ε as a function of the injected noise scale γ for a selection of

δ values. As expected from the privacy curves, Figure 4.7 shows downward pressure on the SaS

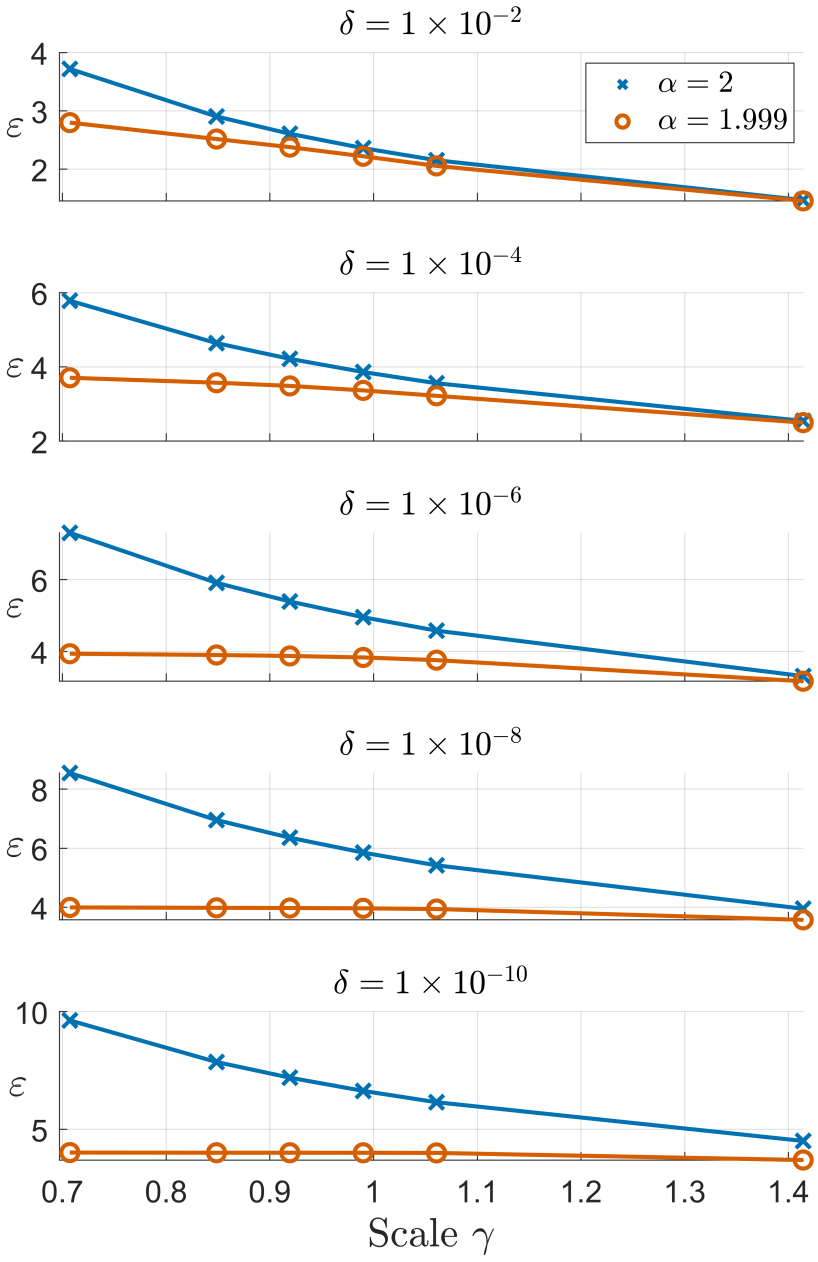

Figure 4.7: A sequence of 5 graphs, each showing the achieved privacy budget ε for the Gaussian mechanism in **blue** and the SaS mechanims with $\alpha = 1.999$ in **orange**. Each graph in the sequence decreases δ by an order of magnitude. These calculations are done with subsampling disabled by setting $p = 1$. As δ decreases, the privacy worsens for both mechanisms. Notably, the results of the Gaussian mechanism progressively diverge from the results of the SaS mechanism as δ decreases.

mechanism's privacy curve as δ tends to 0. This is made more evident by observing the privacy

at the scale values below 1 in Figure 4.7, which appear to approach an asymptote at $\varepsilon \approx 4$ for this given setup.

The second parameter that effects the level of privacy is the number of epochs taken during model training. The number of epochs is directly related to the number of compositions of privacy curves, which by Theorem 2.6.6, is the number of convolution of PRVs. Figure 4.8 depicts a sequence of privacy graphs that vary in the number of training epochs. We observe in Figure 4.8 a restoring force that brings the privacy of the SaS mechanism closer to that of the Gaussian mechanism. These results align with what is observed in Figure 4.9, which depicts the original privacy curves and a single composition of such curves.

Lastly, we examine the effect of the subsampling ratio on the privacy curves [70]. Figure 4.10 presents five plots, each with progressively smaller subsampling ratios. The decreasing subsampling ratio appears to separate and render the privacy curves equidistant. This flattening, observed in Figure 4.10, corresponds to the trend toward equidistant privacy curves in Figure 4.11.

Recall that for the CIFAR-10 trials, we used $\delta = 1 \times 10^{-5}$, 200 epochs, and a subsampling ratio of $p = 0.001$ to achieve the privacy trajectories in Figure 4.7. When combined with the test accuracy results in Figure 4.5, we obtain Figure 4.12. This figure illustrates the improvement in test accuracy obtained by using the SaS mechanism instead of the Gaussian mechanism for a given privacy budget. The results reveal significant improvements in test accuracy when using the SaS mechanism, particularly at higher levels of privacy. Table 4.4 showcases the percentage increase in accuracy on the CIFAR-10 dataset obtained by switching from the Gaussian to the SaS mechanism for different privacy budgets. These results further emphasize the effectiveness of the SaS mechanism in enhancing privacy on Machine Learning tasks.

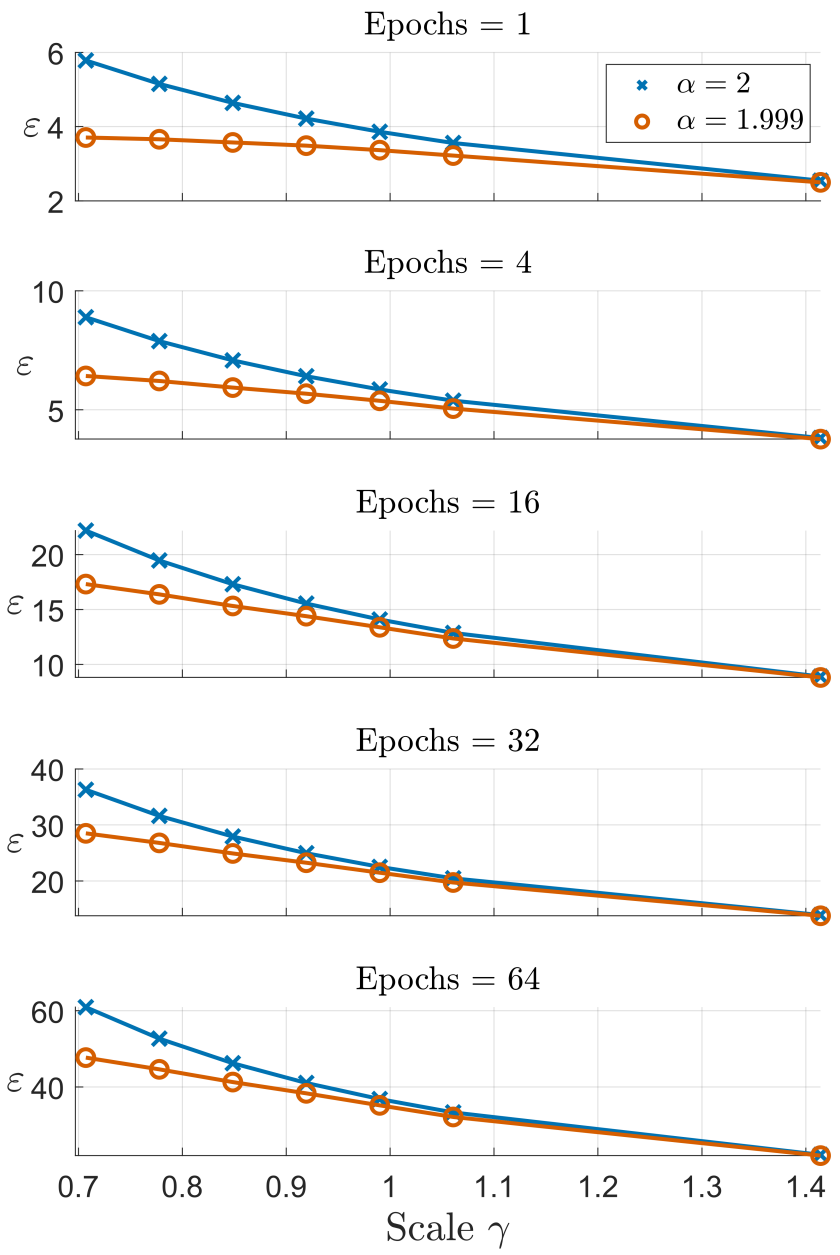

Figure 4.8: A sequence of 5 graphs that depict how changing the number of epochs affects the privacy curves of the Gaussian in **blue** and SaS mechanism **orange**. The subsampling ratio is disabled with $p = 1$ and the accepted privacy error δ is set to 1×10^{-4}. Each epoch corresponds to the composition of privacy curves. While the overall privacy decreased with each composition, the two curves additionally come closer together over iterations.

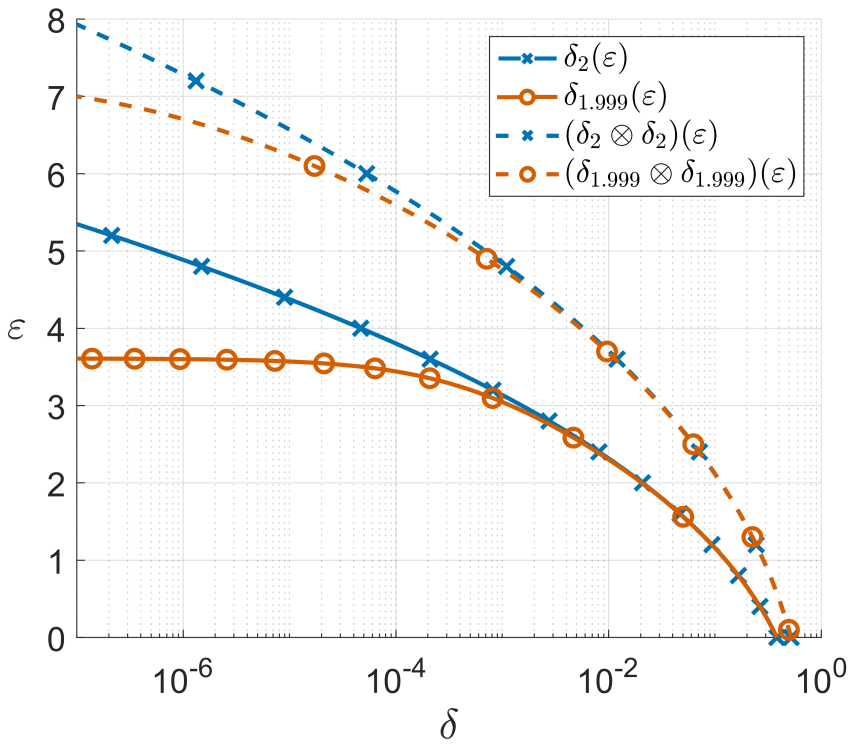

Figure 4.9: The privacy curves for the Gaussian mechanism ($\alpha = 2$ in **blue**) and the SaS mechanism ($\alpha = 1.999$ in **orange**) are depicted with solid lines. After a single composition, each with themselves, the privacy curves shift up and are closer together. The composed curves are shown with a dotted line.

4.7 Concluding Remarks

We have demonstrated that the Symmetric alpha-Stable mechanism exhibits significant privacy improvement over the standard Gaussian mechanism within a Differentially Private framework. We trained Convolutional Neural Networks on the benchmark datasets MNIST, CIFAR-10, and Pediatric Pneumonia and demonstrated that the SaS mechansism achieves higher testing accuracy for a given privacy budget on all accounts. At its best, our results indicate a near 20% improvement in accuracy with the privacy budget of $\varepsilon = 1$ on the CIFAR-10 dataset. The results are more muted on the MNIST and Pediatric Pneumonia datasets, achieving the greatest accuracy improvement of 3% with $\varepsilon = 0.5$ and 5% with $\varepsilon = 0.6$ respectively.

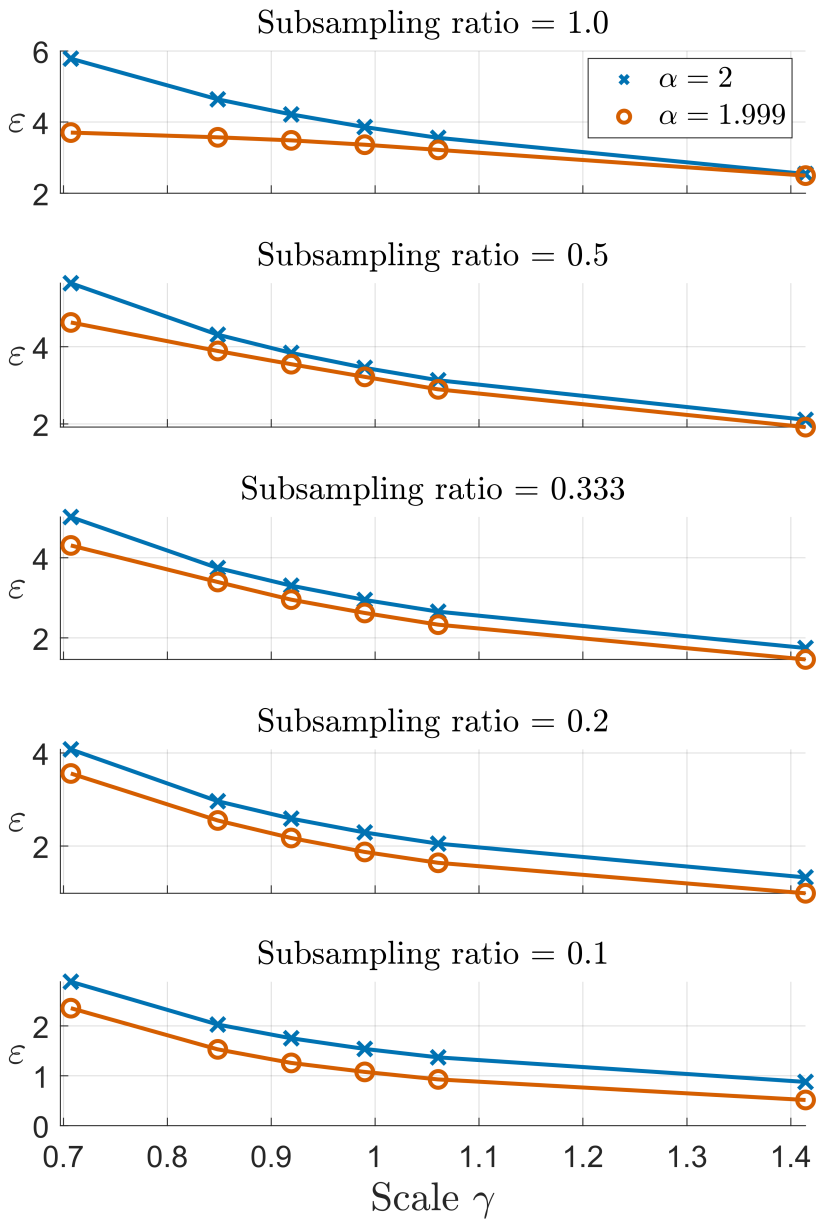

Figure 4.10: A sequence of 5 graphs that depict how changing the number of subsampling ratio affects the privacy curves of the Gaussian (in **blue**) and SaS mechanism (in **orange**). The values of the graphs are calculated for a single epoch and with the accepted privacy error $\delta = \times 10^{-4}$. As the subsampling ratio decreases, the achieved privacy budget for the Gaussian and SaS mechanisms decrease and become increasingly equidistant.

We attribute the improvement to the infinite variance possessed by the SaS density, which is major jump compared with the finite variance of the Gaussian density. When the stability parameter α is close to 2 the SaS mechanism exhibits testing accuracy equivalent to that of the

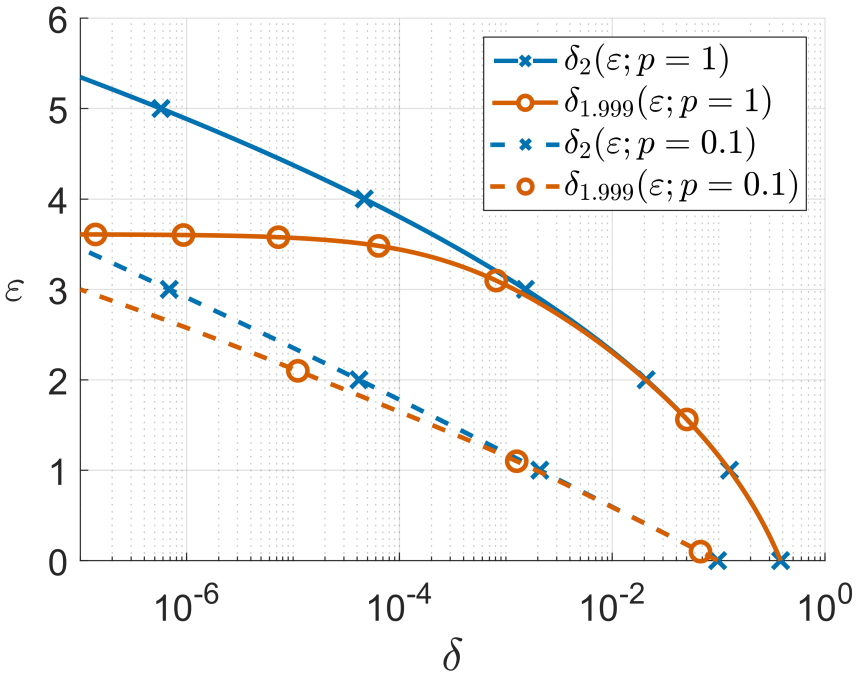

Figure 4.11: The privacy curves for the Gaussian mechanism ($\alpha = 2$ in **blue**) and the SaS mechanism with $\alpha = 1.999$ (in **orange**) are depicted with subsampling disabled ($p = 1$) in solid lines. For comparison, the privacy curves with a subsampling ratio of $p = 0.1$ are given with dotted lines.

Gaussian mechanism for a given scale γ.

It is worth noting that the results presented in this study have not been optimized to achieve state-of-the-art performance (this is due to limited hardware for training). Instead, these results are designed to highlight the direct comparison of the SaS and Gaussian mechanisms with all other settings being equal. Even still, the results are comparable with that of significantly more capable models. Moreover, an immediate avenue for future work is investigate the relationship between the expected distortion and the privacy curves to calculate an optimal tail weighting α.

The results underscore the effectiveness of the SaS mechanism as a promising approach for enhancing privacy in Machine Learning applications. The findings contribute valuable insights for researchers and practitioners working on implementing Differential Privacy mechanisms, encouraging further exploration and refinement of privacy-preserving techniques.

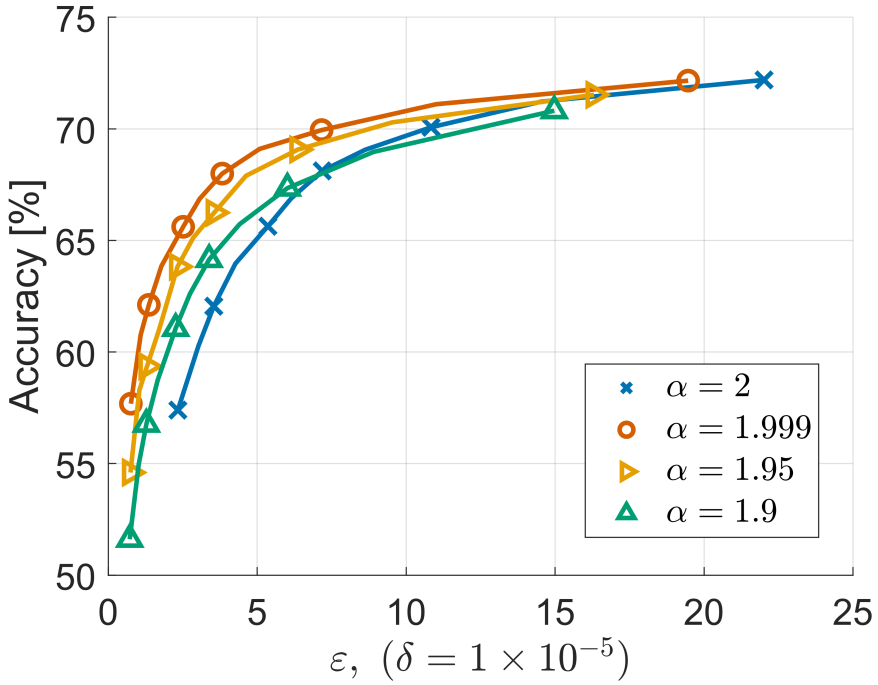

Figure 4.12: Here we provide the test accuracy achieved on the CIFAR-10 dataset as a function of the privacy budget ε. Observe that the usage of a SaS mechanism with $\alpha = 1.999$ in **orange** achieves a significant improved privacy value for a given testing accuracy when compared to the Gaussian mechanism ($\alpha = 2$ in **blue**), especially in the range $\varepsilon \in (0, 8]$. This is because the SaS mechanism is able to inject noise with a smaller scale for a given privacy budget, thus increasing accuracy. We obtained these results training the model for 200 epochs with a subsampling ratio of 0.001.

Part II

Federated Learning

Chapter 5

Background for Federated Learning

In this section we review the mathematical foundations of Machine Learning and provide the environmental constraints that extend it to define the field of Federated Learning. We then present an overview of the development of the field and discuss the challenges and limitations of existing methods.

5.1 Overview of Federated Learning

The shift from knowledge-driven to data-driven Machine Learning methods has resulted in a surge of data collection efforts, which have corresponded with rise in the number of independent data breaches [2]. The growing number of isolated databases increases the attack surface for adversarial agents, and the centralized nature traditional Machine Learning systems create a single point of failure. If a central server is compromised, thousands to millions of individuals can be affected at once.

Federated Learning (FL) is a branch of Machine Learning which focuses on using a *federation* of client devices to train Machine Learning models. This method is in stark contrast with traditional systems, which are built to store user data and train a global model on a centralized

server. Instead of collecting user data, FL trains the model on the client devices, which allows the data to remain distributed. Federated Learning systems increase privacy by limiting multiple copies of the same information while also preventing single point failures affecting possibly millions of individuals. The authors of [18] recognized this privacy issue and developed a method where, instead of sending the data to the model, the model is distributed to the data. In their method, clients maintain ownership of their data, and only transmit model updates to the server. The authors termed this method of Machine Learning *Federated Learning*.

5.2 Mathematical Foundations of Federated Learning

In this section, we review the mathematical construction of a Machine Learning problem and provide the assumptions and constraints which differentiate Federated Learning from the rest of the field.

5.2.1 Machine Learning

The goal of Machine Learning is to build models from data to make predictions. To build a model that identifies the presence of a particular disease, we first collect a dataset by obtaining a group of patients comprised of individuals with either a positive or a negative diagnosis. From these patients we collect a set of measurements or features we believe might correlate to a positive or negative diagnosis. For example, the features may include the patients height, weight, blood pressure, glucose levels, ...etc. We denote the space of features by $\mathcal{X} \subseteq \mathbb{R}^d$. The value space, denoted by $\mathcal{Y} \subseteq \mathbb{R}^q$, corresponds to the space of values we seek to predict. For the presence or absence of a disease, the value space can be taken as the finite set $\{0, 1\}$. The value $\mathbf{y}$ that

corresponds with a features vector **x** is known as its label. As such, we define a labeled dataset as the product space of a feature space and a value space.

Definition 19. *(Labeled Dataset) Let $\mathcal{X}$ and $\mathcal{Y}$ represent a space of features and corresponding values respectively. Denote by $\mathcal{D}$ the set of n tuples $\{(\boldsymbol{x},\boldsymbol{y})\}_n$ where $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$. For a given tuple $(\boldsymbol{x},\boldsymbol{y})$, the $\boldsymbol{x}$ is known as the feature, and the value $\boldsymbol{y}$ is known as the label. Thus, we call $\mathcal{D}$ a labeled dataset.* ◀

Datasets in this contexts are generally finite with the size of the dataset denoted by $|\mathcal{D}| = n$. An element of a labeled dataset is denoted by $(\mathbf{x}, \mathbf{y}) \in \mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$. Often, datasets are split into training and testing partitions. This split allows for a fair evaluation of our Machine Learning method by preventing the memorization of the dataset by a sufficiently complex model.

A Machine Learning model represents our assumptions about the relationship between the feature and value spaces. The model is tuned on available data and is used to make predictions about new observations.

Definition 20. *(Machine Learning Model) Denote by $\mathcal{X}$ and $\mathcal{Y}$ a feature and value space respectively. A Machine Learning model is a function f that maps the elements from the feature space to corresponding elements in the value space. The model is parameterized by a vector $\boldsymbol{w}$ from a parameter space $\mathcal{W} \subset \mathbb{R}^p$. Thus a Machine Learning model is a function that maps the product space of features and parameters to the value space:*

$$f : \mathcal{X} \times \mathcal{W} \to \mathcal{Y}. \tag{5.1}$$

◀

Assumption 1. *(Compact Parameter Set) We assume that the set $\mathcal{W}$ of possible parameters is compact.* ◀

Assumption 1 ensures there exists a nonempty set of minimal values. Once we have defined a Machine Learning model, we must tune the model so that it provides an accurate prediction for $\mathbf{y}$ that corresponds to a given $\mathbf{x}$. This is accomplished through the use of a function that evaluates the predictive quality of the model on a dataset $\mathcal{D}$ for different choices of parameters $\mathbf{w}$.

Definition 21. *(Machine Learning Loss Function) Let f be a Machine Learning model designed to predict the labels that corresponds to features of a given labeled dataset $\mathcal{D}$. A loss function, denoted J_f, tunes a model's parameter vector $\boldsymbol{w}$ to accurately predict the corresponding label of a feature-value pair $(\boldsymbol{x}, \boldsymbol{y})$:*

$$J_f : \mathcal{W} \times \mathcal{D} \to \mathbb{R}. \tag{5.2}$$

The value of a loss function represents the loss incurred upon choosing one set of parameters $\boldsymbol{w}$ over another $\boldsymbol{w}'$. ◀

The objective of a Machine Learning algorithm is to determine the optimal parameter vector $\mathbf{w}^*$ that minimizes this cost function:

$$\mathbf{w}^* := \arg\min_{\mathbf{w}} \; J_f(\mathbf{w}; \mathcal{D}). \tag{5.3}$$

Once $\mathbf{w}$ is selected for a specific model structure, the model is completely determined. Because of this, sometimes in the literature the parameter $\mathbf{w}$ itself is referred to as "the model."

In a typical Machine Learning application, the data is stored on a central server. If, for

example, the data is collected sequentially in time, neighboring data points may have strong mutual dependence. A major benefit of a central database is that data can be rearranged to reduce the dependency between clusters of data. It is through the central database that Machine Learning incurs a high privacy loss. A central database creates a single point of failure where one security flaw may affect a large number of individuals. In the next section, we briefly discuss Federated Learning, which seeks to maintain privacy through controlling how data can be shared with the server.

5.2.2 Federated Learning

Federated Learning aims to ensure user privacy through limiting the data shared by client devices with the central server. This is done by allowing the dataset to reside in a distributed among participating client devices. Let K be the number of clients. We then represent the entire dataset as the union of these K separate clusters, denoted by $\mathcal{D} := \mathcal{D}_1 \cup \mathcal{D}_2 \cup ... \cup \mathcal{D}_K$. Denote the size of the k^{th} dataset by $|\mathcal{D}_k|$.

The communication structure between participants is expressed as a graph with each client (including the central server) represented by a vertex v and each communication channel by an edge e.

Definition 22. *(Graph) Let the set of all vertices and the set of all edges be denoted by $\mathcal{V}$ and $\mathcal{E}$ respectively.Then a graph $\mathcal{G}$ is defined to be the combined set of all vertices and edges, $\mathcal{G} := (\mathcal{V}, \mathcal{E})$.* ◀

The communication architecture on a graph $\mathcal{G}$ can be centralized or decentralized. Centralized communication refers to the use of a central server to coordinate communication. Figure 5.1

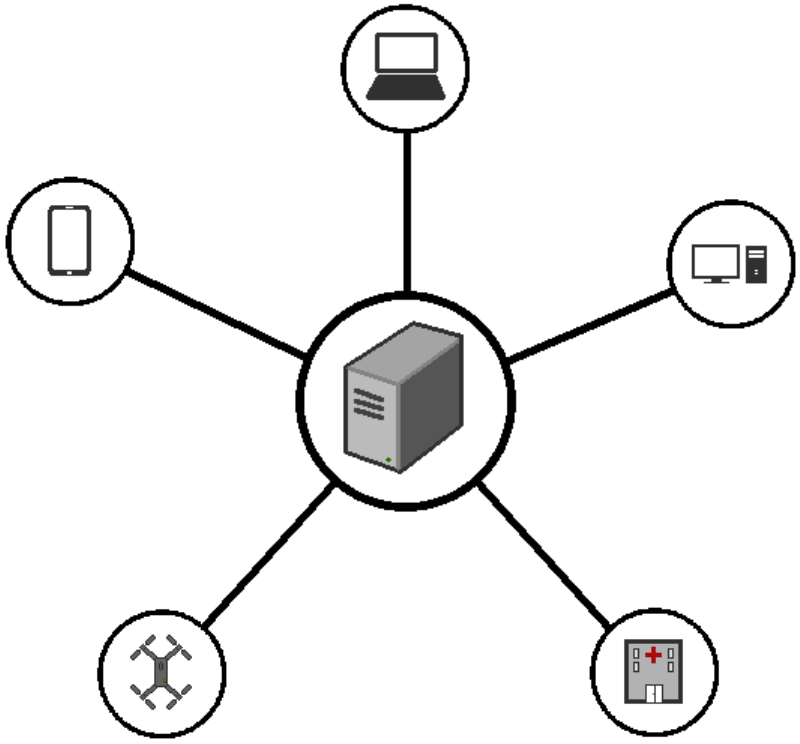

Figure 5.1: The architecture of the graph above is known as a star graph. Communication on a star graph is coordinated through the central server located at the hub. Centralized Federated Learning generally stores a global model on the server. The client edge devices can be phones, personal computers, drones, ...etc, and maintain ownership over their data. This can be beneficial when the data is sensitive, such as patient records in hospitals.

illustrates the common star graph structure of centralized architectures. The server is located at the hub and handles all communication with the clients. In contrast, decentralized communication occurs when clients directly communicate with each other without the use of a central server. This type of communication is known as Peer-to-Peer (P2P).

Depending on the communication structure, Federated Learning can be centralized or decentralized [107]. In centralized FL, a central server maintains and updates a global parameter vector through its communication with the clients. Likewise, Federated Learning over P2P networks is known as decentralized FL. In this work, we focus on centralized Federated Learning,

though extending the work to decentralized networks is an important direction for future research.

Federated Learning makes three key assumptions about this networked environment:

- Client data cannot be transparently transmitted to the server.
- Client devices are not standardized.
- The number of clients may be significantly larger than the size of each individual dataset.

The last assumption is indicative of the type of scenario that Federated Learning is geared toward. This is depicted in Figure 5.1 by the variety of client devices at the leaf nodes of the graph.

Each of these assumptions is made to differentiate Federated Learning from distributed learning, where a dataset may be too large to train on a single system. Distributed learning *distributes* the data to a handful of compute clusters. These clusters are generally standardized and designed for the computation at hand. Distributed learning algorithms can assume the data has been shuffled into nearly independent and identically distributed (i.i.d.) sets prior to training.

Each of the assumptions above can be related to three core challenges:

- **Privacy:** Suggested methods should protect the identity of the individuals involved by being robust to adversarial attacks.
- **System Heterogeneity:** Training primarily takes place asynchronously on client's devices. Some client devices may lag in returning updates, or drop out the network.
- **Statistical Heterogeneity:** Data might not be i.i.d., and local datasets may have different local optima.

It is these challenges that federated algorithms seek to overcome when training a model to fit

a dataset. Synchronous methods (FL or otherwise) struggle in this environment because they require current information, which may be insufficient if clients lag or disconnect.

Since the datasets must remain distributed, each client maintains a local model and loss function. In this work, we take the dataset held by each client to be fixed. As such, the local model and local loss function can be viewed as a function of the parameter $\mathbf{w}$ only. The local model is then

$$f_k := f(\mathcal{X}_k, \cdot) : \mathcal{W} \to \mathcal{Y}, \tag{5.4}$$

and local loss function is

$$J_k := J_{f_k}(\cdot\,; \mathcal{D}_k) : \mathcal{W} \to \mathbb{R}. \tag{5.5}$$

Assumption 2. *(Like Models) In this work, we restrict the problem to identical model structures across all clients, i.e. each client has the same f.*

◀

This assumption is common across the literature [107]. It is worth observing that differences in the local datasets induce different local optimal parameters $\mathbf{w}_k^*$ from (5.7).

Due to the distributed nature, it is necessary to define a global loss function which takes into account the loss associate with each of the K clients.

Definition 23. *(Global Cost Function [18]) Let $\{J_1, ..., J_K\}$ be a collection of local loss functions for a set of K clients. The global loss function $J(\boldsymbol{w})$ is defined to be the weighted sum of local losses:*

$$J(\boldsymbol{w}) := \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} J_k(\boldsymbol{w}). \tag{5.6}$$

◀

The objective of a Federated Learning algorithm is to minimize the global loss function with respect to the global parameter vector **w**:

$$\mathbf{w}^* := \arg\min_{\mathbf{w}} J(\mathbf{w}). \tag{5.7}$$

5.3 Types of Adversaries

There are two main classifications of adversarial agents: passive and active. Passive adversarial agents observe communications without actively interfering. For example, if an adversarial agent is able to listen to communication between a target client and the sever, unprotected client information will be leaked, but the adversary will not affect model training. Another example of a passive adversary is an untrusted central server. In this case, clients may wish to obfuscate their exact information prior to transmission to the server. Local Differential Privacy can be used in both cases to protect the client. Active adversaries directly tamper with the training process in order to gain an advantage over the system. This could be by poising the model with false data to decrease the accuracy of the trained model or by injecting malicious code to exploit vulnerabilities on the training system. The two most common types of active attacks considered on FL systems are poisoning attacks and Byzantine attacks [108]. In a data poisoning attack, the adversary aims to compromise the integrity and effectiveness of the learned model by injecting *poisoned* data through the training process. The process seeks to induce bias into the learned model to distort the response on specific tasks. By allowing clients to calculate their own model updates, Federated Learning is more susceptible to these types of attacks than traditional methods

where the central server can clean suspicious data from the dataset through pre-processing. The second type of attack, known as a Byzantine attack, which broadens the attack to the collusion of multiple clients within the training process. Recent work seeks to develop methods of detecting suspicious behavior and identifying which clients are controlled by a Byzantine agent [109]. In this work, we chose to limit our scope to passive adversaries.

5.4 Existing Solutions

We next present a selection of important contributions to the field of Federated Learning, beginning with its inception by McMahan et al. in 2017 [18] to present day applications.

5.4.1 Federated Averaging

In the seminal work which introduced the concept of Federated Learning, McMahan et al. proposed in [18] an algorithm known as Federated Averaging (FedAvg), that trains Machine Learning models on a distributed set of client devices. In traditional Machine Learning settings, data is often collected and centralized on a single server for training. However, this centralized approach raises privacy concerns, especially when dealing with sensitive user data. FedAvg addresses this challenge by allowing training to be performed locally on individual devices, such as smartphones or Internet of Things (IoT) sensors, without transparently transmitting raw data with a central server.

In Federated Averaging (FedAvg), at the beginning of each round, $t > 0$, the server selects a subset of participating devices, S^t, and broadcasts the current global model, $\mathbf{w}^t$. Each participating device performs s_k local gradient steps. The resulting change in model parameters,

$\Delta_k^t = \mathbf{w}_k^{t,s_k} - \mathbf{w}_k^t$, is sent back to the server. Each update is assigned a weight, p_k, referring to importance of a client's local data set. The server aggregates the updates as a weighted average,

$$\Delta^t = \frac{\sum_{k \in S^t} p_k \Delta_k^t}{\sum_{k \in S^{(t)}} p_k}. \tag{5.8}$$

With a given learning rate, η, the global model is updated by taking a step in the direction of decreasing gradient,

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \Delta^t. \tag{5.9}$$

The process loops until a termination condition is met.

This learning process is depicted in Figure 5.2. The FedAvg algorithm operates through a process of collaboration between the central server and the distributed client devices. Initially, the server broadcasts a global model to all participating clients. Each device then independently trains the model using its local data. Next, only the model updates, in the form of parameter gradients, are transmitted back to the central server. This ensures that sensitive data remains on the device. Once the all clients have finished local training, the central server collects the model updates from all devices and averages them to calculate a gradient step for the global model. The new global model is then broadcast back to the devices, initiating the next round of training. Through this iterative process, the global model gradually improves while preserving the privacy of individual data sources.

When the number of local steps taken between global updates is one, $s_k = 1$, the algorithm is a variation of projected stochastic gradient descent and is known to converge [110]. This sub-variant of FedAvg is known as FedSGD. When multiple local steps are taken in FedAvg, $s_k > 1$,

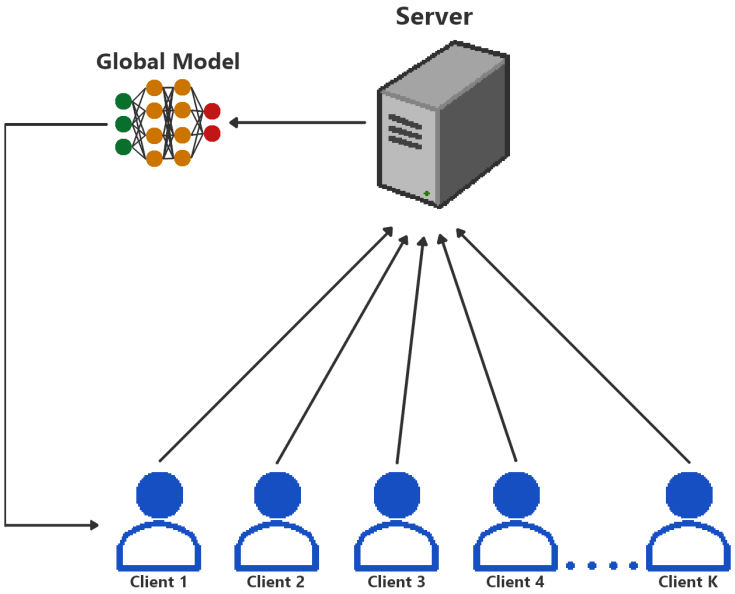

Figure 5.2: In a Federated Learning environment, a set of K participating clients collaboratively train a Machine Learning model. The server maintains a set of global model parameters. Rather than have the clients pass their data to the server, the server transmits these model parameters to each client. The participants then calculate a model update using their local data and transmit only the update to the server. The server then aggregates the updates, modifies the global model, and transmits the new model to the clients. This repeats until the training converges.

it is shown by Pathak et al. [111] that the algorithm does not necessarily converge to a minimizer.

FedAvg offers several advantages over centralized training approaches. Firstly, it enables training on decentralized data sources without compromising privacy, making it suitable for applications in healthcare, finance, and other sensitive domains. Additionally, by distributing computation across multiple devices, FedAvg can leverage the computational power of edge devices, which allows the training of complex Machine Learning models without the cost barrier of large central servers. Moreover, because Federated Learning assumes clients may have limited connectivity to participate in the learning process, it is more robust to heterogeneous network environments. Overall, Federated Averaging represents a promising approach for collaborative and privacy-preserving Machine Learning in a distributed setting.

Global updates are only made when all participating clients have returned information to the server. By synchronizing model updates, the method avoids the problem of integrating old

data from a straggler. Issues from statistical heterogeneity are minimized by uniformly randomly selecting the participating subset. If we select a large enough subset to participate in each round, by the linearity of the gradient, we approximate an IID data selection and minimize dependencies. However, this is shown not to be robust in all circumstances. FedAvg suffers from client drift in the presence of sufficient statistical heterogeneity [111–116]. Client drift occurs when each client model shifts toward the minimum of its local loss function more so than the global optimum. This drift slows down the algorithm and, in the worst case, causes convergence to the wrong point. The work provides a good starting point from which the field has begun to flourish. A selection of more recently developed works that target the aforementioned shortcomings is provided below.

5.4.2 Federated Proximal Learning

In order to more rigorously handle statistical heterogeneity, the authors of [117–120] provide extended FedAvg to guarantee convergence in the presence of non-IID data. However, these additional works assume that the system is synchronized with all devices participating each round. This makes the algorithm susceptible to system heterogeneity. One of the first attempts to handle statistical and system heterogeneity is proposed by Li et al. [121], known as FedProx. The authors propose using a proximal term to stabilize the gradient step in the presence of heterogeneous datasets. Their method, named, FedProx, is an extension of the Federated Averaging (FedAvg) algorithm that addresses issues related non-IID (non-identically distributed) data among decentralized devices. While FedAvg assumes that all devices have similar data distributions and contributions, real-world scenarios often involve devices with diverse data characteristics, such as varying data distributions or different amounts of available data. FedProx introduces a regu-

larization term to mitigate the effects of data heterogeneity and non-IIDness, thereby improving the convergence and generalization performance of Federated Learning models.

Similar to FedAvg, the FedProx algorithm begins with the central server broadcasting a global model to all participating devices. Each device independently trains the model using its local data, but with the addition of a proximal term during optimization. This proximal term penalizes discrepancies between the local model parameters and the global model, encouraging devices to stay close to the global model while still adapting to their local data distributions. By incorporating this regularization mechanism, FedProx promotes greater model consistency across devices.

Specifically, each device k minimizes the new objective h_k,

$$\min_{\mathbf{w}} h_k(\mathbf{w}; \mathbf{w}^t) = f_k(\mathbf{w}) + \frac{\mu}{2}||\mathbf{w} - \mathbf{w}^t||^2. \tag{5.10}$$

The right most term acts as a regularizer, keeping the updates close to the current global model $\mathbf{w}^t$. The hyperparameter μ weights the importance between the regularizer and the objective function, $f_k(\mathbf{w})$. The proximal term prevents statistical heterogeneity from dominating an update.

The authors of FedProx also proposed a partial work strategy to mitigate the effect of slower devices while taking advantage of faster clients. By rephrasing the local objective into a minimization, clients can solve the optimization inexactly. This technique prevents slower devices from dragging down the convergence rate of the whole network. The authors defined a γ-inexact solution, $\mathbf{w}^*$, as one that solves the local minimization while satisfying the following constraint,

$$||\nabla h_k(\mathbf{w}^*, \mathbf{w}_0)|| \leq \gamma||\nabla h_k(\mathbf{w}_0; \mathbf{w}_0)||, \tag{5.11}$$

with h_k defined above and $\gamma \in [0, 1]$. The notion of partial work introduces a trade off between computation and communication, preventing the system from being blocked by long computation times. Proximal Federated Averaging is better designed to handle system and statistical heterogeneity with the proximal term partial work strategy. The pair of upgrades work together to fully utilize client's computational power while allowing for guaranteed convergence.

FedProx offers several advantages over FedAvg, particularly in scenarios when the distribution of data vararies greatly across the participating clients. By regularizing the optimization process, FedProx mitigates the negative impact of data diversity, leading to more robust and generalizable Federated Learning models. FedProx represents a significant advancement in Federated Learning, enabling effective model training in decentralized environments with diverse data characteristics.

The method is not without its drawbacks. The weighting term the authors use to accommodate statistical heterogeneity must be empirically tuned. When not tuned well, convergence rates can suffer substantially. Moreover, the variable work principal can prevent slower devices from sending critical information if the update takes too long to calculate. As a result, there exists a trade off between convergence speed and accuracy [115,116]. When the learning rate is held constant the convergence may be to a non-globally optimal point, leading to a lowing of the step size and slowdown on total convergence. It is shown in [111, 122] that under statistical heterogeneity the convergence is not necessarily to the minimizer. Similar to Federated Averaging, FedProx is only guaranteed to converge to the correct minimizer when the local objectives, f_k, are common to all devices.

5.4.3 Stochastic Controlled Averaging for Federated Learning

To address the impaired convergence rates of FedProx, the authors of [112] extended variance reduction techniques discussed in [123–125] to the Federated Learning environment. They propose an algorithm, named Stochastic Controlled Averaging for Federated Learning (SCAFFOLD)[1], to correct for client drift. The method is robust to statistical heterogeneity and shows improved convergence rate when compared with FedAvg and FedProx. The authors give convergence bounds for varying levels of convexity as well as convergence to a first order extrema for non-convex functions.

In their work, each client k maintains a local model which is indexed by the round t and the local step i: $\mathbf{w}_k^{t,i}$. A control variate for round each is held by both the server and by each client, given by $\mathbf{c}^t$ and $\mathbf{c}_k^t$ respectively. Each round begins by selecting a subset of participating clients, S^t, and initializing their local model with the current global parameters,

$$\mathbf{w}_k^t = \mathbf{w}^t, \quad k \in S^{(t)}. \tag{5.12}$$

In parallel, the subgroup of participating nodes each calculate the gradient $g_k(\mathbf{w}_k^{t,i})$ and update their local model for a prescribed number of steps $i \in s_k$,

$$\mathbf{w}_k^{t,i+1} = \mathbf{w}_k^{t,i} - \eta_l(g_k(\mathbf{w}_k^{t,i}) - \mathbf{c}_k^t + \mathbf{c}^t). \tag{5.13}$$

[1]Reference [112] explains that the meaning of SCAFFOLD as Stochastic Controlled Averaging For Federated Learning, so all the letters in the acronym comem from the name.

The authors provide two options for updating the local control variate.

$$\mathbf{c}_k^+ = \begin{cases} g_k(\mathbf{w}^t) \\ \mathbf{c}_k^t - \mathbf{c}^t + \frac{1}{N\eta_l}(\mathbf{w}^t - \mathbf{w}_k^{t,N}) \end{cases} \tag{5.14}$$

The first option makes an additional pass over the data and results in more stable convergence. Option two is cheaper and sufficient for each example they provide. The changes in model weights and control variate are then transmitted back to the server,

$$(\Delta \mathbf{w}_k, \Delta \mathbf{c}_k) = (\mathbf{w}_k^{t,s_k} - \mathbf{w}^t, \mathbf{c}_k^+ - \mathbf{c}_k^t). \tag{5.15}$$

Once all the participating clients have returned an update, the server updates the global model,

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \frac{\eta_g}{|C|} \sum_{k \in C} \Delta \mathbf{w}_k, \tag{5.16}$$

$$\mathbf{c}^{t+1} = \mathbf{c}^t + \frac{1}{K} \sum_{k \in C} \Delta \mathbf{c}_k. \tag{5.17}$$

While the method is well suited for statistical heterogeneity, the lack of a solution to system heterogeneity make this method susceptible to the stragglers. Furthermore, there is evidence that variance reduction techniques perform poorly on non-convex problems [126].

5.4.4 Federated Learning with Splitting Operators

While prior techniques guaranteed the algorithm would converge, they did not guarantee that the algorithm converged to the optimal solution. The authors of [111] put forth a class of

algorithms based on the splitting procedures for solving distributed convex minimization with additive structure. Operator splitting can be exploited to allow the development of provably correct convergence [127–130]. Specifically, their algorithm, named FedSplit, is based upon Peaceman-Rachford splitting [131]. This procedure is designed to reach the correct minimizer under broader conditions than previously considered. As stated above, FedAvg and FedProx are shown to only converge to the correct fixed point under the condition of a shared objective function between the clients. These conditions are unrealistic as we could avoid the network entirely if we only need a single client to reach the solution.

FedSplit is composed of three main steps. First, a local proximal solver is used to generate an initial update. In the ideal case, an exact evaluation of the proximal operator is be used. However, in a practical setting, an approximate is often quicker to calculate,

$$\hat{\mathbf{w}}_k^{t+1/2} = \text{prox\_update}_k(2 * \mathbf{w}^t - \hat{\mathbf{w}}_k^t). \tag{5.18}$$

Next, a centering step is used to readjust the estimate,

$$\hat{\mathbf{w}}_k^{t+1} = \hat{\mathbf{w}}_k^t + 2(\hat{\mathbf{w}}_k^{t+1/2} - \mathbf{w}^t). \tag{5.19}$$

Lastly, the local model parameters are sent back to the server for aggregation,

$$\mathbf{w}^{t+1} = \frac{1}{K} \sum_{k=1}^{K} \hat{\mathbf{w}}_k^{t+1}. \tag{5.20}$$

Given the necessity of approximate solvers, the method suffers from an accuracy-speed trade off. This is exacerbated by the lack of straggler mitigation. The authors note that extending

this technique to asynchronous updates is a next future step. The algorithm's dynamics have been examined under deterministic conditions, and it is not yet known if the results extend to the stochastic implementation.

5.4.5 FedNova

Wang et al. [122] posit that in order to account for the variability in speed of participating devices, clients, in a Federated Learning algorithms, must adjust the number of local steps accordingly. The authors generalize the results of FedAvg and FedProx with an algorithm named FedNova. The new method weights the local updates based on the number of local steps taken. This method preserves objective consistency while being easily merged with other algorithms that don't yet account for system heterogeneity. For a given training time period, the number of local steps is given by the number of complete passes of the data set per step, E, the number of data points on the local device, n_k, and the size of subset used for a local step, B,

$$s_k = \lfloor \frac{En_k}{B} \rfloor. \tag{5.21}$$

For the algorithms considered so far, the update rule becomes

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \sum_k^K s_{eff} \frac{n_k}{n} \Delta_k^t. \tag{5.22}$$

where $s_{eff} = \sum_k p_k s_k$, a weighted sum of the number of local steps. The authors further generalize these results to accommodate different variations of local solvers, e.g., decayed learning rate and momentum. While these generalization bring FedAvg, FedSGD, and FedProx into a single

algorithm, it does not fully alleviate the problems of all of them. Once more, a speed-accuracy trade off is introduced. Additionally, to achieve convergence to the correct point, the algorithm requires a the diminishing step size; this, in turn, induces a sublinear convergence rate.

5.4.6 Linear Federated Learning

More recently, the authors in [132] propose an algorithmic framework, named FedLin, with linear convergence guarantees. They utilizes gradient-tracking to account for statistical heterogeneity and client specific learning rates, η_i, to adjust for system heterogeneity. The authors provide tight linear convergence rates for smooth, strongly convex functions and provide some additional insight into how compression techniques affect convergence.

During a the local update step, the k^{th} client performs s_k local steps, with faster devices taking more steps. One downside to letting fast clients take more steps is that they will experience the most drift toward their local optimum. As a result, it is necessary to decrease their learning rate to mitigate this effect. The update rule for FedLin without compression is given by

$$\mathbf{w}_k^{t,i+1} \leftarrow \mathbf{w}_k^{t,i} - \eta_i(\nabla f_k(\mathbf{w}_k^{t,i}) - \nabla f_k(\mathbf{w}^t) + \nabla f(\mathbf{w}^t)). \tag{5.23}$$

During each local step, to estimate the global gradient, the client devices use the last known global gradient but replace their contribution with an updated term. The author's gradient-tracking is equivalent to the exact formulation of SCAFFOLD, described above, and the client specific learning rates share many similarities with the local step normalization process of FedNova. The authors conclude that, in the presence of statistical heterogeneity, when the objectives of local cost functions do not align, it is unclear if local steps are beneficial toward learning a global

model.

5.4.7 Mode-Contrastive Federated Learning

More recent approaches that seek to mitigate the effect of non-IID stem from the recognition that a global model trained on the entire dataset often outperforms a local model trained on a biased subset. From this intuition, Model-based Contrastive Learning (MOON), adjusts local updates to enhance agreement between the representations learned by the current local model and those learned by the global model [133]. Unlike traditional contrastive learning methods that excel in visual representation learning by comparing representations of different images, MOON operates at the model level, comparing representations learned by different models. Denote by $\mathbf{z}$ the intermediate representation learned by a model. For example, this would represent the output of a particular hidden layer in a neural network as opposed to the resulting classification of the last layer. Specifically, MOON introduces the following model-contrastive loss

$$\ell_{con} = -\log \frac{\exp(\text{sim}(\mathbf{z}, \mathbf{z}_{glob})/\tau)}{\exp(\text{sim}(\mathbf{z}, \mathbf{z}_{glob}/\tau)) + \exp(\text{sim}(\mathbf{z}, \mathbf{z}_{prev}/\tau))}, \tag{5.24}$$

where $\mathbf{z}$ and $\mathbf{z}_{glob}$ are the output of the local and global representations respectively, $\mathbf{z}_{prev}$ is the output of the previous global model's representation, τ is a temperature parameter, and $\text{sim}(\cdot, \cdot)$ is a cosine similarity function. This additional loss is then summed with the more traditional loss and used to take gradient steps while ensuring the internal representation learned by the local models does not deviate far from the global model. Moon is not the only algorithm that uses a contrastive approach to enhance performance on non-IID sets, for example Prototypical Contrastive Federated Learning (FedProc) extend the concept further using class-prototypes to

further correct deviations in the local models [134].

5.4.8 Federated Bayesian Optimization

Gradient free Federated Learning algorithms are less common than their slope-focused counter parts. Nonetheless, a few groups have investigated non-gradient based techniques for global optimization as well as alternative aggregation methods.

In some circumstances, when the function call is expensive or first-order approximations cannot be performed, zeroth-order methods have seen great success. For example, tuning the hyperparameters of deep neural networks requires retraining the entire network [135]. Bayesian Optimization generates a probabilistic surrogate model which can be optimized over to minimize expensive function calls. The authors of [136] present Federated Thompson Sampling (FTS), an algorithm that extends Gaussian Process (GP) based BO to a federated environment. As Gaussian Processes are non-parametric methods, they require building a covariance matrix over the entire data set. The nature of Federated Learning makes this impossible. The authors instead propose sharing the feature vector of a parameterized GP approximation built from Random Fourier Features (RFF) [137]. They designed a Thompson Sampling [138] scheme which reduces the communication overhead of the system. Their work suggests a robustness against statistical heterogeneity, but provides no solution to system heterogeneity aside from leaving dropped clients out of the sampling. While other works suggests dropping straggling devices as well, [139], this can have a negative affect on convergence accuracy and may introduce bias against the data collected by slower clients. Achituve et al. [140] extended the work on Federated Bayesian Optimization to included personalized results. Their client specific models are shown to operate well

even when devices have small data sets with strongly different distributions.

5.5 Limitations and Alternative Methods

As discussed in Section 5.2.2, the core assumptions that differentiate Federated Learning from other forms of Machine Learning, also introduce the core challenges to the field. Without the common assumption of i.i.d. data in distributed learning, federated environments often contend with datasets exhibiting statistical disparities and local variations. These distributional discrepancies pull local updates toward local optima, which can decrease global model's rate of convergence. Moreover, the problem of non-identical distributions is exacerbated when slow clients contain critical information. These devices may exhibit system heterogeneity because they lack computational capability, network connectivity, or network reliability. Some clients may experience delays in transmitting updates or may intermittently disconnect from the network. Synchronous methods, including those employed in Federated Learning, encounter difficulties in such environments due to their reliance on immediate and consistent updates, which may prove inadequate in scenarios where clients experience delays or disconnections.

Recently, a handful of groups [141–145] have proposed an alternative method of aggregation, ensemble distillation for model fusion. Ensemble methods combine the prediction of weak heterogeneous classifiers to produce a more accurate model [146–152]. Rather than train the global model with updates from the clients, ensemble methods train full models on each of the clients. The server merges the local models to create a mixture model. The advantage of distributed ensembles is their flexibility. Each local model can be entirely client specific. Ensemble distillation has show great success in handling statistical heterogeneity. Most existing attempts

still rely on a synchronized communication flow, so straggler mitigation remains an active field of research. Furthermore, if the local models are poorly designed, the model may memorize a client's and dataset therefore leak sensitive information.

An alternative method to training Machine Learning models on a decentralized network is Transfer Learning, which involves pretraining a model on a large, diverse dataset and fine-tuning it on smaller, device-specific datasets [145, 153, 154]. Unlike Federated Learning, which requires training from scratch on each device, Transfer Learning leverages knowledge transfer from the pretraining phase to expedite convergence and improve performance on device-specific tasks. This approach is particularly effective when there is significant overlap in the underlying data distributions across devices. However, Transfer Learning may not be well-suited for scenarios where data distributions vary widely among devices, as it relies on the assumption of transferability between source and target domains.

Chapter 6

Case Study: When Privacy Techniques Do Not Provide Privacy

One key point to remember in the field of data privacy is that it is essential for a participating client to provide enough information for a model to be trained, but it is impossible to learn trends from purely private data. It is then crucial to understand what information is being transmitted from the client to the server, where it could be the target of attack. In this chapter, we provide two examples where the naive implementation of federated learning algorithms result in a direct breach of the participating client's privacy. The first example is a trivial case where the gradient information directly encodes a participating client's data which directly contradicts one of the main tenets of federated learning, that client data cannot transparently leave the client's device. The second example highlights that even for much more complex systems, such as federated reservoirs, the transmitted model update may contain significantly more information about the client than anticipated.

6.1 Regarding Privacy of Model Updates

One of the key tenets that defines a federated environment for machine learning is that a participating client's sensitive data does not transparently leave their device. Instead, the machine learning model is distributed to each participant where a local update is calculated and subsequently transmitted to the coordinating server. While the raw data does not directly leave the device, recent work has shown that gradient information is not as private as initially supposed.

In [155], the authors present an approach which obtains private training data from the gradient step of a federated deep neural network update. These results were then extended in [156] where the authors improved the stability of an adversaries convergence and extracted ground-truth labels for any differentiable model trained with cross-entry loss over one-hot labels. Similarly, the authors in [157] extract user labels from shared gradient information.

As gradient leakage attacks have garnered more attention in recent years, Wei et al. [158], developed a framework for evaluating gradient leakage attacks. Even as some methods appear promising, increasing capabilities of adversaries increasingly puts pressure on developed solutions. For example, some early empirical results suggested that drop-out during training was sufficient to protect clients from gradient leakage attack however it has been shown in [159] that drop-out is not enough.

6.2 A Trivial Example: Federated Support Vector Machines

In this section, we examine a common machine learning model, its usage in federated learning, and how the gradient encodes client information.

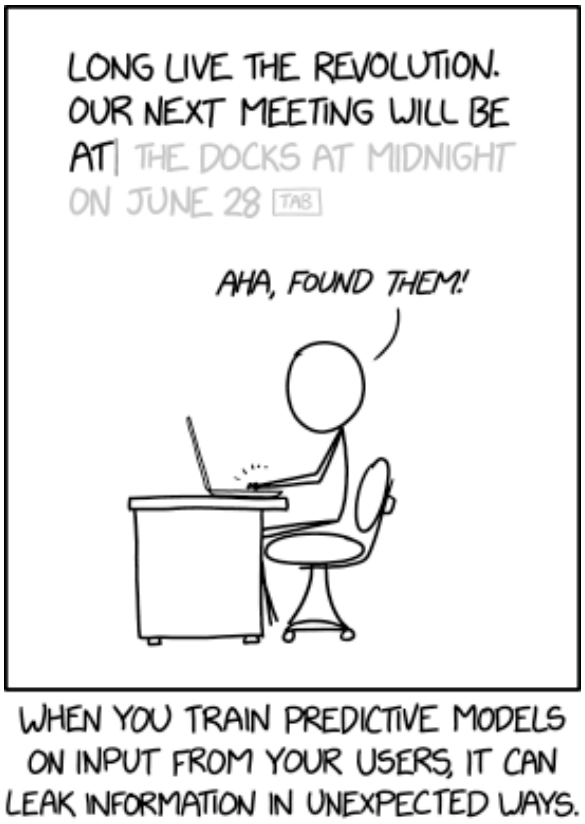

Figure 6.1: An xkcd web-comic by cartoonist Randall Munroe which depicts a situation where client information may be memorized during the training of a complex model and used to determine otherwise private information.

6.2.1 Support Vector Machines

Support Vector Machines (SVMs) are a type of supervised machine learning algorithm used for classification and regression tasks. The primary objective of SVM is to find the optimal hyperplane that best separates the different classes in the feature space while maximizing the distance between the hyperplane and the nearest data points from each class. Figure 6.2 depicts a simple example of a linearly separating hyperplane between two classes of data.

SVMs are particularly effective in high-dimensional spaces and can handle both linear and non-linear classification tasks through the use of kernel functions, which implicitly map the input data into a higher-dimensional space. SVMs are known for their ability to generalize well and for their robust performance in various real-world applications, including text classification,

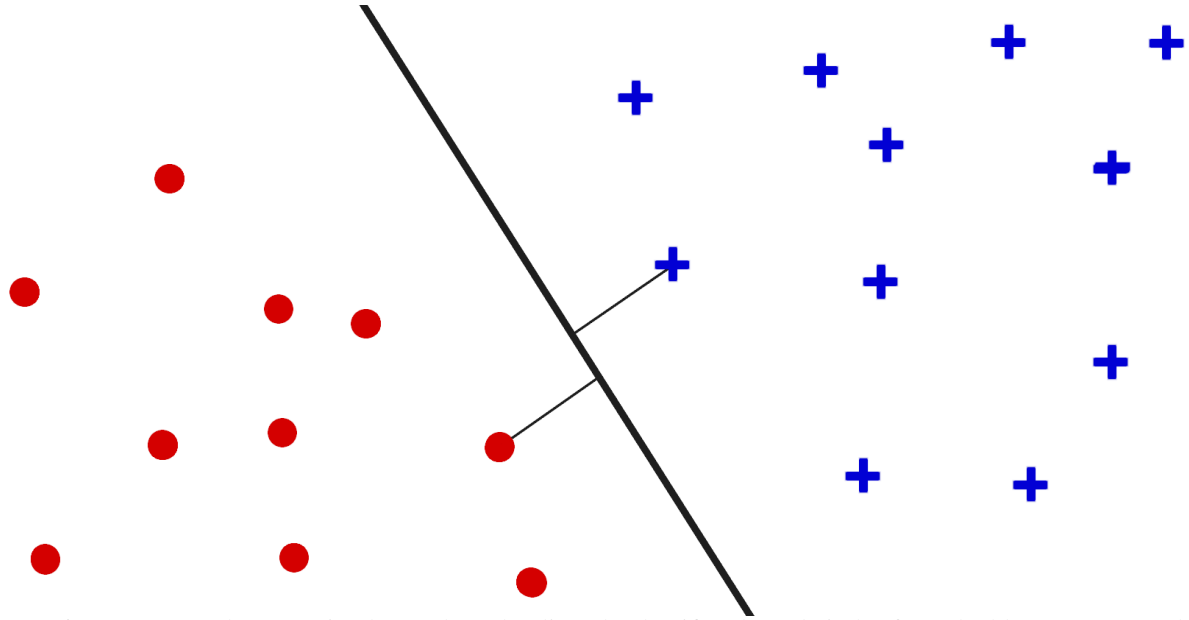

Figure 6.2: An example separating hyperplane that linearly classifies the red circles from the blue crosses. Such a hyperplane is frequently the model generated by Support Vector Machines. Through feature expansion, the model can capture nonlinear decision boundaries, greatly increasing the types of problems that SVMs can solve.

image recognition, and bioinformatics. Due to their well studied proprieties and easy to understand model, SVMs are excellent candidates for machine learning models on sensitive data where understandably enhances trust in the results. As such, many groups have extended SVMs to federated environments [160–163]. However, while the model is easy to understand, we will proceed to show that the gradient step for training Support Vector Machines directly encodes client data, violating one of the core tenets of federated learning.

6.2.2 Gradient step for training SVMs

Most Federated Learning algorithms are, in differing ways, improvements on the original Federated Averaging algorithm [18, 107]. These are learning methods that exchange gradient information, rather than client data, with the server in place of having large centralized databases. One of the tenets of Federated Learning is that raw client data does not leave the client's device. However, while ostensibly the sharing of gradient or other information does not equate to shar-

ing client data, it may nonetheless allow a sophisticated observer to determine sensitive client data. For example, consider training a support vector machine (SVM) [164] as a linear classifier, $(\mathcal{X}, \mathcal{Y}) = (\mathbb{R}^2, \{0, 1\})$. For simplicity, let the model be a one-dimensional hyperplane

$$f(\mathbf{x}, \mathbf{w}) = w \cdot \mathbf{x} + b, \quad \mathbf{w} = [w\ b]. \tag{6.1}$$

A common loss function for such a scheme is the hinge loss [165]:

$$J_k(\mathbf{w}) = \sum_{i=1}^{n_k} \max\left(0, 1 - y_i f(\mathbf{x}_i, \mathbf{w})\right),\ (\mathbf{x}_i, y_i) \in (\mathcal{X}, \mathcal{Y})_k. \tag{6.2}$$

This loss can be trained using gradient descent by taking the derivative of the loss with respect to the parameter vector $\mathbf{w}$. The individual gradients for the i^{th} data point are given by:

$$\frac{dJ_k}{dw} = \begin{cases} -y_i \mathbf{x}_i & y_i f(\mathbf{x}_i, \mathbf{w}) < 1 \\ 0 & y_i f(\mathbf{x}_i, \mathbf{w}) \geq 1 \end{cases}, \tag{6.3}$$

and

$$\frac{dJ_k}{db} = \begin{cases} y_i & y_i f(\mathbf{x}_i, \mathbf{w}) < 1 \\ 0 & y_i f(\mathbf{x}_i, \mathbf{w}) \geq 1 \end{cases}. \tag{6.4}$$

When the gradient is nonzero, the classification and data are compromised. This violates the assumption that client data does not leave the device.

Support vector machines are not the only machine learning technique to suffer from data leaks through the transmission of gradients. For example, neural networks have been known to encode the training dataset within back-propagation updates, allowing for client data to be

reconstructed [155–157,166]. Thus, training the model on the client devices alone is not sufficient to prevent data from being leaked.

6.3 A More Intricate Example: Federated Echo State Networks

Reservoir Computing, with its ability to efficiently process temporal data, has found applications across a wide variety of domains. Prominent applications of Reservoir Computing are in time series prediction and forecasting [167, 168]. RC models have been successfully employed to predict stock market prices [169, 170], weather patterns [171–173], and other time-varying phenomena [174–182]. Echo State Networks are a sub-field of Reservoir Computing that consist of a randomly generated reservoir of recurrently connected nodes [183, 184]. Due to their fixed initialization, ESNs tend to be more computationally efficient as the reservoir weights are not updated during training. Additionally, their randomly initialized and unchanging connections provide inherent stability to the network, making them robust to variations in the input data and noise [185]. While recent years have witnessed a tremendous amount of success with transformer based architectures, some groups have extended the Recurrent Neural Networks (RNN) used by ESNs with transformers [186, 187]. Moreover, it has been shown in [188] that transformer networks can be viewed as RNNs, suggesting that the results of this chapter could be extended to those models as well.

In this section, we examine Incremental Federated Learning, a method proposed in [189] for training Echo State Networks within a federated framework. We show that the method does not ensure client privacy. We do so by providing an algorithm that allows an adversarial agent to reconstruct a client's input data. We present simulated results in Incremental Federated Learning

scenario where a client's data is successfully reconstructed by an adversary using the exhibited algorithm. We additionally present an enhancement to Incremental Federated Learning that can be implemented to maintain the privacy of client data and bring the power of Reservoir Computing to federated environments.

6.3.1 Echo State Networks

Echo State Networks (ESN) are a special case of Reservoir Computing that use a recurrent neural network as the reservoir [183, 184]. Echo State Networks consist of a *fixed* reservoir of recurrently connected nodes. The connections between nodes are represented by a weight matrix $\mathbf{W} \in \mathbb{R}^{n_x \times n_x}$ where n_x is the number of nodes in the reservoir. The weight matrix $\mathbf{W}$ remains fixed throughout training. The reservoir's connections are typically sparse. The reservoir's dynamics can exhibit a rich set of nonlinear behaviors [190]. In order to ensure that the input signal propagates throughout the reservoir for a sufficiently long time interval, the spectral radius $\rho(\mathbf{W})$ of the network is generally required to be less than one, $\rho(\mathbf{W}) < 1$ [191].

To train an Echo State Network, the input values $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_T\}$ with $\mathbf{u}_t \in \mathbb{R}^{n_u}$ are sequentially passed into the reservoir. Because ESNs are designed to identify patterns in temporal data, the input values $\{\mathbf{u}\}$ are assumed to be correlated. The input matrix $\mathbf{W}_{in} \in \mathbb{R}^{n_x \times n_u}$ specifies the weight of the input value at each node. As $\mathbf{u}_t$ is fed into the reservoir, denote the state of all nodes at time t by the vector

$$\mathbf{x}_t = \begin{bmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{n_x,t} \end{bmatrix}. \tag{6.5}$$

The state of the reservoir at time t updates via an activation function:

$$\mathbf{x}_{t+1} = g(\mathbf{W}\mathbf{x}_t + \mathbf{W}_{in}\mathbf{u}_t), \tag{6.6}$$

The activation function $g(\cdot)$ is applied *element-wise* to the reservoir's state. The initial state $\mathbf{x}_1$ is generally set to 0. As the network evolves, the states are recorded in the follow matrix $\mathbf{X}$ as in [189]:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_T \\ \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_{T+1} \end{bmatrix}. \tag{6.7}$$

The objective of an Echo State Network is to learn a linear mapping between the recorded states and an output value:

$$\mathbf{y}_t = \mathbf{W}_{out} \begin{bmatrix} 1 \\ \mathbf{u}_t \\ \mathbf{x}_{t+1} \end{bmatrix}, \tag{6.8}$$

where $\mathbf{y}_t \in \mathbb{R}^{n_y}$ and $\mathbf{W}_{out} \in \mathbb{R}^{n_y \times (n_x + n_u + 1)}$. The set of output values $\{\mathbf{y}\}$ can be tabulated in the matrix $\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \cdots\ \mathbf{y}_T]$ and used to solve for $\mathbf{W}_{out}$. Commonly, this is done using Ridge regression [192],

$$\mathbf{W}_{out} = (\mathbf{Y}\mathbf{X}^T)(\mathbf{X}\mathbf{X}^T + \beta\mathbf{I})^{-1}, \tag{6.9}$$

where $\mathbf{I}$ is the identity matrix and $\beta \geq 0$ is a regularization term. Because the only layer trained is the final layer which produces the output, Echo State Networks are generally much quicker to train than other types of neural networks.

6.3.2 On the privacy of Federated Echo State Networks

The goal of a federated Echo State Network is to collaboratively estimate the output matrix weights, $\mathbf{W}_{out}$, without sharing the clients' raw local datasets $\mathcal{D}_k$. One method, proposed by Bacciu et al. [189], suggests specific intermediate matrices, recalled below, to share with the server. By intelligently selecting these matrices, the resulting $\mathbf{W}_{out}$ matrix is demonstrated to be identical to one generated by centralized data. Reference [189] says that the complexity of the reservoir is surely enough to obfuscate the input data. However, in this section we show that, in actuality, an adversary can reconstruct a client's input data.

To ensure that each client generates compatible matrices, the participants maintain local copies of the reservoir weights $\mathbf{W}$ and $\mathbf{W}_{in}$. Let $\mathbf{X}_k$ and $\mathbf{Y}_k$ be the matrices generated by passing the k^{th} client's data through its local copy of the reservoir. The federated method proposed in [189] has client k calculate the following intermediate matrices:

$$\mathbf{A}_k = \mathbf{Y}_k \mathbf{X}_k^T, \tag{6.10}$$

and

$$\mathbf{B}_k = \mathbf{X}_k \mathbf{X}_k^T + \beta \mathbf{I}, \tag{6.11}$$

where $\mathbf{I}$ is the identity matrix and $\beta \geq 0$ is a regularization term. These matrices are derived from the calculation of $\mathbf{W}_{out}$ using ridge regression (6.9). In a federated framework, the global output matrix $\mathbf{W}_{out}$ is calculated on the server using an aggregation of the clients' $\mathbf{A}_k$ and $\mathbf{B}_k$ matrices,

$$\mathbf{W}_{out} = (\sum_{k=1}^{K} \mathbf{A}_k)(\sum_{k=1}^{K} \mathbf{B}_k)^{-1}. \tag{6.12}$$

The authors in [189] show that Equations (6.9) and (6.12) are equivalent under reasonable conditions.

To simplify the calculations, we take the input to be one dimensional, $u_t \in \mathbb{R}$, and we consider only the calculations done by client k, dropping the subscript. Let the objective of an adversarial agent targeting client k be to determine the client's input sequence $\{u_1, u_2, ..., u_T\}$.

We take the following quantities to be public knowledge as they are known to all clients, including any clandestine participating adversaries: the number of clients K, the reservoir weights $\mathbf{W}$ and $\mathbf{W}_{in}$, and the regularization term β. We additionally assume that the adversary is able to intercept the intermediate matrices, $(\mathbf{A}, \mathbf{B})$, from the k^{th} client. With this information, we detail below how participant k's input data, $\{u_1, ..., u_T\}$, can be leaked.

Implementing the same activation function used in [189],

$$g(\mathbf{x}_t, u_t) = (1-a)\mathbf{x}_t + a\tanh(\mathbf{W}\mathbf{x}_t + \mathbf{W}_{in}u_t), \tag{6.13}$$

the transferred matrices given in (6.10) and (6.11) can be expanded as

$$\mathbf{A} = \sum_{t=2}^{T} \begin{bmatrix} u_t & u_t u_{t-1} & u_t x_{1,t} & \cdots & u_t x_{n,t} \end{bmatrix}, \tag{6.14}$$

and

$$\mathbf{B} = \sum_{t=2}^{T} \begin{bmatrix} 1 & u_{t-1} & x_{1,t} & \cdots & x_{n,t} \\ u_{t-1} & u_{t-1}^2 & u_{t-1}x_{1,t} & \cdots & u_{t-1}x_{n,t} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n,t} & u_{t-1}x_{n,t} & x_{n,t}x_{1,t} & \cdots & x_{n,t}^2 \end{bmatrix} + \beta\mathbf{I}. \tag{6.15}$$

Denote the adversarial agent's estimated input by $\hat{\mathbf{u}}$, which is passed through the local reservoir to generate the estimated intermediate matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$. We use mean squared error (MSE) to generate a loss from the squared sum of unique elements in (6.14) and (6.15),

$$\begin{aligned} L = &(\hat{A}_{1,2} - A_{1,2})^2 + (\hat{A}_{1,3} - A_{1,3})^2 + ... \\ &+ (\hat{B}_{n+2,n+2} - B_{n+2,n+2})^2. \end{aligned} \tag{6.16}$$

The gradient of the loss with respect to each element is given by

$$\frac{dL}{dA_{i,j}} = 2(\hat{A}_{i,j} - A_{i,j}), \quad \frac{dL}{dB_{i,j}} = 2(\hat{B}_{i,j} - B_{i,j}). \tag{6.17}$$

The gradients $\frac{dA_{i,j}}{du_t}$ and $\frac{dB_{i,j}}{du_t}$ are calculated using the chain rule. To wit, we calculate the partial derivative of each unique element of $\mathbf{A}$ and $\mathbf{B}$ with respect to the individual input values $u_t,\ t \in \{1, 2, ..., T\}$. The partial derivative of the i^{th} state at time $t+1$ with respect to the value of the j^{th} state at the preceding time t is given by

$$\frac{\partial x_{i,t+1}}{\partial x_{j,t}} = [(1-a) + a\mathbf{W}\text{sech}^2(\mathbf{W}\mathbf{x}_t + \mathbf{W}_{in}u_t)]_j, \tag{6.18}$$

and the partial derivative with respect to u_t is

$$\frac{\partial x_{i,t+1}}{\partial u_t} = a\mathbf{W}_{in}\text{sech}^2(\mathbf{W}\mathbf{x}_t + \mathbf{W}_{in}u_t). \tag{6.19}$$

To update the estimated input $\hat{\mathbf{u}}$, we use gradient descent with a learning rate $\alpha = 10^{-8}$ and

momentum coefficient $\gamma = 0.1$,

$$\hat{\mathbf{u}}^{(k+1)} = \hat{\mathbf{u}}^{(k)} - \alpha \frac{\partial L}{\partial \mathbf{u}} - \gamma(\hat{\mathbf{u}}^{(k)} - \hat{\mathbf{u}}^{(k-1)}), \tag{6.20}$$

where $\hat{\mathbf{u}}^{(k)}$ represents the estimated value after k gradient steps.

Below we demonstrate that algorithm (6.20) allows an adversary to learn client k's input data. In addition, an adversary can enhance convergence of (6.20) by using an advanced method such as AMSGrad [193].

6.3.3 Simulation results

To illustrate that the suggested method can accurately determine client input, we conduct two simulations mirroring an adversary's calculations in a scenario with a modest reservoir and non-trivial dataset. We utilize a reservoir of size $n = 100$ that is 80% sparse with the nonzero weights of $\mathbf{W}$ and $\mathbf{W}_{in}$ generated by sampling a Gaussian distribution with zero mean and unit variance. The spectral radius of the reservoir is set to 0.9 to ensure the echo state property [191]. As in [189], set $a = 0.3$ in the activation function, Eq. (6.13). The true client input $\mathbf{u} = \{u_1, u_2, ..., u_T\}$ is generated from

$$u_t = 5\sin(\frac{t^2}{60}), \quad t \in \{1, 2, ..., T = 20\}, \tag{6.21}$$

and is illustrated in Figure 6.3. The adversary's input estimate $\hat{\mathbf{u}}$ is initialized to 0. Figure 6.4 shows the evolution of the adversary's estimate under the gradient descent algorithm (6.20) for 10,000 iterations.

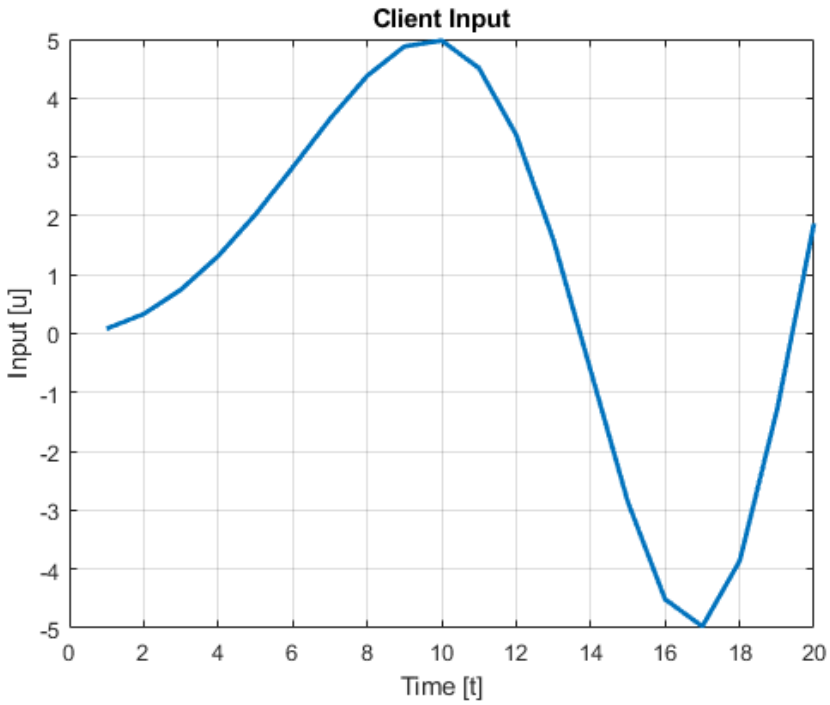

Figure 6.3: To train an Echo State Network in a Federated Learning environment, each client holds sensitive time series. This data is used to generate the intermediate matrices **A** and **B**. In our simulation, this data is generated from Equation (6.21).

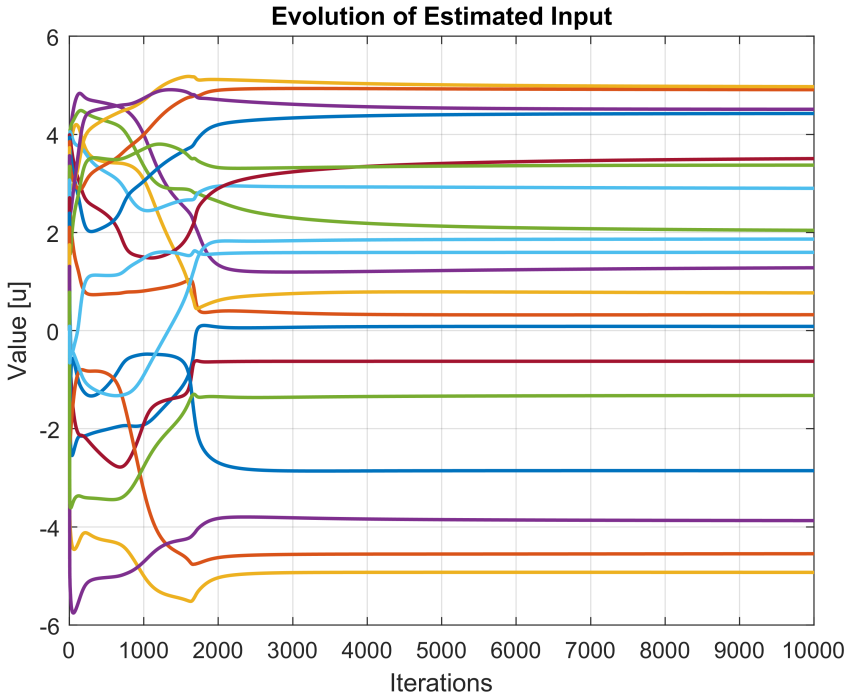

Figure 6.4: Even through the update step for Federated Echo State Networks is complex, an adversary can train the model in reverse using gradient descent to obtain the clients sensitive information. Each trajectory in the figure represents the value of one input element $\hat{u}_t$ as the adversary's estimate evolves during training.

The error between the true client data and the estimated data at each gradient step is shown in Figure 6.5; note that the y-axis is on a logarithmic scale.

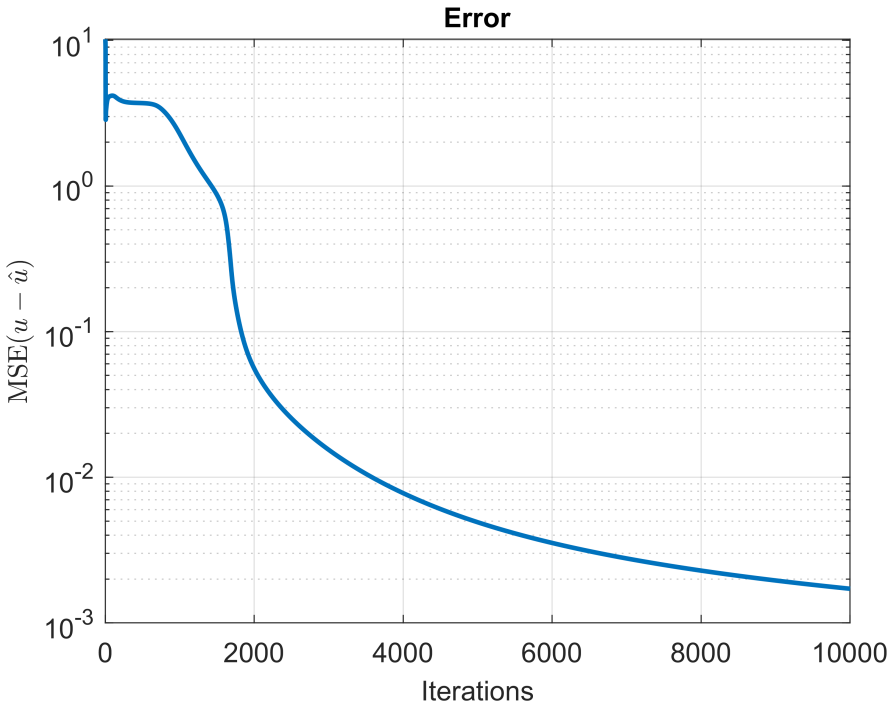

Figure 6.5: During the training of the adversary's network, the mean squared error between the client's vector input $\mathbf{u}$ and the adversary's estimate of that input, $\hat{\mathbf{u}}$ decreases.

An average of the adversary's final estimated input $\hat{\mathbf{u}}$, after 10 trials, is shown alongside the true input in Figure 6.6. Each error bar is one standard deviation of the trials. The true input values (solid **blue**) and the estimated input values (dashed **orange**) are nearly equivalent. This illustrates that the adversary is able to correctly identify client $k's$ information.

6.3.4 Enhanced privacy through Differential Privacy

To ensure participants' privacy, further steps are required to prevent data from leaking during the update step. One possible approach to protect client data during transmission is Federated Inference Learning (FedInfer) [94], a method of injecting structured noise before the communication step to obfuscate personal data.

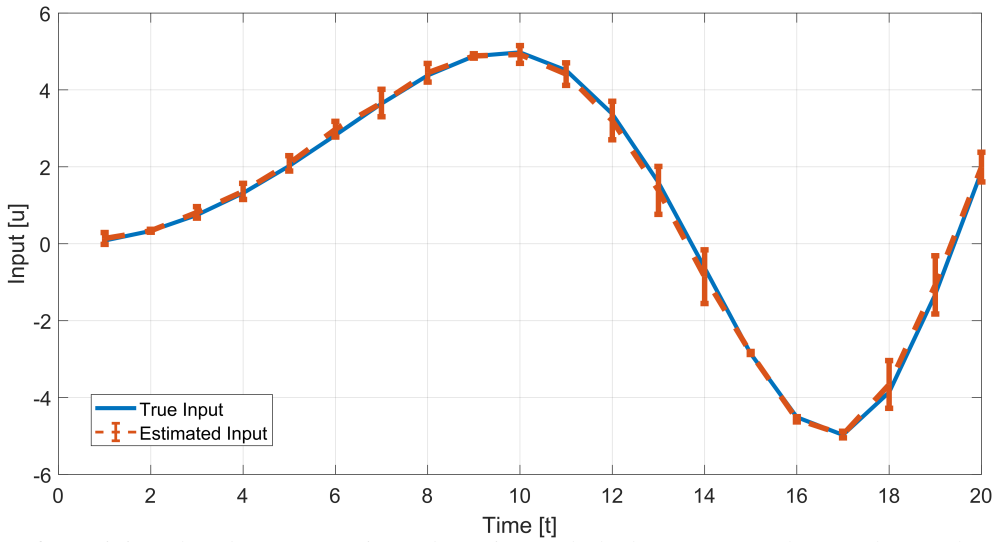

Figure 6.6: After training, the adversary's estimated user input (dashed **orange**) matches nearly exactly with the true client input (**blue**). The results summarize results of 10 trials with randomized initial conditions for the adversary's network. The provided error bars are one standard deviation.

Federated Inference uses an iterative procedure to estimate the aggregate of client data. To apply FedInfer to the algorithm from [189] recalled in Section 6.3.2, noise is injected into the intermediate matrices (6.10) and (6.11) before transmission to the server. Implementing FedInfer with Gaussian noise with a signal-to-noise-ratio (SNR) of 1:4 into our simulation prevents an adversary from determining the client's data, as illustrated in Figure 6.7. These results are obtained after the same number of iterations as in the prior test.

6.4 Closing Remarks

Federated learning still remains an effective method for preventing mass breaches in data privacy through the compromise of central servers. However, it is crucial to recognize the limited reach that simply transmitting gradients rather than data directly can have. In simple cases, such as with the Support Vector Machine, the gradient information directly encodes the client's personal data. Moreover, even when the method increases in complexity, such as with Federated

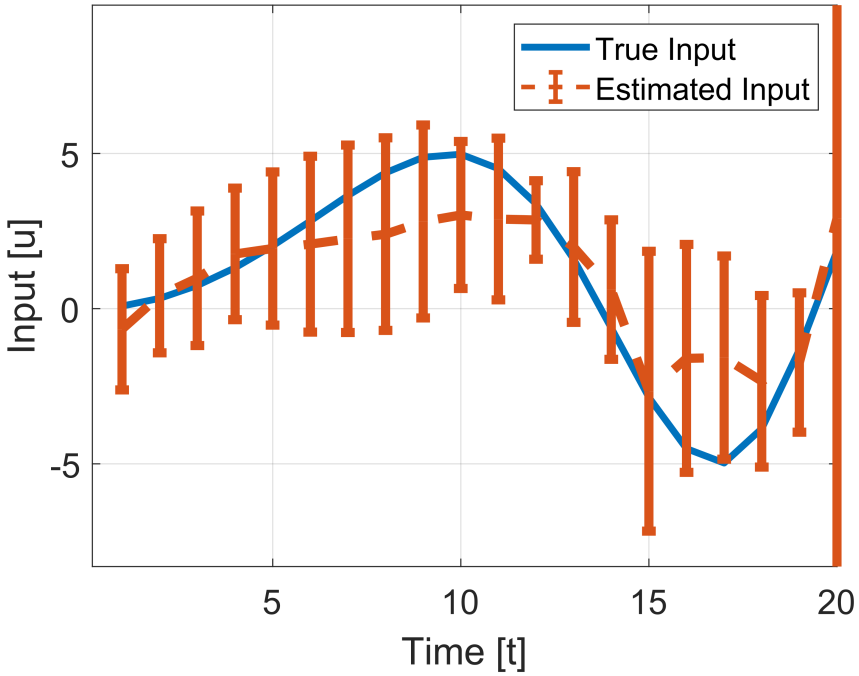

Figure 6.7: After noise has been injected into the client's intermediate matrices **A** and **B**, the adversarial agent's network is unable to converge on the client's input data. The true input (solid **blue**) is the data **u** the client used to generate the matrices **A** and **B**. The estimated input (dashed **orange**) results from the calculations described in Section 6.3.2. Because the data is Differentially Private, the adversarial agent's ability to determine the client's personal data is severely diminished. The error bars display one standard deviation after 10 independent trials.

Echo State Networks, the data transferred from the client to the server still encodes the entirety of the participant's data. It is with this backdrop that the next chapter introduces a novel federated learning method that takes advantage of statistical methods to circumvent the challenges of the field while using differential privacy techniques to quantify how much privacy is lost.

Chapter 7

Federated Inference

In this chapter, we present a novel federated leaning algorithm which we call Federated Inference (FedInfer). Based on Bayesian inference, FedInfer addresses the challenges of heterogeneity in federated learning environments. We first recall the main assumptions of a federated environment and the resulting challenges induced by those assumptions. Then, we discuss how existing solutions are unable to fully mitigate the negative impact these challenges have on model convergence. The shortfall of existing methods motivate the development of Federated Inference, which efficiently solves certain classes of federated learning problems while protecting client privacy.

7.1 Motivation

We recall from Chapter 5 the three main assumptions that define a federated learning environment:

- Client data cannot be transparently transmitted to the server.
- Client devices are not standardized.

- The number of clients may be significantly larger than the size of each individual dataset.

The first assumption relates to the level of privacy federated learning algorithms seek to provide participating clients. By preventing the server from directly accessing client data, federated learning protects clients from large data breaches where a single compromised server affects tens of thousands to hundreds of thousands of individuals. However, as explored by Chapter 6, simply passing gradient information in the place of raw data does not ensure adversarial agents or servers cannot retrieve sensitive client information.

The second assumption relates to types of client devices that participate in the training and the reliability of each devices network connectivity. This problem is frequently known as *straggler mitigation*. Synchronized algorithms such as FedAvg [18] and SCAFFOLD [112], make no attempt to reconcile the different rate of update speeds each client is capable of achieving. Because of this, the learning process must proceed at the speed of the slowest client. FedProx [121], sought to mitigate the effect of straggling clients by rephrasing the gradient step in terms of local optimizations which could be halted early to return synchronized updates with varying levels of accuracy. While this prevents single clients from bogging down the learning process, by halting the optimization early, FedProx introduces and speed vs. accuracy trade-off. Rather than reformulate the optimization, FedLin [132] allows each client to perform a variable number of updates relative to the local speed of each client. This allows for linear convergence guarantees on ideal[1] optimizations. However, the mechanisms by which FedLin allows faster clients to update more frequently is the reduction of the local gradient step. Thus, while a fast client may be able to take 10 steps, each step is $1/10^{th}$ the size.

[1]By ideal, we refer to constructed problems where the cost function is specified directly, frequently as quadratic, as opposed to derived by an optimization over datapoints.

Straggling devices would not pose such a problem if each client maintained a local data set with nearly identical distributions to each other dataset. However, the last assumption relates to the differences in client's local datasets. Because the local datasets are assumed to be small when compared to the number of clients, it is frequently assumed that each has a unique local optima. These local optima limit the usefulness of local gradient steps by clients as the update steps diverge from the global model. Both SCAFFOLD and FedLin introduce extra terms to the gradient update step in order to mitigate the difference in local optima by influencing the local steps to remain close to the last global update. Combined with system heterogeneity, statistical heterogeneity frequently prevents federated algorithms from converging to the true optima.

To address these challenges we develop the method of Federated Inference. By utilizing a Bayesian Inference step rather than a gradient step, the latter two challenges are naturally mitigated because updates are no longer tied to specific points on the cost surface. To ensure client privacy, FedInfer naturally incorporates differential privacy through injected noise, ensuring that client privacy can be quantified. In the next section, we review and define notation for Bayesian Inference, and essential step in the Federated Inference algorithm.

7.2 Bayesian Inference

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space comprised of a sample space Ω, a σ-algebra $\mathcal{F}$ of events on Ω, and a probability measure $\mathbb{P}$. Consider a random variable Z that operates on $(\Omega, \mathcal{F}, \mathbb{P})$ and takes values in some measurable space (E, Σ_E) where E is the set of possible values and Σ_E is a σ-algebra of events [194, p. 39]. Denote a finite sequence of observations by $\{Z\}_n := \{Z_1, Z_2, \ldots, Z_n\}$. The random variable comes equipped with a distribution function

$\mathbb{P}^Z(A) := \mathbb{P}(Z \in A)$ defined on the $\sigma-$algebra Σ_E. In a statistical experiment, we assume *a priori* knowledge of a parametric family of distributions containing $\mathbb{P}^Z \in \{\mathbb{P}_\theta, \theta \in \Theta\}$, where θ is parameter taking values in the space Θ. The objective of a statistical experiment is to use observations of Z to determine the parameter θ^* that yields the true distribution $\mathbb{P}^Z$; we seek θ^* such that $\mathbb{P}^Z = \mathbb{P}_{\theta^*}$. Its natural to assume that for different values θ_1 and θ_2, the distributions $\mathbb{P}_{\theta_1}$ and $\mathbb{P}_{\theta_2}$ are identifiable.

We require the existence of a probability density function. This can be guaranteed using the Radon-Nikodyn Theorem under the assumption of a dominating measure [194, p. 53].

Theorem 7.2.1. *(Radon-Nikodym [194]) Denote by $\{\mathbb{P}_\theta\}$ a family of distributions parameterized by a vector $\theta \in \Theta$, defined on a measurable space (E, Σ_E). Let μ be a measure defined on the measurable space (E, Σ_E) that dominates a family of distributions $\{\mathbb{P}_\theta\}$. Then, there exists a unique density function $p(z;\theta)$ relative to the measure μ for which*

$$\mathbb{P}_\theta(A) = \int_A p(z;\theta) d\mu, \quad \forall A \in \Sigma_E. \tag{7.1}$$

◀

The density $p(z;\theta)$ is unique up to a μ-null set. Equation 7.1 is often written as

$$p(z;\theta) = \frac{d\mathbb{P}_\theta}{d\mu}, \tag{7.2}$$

which is referred to as known as the Radon-Nikodym derivative.

Next, we define a statistic which can be used to estimate the distribution's parameter vector θ.

Definition 24. *(Statistic [195, p. 12]:) A statistic T is any measurable function that maps values of the random variable to a measurable space:*

$$T : (E_1, \Sigma_{E_1}) \rightarrow (E_2, \Sigma_{E_2}). \tag{7.3}$$

Let $\{Z\}_n$ be a set of n samples from the density $p(z; \theta)$. A statistic T is called sufficient for the family $\{\mathbb{P}_\theta\}$ if knowledge of $T(\{Z\}_n)$ provides all the information about θ that would be obtainable by instead knowing the whole set of samples $\{Z\}_n$ [196, p. 246].

◀

A commonly used statistic is the sample mean. Often, we would like to integrate prior knowledge about a system into a statistic. A common method to achieve this is known as Bayesian Inference [194] which we now detail below.

Let $\{Z\}_n$ be a set of n samples independently drawn from a distribution $\mathbb{P}_{\theta^*} \in \{\mathbb{P}_\theta\}$, where θ^* denotes the true parameter vector. Let the true value θ^* be unknown and the objective to determine the true value from the set of observation $\{Z\}_n$.

First, assume we have some prior knowledge about the parameter vector θ that we encode as a density over the space Θ. This density is known as the *prior* density; we denote it by $\pi(\theta)$. We assume that π has a density and that the density is dominated by μ. This prior describes our belief that a given θ parameter vector is the true value θ^*. In the case where there is no prior knowledge, we can take what is called a *flat* prior, which means that we ascribe an equal probability to all elements of the set. When the space of parameter values Θ is continuous, this creates an *improper* prior as the function is not a true probability density [194].

The conditional distribution $p(\{Z\}_n|\theta)$ describes the likelihood of observing the set of values $\{Z\}_n$ from the density p for a given parameter value θ. In Bayesian statistics, this is not viewed as a probability density as the observations, $\{Z\}_n$ are known and θ is unknown. Because of this, it is common to introduce the likelihood function $\ell_p(\theta|\{Z\}_n) := p(\{Z\}_n|\theta)$ to clarify that the independent variable is θ.

Using $\pi(\theta)$ and $p(\{Z\}_n|\theta)$, Bayes' Theorem provides a method for calculating the posterior density, which we denote by $q(\theta|\{Z\}_n)$ [195]. The posterior density describes the conditional probability of the parameter vector θ given a set of samples $\{Z\}_n$. Employing Bayes' Theorem to update the prior density using new samples is known as Bayesian Inference.

Theorem 7.2.2. *(Bayesian Inference [195, p. 23]) Let $\pi(\theta)$ and $p(z|\theta)$ be a prior and conditional density respectively. Then, the posterior density is given by*

$$q(\theta|\{Z\}_n) := \frac{p(\{Z\}_n|\theta)\pi(\theta)}{\int_\Theta p(\{Z\}_n|\tau)\pi(\tau)d\tau}. \tag{7.4}$$

◀

The denominator in equation (7.4) is the distribution obtained by marginalizing over the possible parameter values. In practice this value is difficult to estimate *a priori*. However, its purpose is to normalize the distribution and is often estimated indirectly by requiring the integral of the posterior distribution to be 1. Because of this, the inference step is commonly reduced to a proportionality relation given by:

$$q(\theta|\{Z\}_n) \propto \ell_p(\theta|\{Z\}_n)\pi(\theta). \tag{7.5}$$

The convergence properties of Bayesian estimators have been considered since the first results for martingales by Doob (1949) in [197]. The results were later applied under more general assumptions by Le Cam (1986) in [198]. The specific regularity conditions presented here are from DasGupta (2008) in [199]. For finite dimensional problems with suitably chosen distributions, the effect of the prior falls off as the number of samples increases. The Bernstein-von Mises theorem is a reformulation of the central limit theorem (CLT) and states that, in the limit, the posterior approaches a multivariate normal distribution with a mean of the maximum likelihood estimator (MLE). DaGupta's formulation and the required regularity conditions are recalled next.

Theorem 7.2.3. *(Bernstein-von Mises Theorem [200, p. 339]) Let p and π be the two absolutely continuous densities with respect to a measure μ as in (7.1). Denote a set of n independent samples drawn from the density p by $\{Z\}_n \sim p$. Let the parameter θ be follow the density π with support on Θ, $\theta \sim \pi(\theta)$. Assume the following regularity conditions [199]:*

- *Θ is open.*
- *The likelihood of the parameter θ satisfies:*

$$\int \frac{\partial^2}{\partial \theta^2} \ell_p(\theta | \{Z\}_n) d\mu = 0, \quad \forall \theta \in \Theta.$$

- *The Fisher information function is positive and bounded:*

$$0 < \mathcal{I}(\theta) = -\mathbb{E}\big[\frac{\partial^2}{\partial \theta^2} \log p(\{Z\}_n | \theta)\big] < \infty.$$

- *The prior density has a positive value at the true value θ^*, $\pi(\theta^*) > 0$, and is continuous in the set $\{\theta \mid |\theta - \theta^*| \leq \epsilon, \epsilon > 0\}$ for some $\epsilon > 0$.*

- *There exists a function* $k(\{Z\}_n)$*, satisfying* $\mathbb{E}_{\theta^*}[k(\{Z\}_n)] < \infty$*, such that the second derivative of the likelihood is bounded by* $k(\{Z\}_n)$ $\forall\theta \in \{\theta \mid |\theta - \theta^*| \leq \epsilon, \epsilon > 0\}$*:*

$$\left|\frac{\partial^2}{\partial\theta^2}\ell_p(\theta|\{Z\}_n)\right| \leq k(\{Z\}_n).$$

Let $\{\hat{\theta}\}_n$ *be any almost surely consistent sequence of zeros [201, Sec. 10.4] of the derivative of the likelihood function after* n *observations, otherwise known as the maximum likelihood estimate (MLE), and denote* $\psi_n = \sqrt{n}(\theta - \hat{\theta}_n)$*. Then the posterior of* ψ_n*,* $q(\psi_n|\{Z\}_n)$ *satisfies*

$$\begin{aligned}&\left|q(\psi_n|\{Z\}_n) - \sqrt{I(\theta^*)}\phi(\psi\cdot\sqrt{I(\theta^*)})\right| \xrightarrow{a.s.} 0\\ &\text{with } \psi_n := \sqrt{n}(\theta - \hat{\theta}_n)\end{aligned} \tag{7.6}$$

and where ϕ *is the probability density function of a standard normal* $\mathcal{N}(0,1)$*. The posterior distribution of the* ψ_n *almost surely converges to a normal distribution with a variance of the inverse Fisher information function evaluated at the maximal likelihood estimate. Also* $\hat{\theta}_n$ *converges to the true value* θ^* *in the limit.*

Proof. See Bickel and Doksum [200, p. 339] and similar constructions in chapter 7.4.2 of [202], chapter 10.2 of [201], or chapter 1.4 of [203]. □

The objective, in this work, is to construct conditions within the federated environment so that a statistic of the posterior can be used to estimate the true parameter of the distribution $\mathbb{P}_{\theta^*}$. We show that, under suitable conditions, the expected value of the posterior distribution converges to the true parameters almost surely. For a more detailed derivation of Bayesian statistics, we refer the reader to [195].

7.3 Federated Inference

In this work, we propose a novel federated learning algorithm, Federated Inference (FedInfer). FedInfer is a method of machine learning for federated environments that uses Bayesian Inference and intentionally noisy observations to learn a global model while obfuscating the clients' local data. Many of the challenges of federated learning stem from the gradient based approach used in the original Federated Averaging [18]. We note that gradient based methods perform well when client updates have minimal latency or share a common data distribution. FedInfer, proposed here, has broader applicability achieved through using Bayesian Inference.

7.3.1 Assumptions

The first assumption is that the loss function is separable. This will be sufficient to ensure convergence of FedInfer to an optimal parameter $\mathbf{w}^*$.

Assumption 3. *(Separable loss function) The loss function J_k is separable, i.e. there exist functions g operating on the $\mathcal{W}$ and h operating on datasets $\mathcal{D}$ such that the loss function J_k is expressible as the product*

$$J_k(\boldsymbol{w}) = g(\boldsymbol{w})h(\mathcal{D}_k), \tag{7.7}$$

for all $\boldsymbol{w} \in \mathcal{W}$ and $k \in \{1, 2, \ldots, K\}$. Here, g and h are vector functions of a dimension that allows (7.7) to hold. That is, $g(\boldsymbol{w})$ returns a row vector and $h(\mathcal{D}_k)$ returns a column vector such that the resulting cost is scalar.

◀

Remark 4. *While at first glance this assumption may appear restrictive, consider that the Taylor Series can be used to approximate an analytic function by a polynomial in its arguments. For example, the function* $\tanh(wx)$ *where* w *is the parameter and* x *is a data point is not separable. However, we can approximate the function by the polynomial*

$$\tanh(wx) = wx - \frac{(wx)^3}{3} + \frac{2(wx)^5}{15} - \frac{17(wx)^7}{315} + O(x^9). \tag{7.8}$$

Hence the $\tanh(wx)$ *can be approximated as:*

$$\tanh(wx) \approx \begin{bmatrix} w & \frac{-w^3}{3} & \frac{2w^5}{15} & -\frac{17w^7}{315} \end{bmatrix} \begin{bmatrix} x \\ x^3 \\ x^5 \\ x^7 \end{bmatrix}. \tag{7.9}$$

This approximation allows us to express the $\tanh(xw)$ *function as a linear combination of the basis functions* $[x\ x^3\ x^5\ x^7]$*. By utilizing Taylor Series expansion, we can capture (at least locally) the behavior of nonlinear functions in terms of a finite number of basis functions, thereby reducing the limitation imposed by Assumption 3.* ◀

Our second blanket assumption is the existence of a sufficient streaming statistic.

Assumption 4. *(Sufficient Streaming Statistic) Denote by* $\{Z\}_n$ *a set of* n *independent samples of the distribution* $\mathbb{P}^Z = \mathbb{P}_{\theta^*}$*. Let* T *be a sufficient statistic (Eq. 7.3) for the parameter* θ^**. We denote the value of the statistic applied to the* n *samples by* $t_n = T(\{Z\}_n)$ *and assume that there*

exists a function T' that updates the current statistic with the newest observation:

$$\begin{aligned} t_{n+1} &= T(\{Z\}_{n+1}) \\ &= T'(t_n, Z_{n+1}). \end{aligned} \tag{7.10}$$

◀

Note that either or both of T and T' may depend on n. Assumption 4 allows the server to maintain an estimate of the posterior without requiring the storage of all previous transmissions

We now proceed to set up some useful notation to simplify the description of Federated Inference and its subsequent analysis.

7.3.2 Client data blocks

Using the function h in Assumption 3, denote each client's transformed data to be shared with the server by

$$\mathbf{z}_k := \frac{|\mathcal{D}_k|}{|\mathcal{D}|} h(\mathcal{D}_k). \tag{7.11}$$

Note, this differs from the sharing of the gradient in traditional Federated Learning in that the value of $\mathbf{w}$ does not enter into what is shared with the server. We call $\mathbf{z}_k$ the k^{th} client's data

block. Then, using (5.6) and (7.7), the global minimization problem can be formulated as

$$\begin{aligned}\arg\min_{\mathbf{w}} J(\mathbf{w}) &= \arg\min_{\mathbf{w}} \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} J_k(\mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} g(\mathbf{w}) h(\mathcal{D}_k) \\ &= \arg\min_{\mathbf{w}} \; g(\mathbf{w}) \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} h(\mathcal{D}_k) \\ &= \arg\min_{\mathbf{w}} \; g(\mathbf{w}) \sum_{k=1}^{K} \mathbf{z}_k.\end{aligned} \tag{7.12}$$

Let $\mathbf{z}^*$ be the sum of the clients' transformed data, which we call the global data block:

$$\mathbf{z}^* := \sum_{k=1}^{K} \mathbf{z}_k. \tag{7.13}$$

Then, the optimal global model parameter $\mathbf{w}^*$ is determined by the following minimization:

$$\mathbf{w}^* := \arg\min_{\mathbf{w}} \; g(\mathbf{w}) \mathbf{z}^*. \tag{7.14}$$

Assumption 3 ensures that the value reached by the server is globally optimal. Because the objective function is known, the server has access to the functions g and h. To solve (7.14), the server requires each client to provide its local data block $\mathbf{z}_k$. However, this data may contain sensitive information, so directly sending $\mathbf{z}_k$ to the server may breach a client's privacy. FedInfer protects the client's privacy by injecting noise into the communicated value. The server then uses Bayesian Inference to update a posterior estimate of the global data block.

To more intuitively understand the FedInfer algorithm, we first consider the case of synchronous communication between the clients and server and then extend the notation and results

to case of asynchronous communication.

7.3.3 Synchronous communication

Take the communication between the clients and the server to be synchronous. In this case, all clients simultaneously return observations of their local data blocks. Because the federated framework does not permit transmission to the server of raw client data, FedInfer injects noise into the client update to obfuscate client information. Denote by R a zero-mean random variable on the probability space $(E, \Sigma_E, \mathbb{P}^R)$. Let each client communicate to the server an observation of its local $\mathbf{z}_k$, perturbed by a noise vector with elements drawn independently from R. The j^{th} observation of the k^{th} client's communicated block is then:

$$\hat{\mathbf{z}}_{k,j} = \mathbf{z}_k + \mathbf{r}_{k,j}, \tag{7.15}$$

where $\mathbf{r}_{k,j}$ is the j^{th} vector of samples from R. Denote by S_K the random variable obtains as the sum of K independent random variables R_k all having distribution R:

$$S_K = \sum_{k=1}^{K} R_k. \tag{7.16}$$

Since expectation is linear, the mean of S_K is zero. We introduce a parameterized family of distributions

$$S_{K,\theta} := \theta + S_K, \tag{7.17}$$

where $\theta \in \Theta$ is the expected value of members of the family.

In the case of synchronous communication, the server receives all client observations si-

multaneously. By equations (7.13) and (7.15), the sum of client responses form an observation (the n^{th} observation in this case) of the global data parameter,

$$\begin{aligned} \hat{\mathbf{z}}_n^* &= \hat{\mathbf{z}}_{1,n} + ... + \hat{\mathbf{z}}_{K,n}, \\ &= \mathbf{z}^* + \sum_{k=1}^{K} \mathbf{r}_{k,n}. \end{aligned} \tag{7.18}$$

Thus, a uniformly weighted sum of samples from all clients has the distribution of a random variable $\hat{\mathbf{z}}_n^*$ with mean $\mathbf{z}^*$, the global data parameter:

$$\hat{\mathbf{z}}_n^* \sim S_{K,\mathbf{z}^*} \quad \text{(Sync)}. \tag{7.19}$$

Federated Inference uses observations of the form in (7.19) with a predefined prior over $\mathbf{z}$ to estimate the true global data block $\mathbf{z}^*$. Next, we extend this notation to handle system heterogeneity, i.e. when the server receives the clients' updates asynchronously.

7.3.4 Asynchronous communication

In the case of asynchronous communication, the process from the client's perspective is identical to that for the synchronous case. That being, each client returns an observation of their local data block by equation (7.15). However, due to computational or network constraints, client updates may arrive at the server at different rates. Thus, the server must account for the differing arrival times. Without loss of generality, let the clients be ordered according to increasing latency of communication; we make the simplifying assumption that the relative rates are fixed and finite. Let $\{\gamma_1, \gamma_2, ..., \gamma_K\}$ denote client transmission rates relative to client 1, the fastest client. By

construction, $\gamma_1 = 1$ and the value γ_k indicates that the k^{th} client returns one update for every γ_k updates from client 1.

Let Γ denote the least common multiple of $\{\gamma_2, \gamma_3, \ldots, \gamma_K\}$. Then, after Γ transmissions of the fastest client, the k^{th} client will have sent Γ/γ_k samples to the server. We weight client k's updates by γ_k/Γ so that the sum of observations, after these Γ/γ_k transmissions, retains a mean of $\mathbf{z}_k$. Consider the first Γ/γ_k samples transmitted to the server by client k:

$$\frac{\gamma_k}{\Gamma}\hat{\mathbf{z}}_{k,1} + \ldots + \frac{\gamma_k}{\Gamma}\hat{\mathbf{z}}_{k,\Gamma/\gamma_k} \sim \frac{\gamma_k}{\Gamma} S_{\Gamma/\gamma_k, \mathbf{z}_k \Gamma/\gamma_k}. \tag{7.20}$$

Observe that the sum of local observations in (7.20) is drawn from the distribution $S_{\Gamma/\gamma_k, \mathbf{z}_k \Gamma/\gamma_k}$, which has the following expectation:

$$\mathbb{E}\Big[\frac{\gamma_k}{\Gamma}\hat{\mathbf{z}}_{k,1} + \ldots + \frac{\gamma_k}{\Gamma}\hat{\mathbf{z}}_{k,\Gamma/\gamma_k}\Big] = \mathbb{E}\Big[\sum_{j=1}^{\Gamma/\gamma_k} \frac{\gamma_k}{\Gamma}\hat{\mathbf{z}}_{k,j}\Big]. \tag{7.21}$$

Bringing the weights outside the sum on the right side of (7.21) gives the equivalent expression

$$\mathbb{E}\Big[\sum_{j=1}^{\Gamma/\gamma_k} \frac{\gamma_k}{\Gamma}\hat{\mathbf{z}}_{k,j}\Big] = \mathbb{E}\Big[\frac{\gamma_k}{\Gamma}\sum_{j=1}^{\Gamma/\gamma_k} \hat{\mathbf{z}}_{k,j}\Big]. \tag{7.22}$$

Substituting (7.11) into (7.22) gives

$$\begin{aligned} \frac{\gamma_k}{\Gamma}\mathbb{E}\Big[\sum_{j=1}^{\Gamma/\gamma_k} \mathbf{z}_k + r_{k,i}\Big] &= \frac{\gamma_k}{\Gamma}\frac{\Gamma}{\gamma_k}\mathbf{z}_k + \mathbb{E}\big[S_{\Gamma/\gamma_k}\big] \\ &= \mathbf{z}_k. \end{aligned} \tag{7.23}$$

We call each block of consecutive Γ/γ_k samples of client k's local data block a 'full'

observation of local block $\mathbf{z}_k$. We can then denote the n^{th} observation of the global data block by $\hat{\mathbf{z}}_n^*$ which is constructed as the sum of 'full' observations of each client:

$$\begin{aligned}\hat{\mathbf{z}}_n^* = &\sum_{j=n\Gamma/\gamma_1}^{(n+1)\Gamma/\gamma_1} \frac{\gamma_1}{\Gamma}\hat{\mathbf{z}}_{1,j} + \sum_{j=n\Gamma/\gamma_2}^{(n+1)\Gamma/\gamma_2} \frac{\gamma_2}{\Gamma}\hat{\mathbf{z}}_{2,j} + \ldots \\ \ldots + &\sum_{j=n\Gamma/\gamma_K}^{(n+1)\Gamma/\gamma_K} \frac{\gamma_K}{\Gamma}\hat{\mathbf{z}}_{K,j}.\end{aligned} \tag{7.24}$$

As a result, each 'full' observation of $\hat{\mathbf{z}}^*$ is drawn from the following distribution:

$$\hat{\mathbf{z}}_n^* \sim \sum_{k=1}^{K} \frac{\gamma_k}{\Gamma} S_{\Gamma/\gamma_k, \mathbf{z}_k\Gamma/\gamma_k} \quad \text{(Async)}, \tag{7.25}$$

which, by linearity of the expectation, has an expected value $\mathbf{z}^*$.

7.3.5 The FedInfer algorithm

Federated Inference is comprised of a client algorithm and a server algorithm. The clients invoke Algorithm 1 below, which gives the initialization of their data blocks $\mathbf{z}_k$ by (7.11) and the method for generating observations of said data blocks (7.15). Line 4, "**loop** While training", indicates that different implementations may utilize different stopping conditions for determining the end of training. Borrowing from Python notation, the *yield* statement on line 7 indicates that the loop continues after returning the latest sample to the server.

Algorithm 2 below applies to both the synchronous and asynchronous cases. It gives the complete set of steps the server takes to calculate the estimate $\mathbf{w}_i^*$, where i denotes the server's current loop count. The count i is incremented each time the server receives an update from a client. To accommodate asynchronous communication, we introduce an intermediate estimate

Algorithm 1 Federated Inference: Part A - k^{th} Client

1: **function** FEDINFER-CLIENT($R, h, \mathcal{D}_k, |\mathcal{D}|$)
2: $\quad \mathbf{z}_k \leftarrow \frac{|\mathcal{D}_k|}{|\mathcal{D}|} h(\mathcal{D}_k)$
3: $\quad j \leftarrow 1$
4: $\quad$ **loop** While training
5: $\quad\quad r_{k,j} \sim R$
6: $\quad\quad \hat{\mathbf{z}}_{k,j} \leftarrow \mathbf{z}_k + r_{k,j}$
7: $\quad\quad$ **yield** $\hat{\mathbf{z}}_{k,j}$
8: $\quad\quad j \leftarrow j + 1$
9: $\quad$ **end loop**
10: **end function**

ξ_i which interpolates between the latest streaming statistic comprised of 'full' global samples, denoted by $\bar{\mathbf{z}}_n^* := T(\{\hat{\mathbf{z}}^*\}_n)$, and the next 'full' global data block sample $\hat{\mathbf{z}}_{n+1}^*$. The interpolation is given by

$$\xi_{i+1} = \xi_i - \frac{|\mathcal{D}_k|\gamma_k}{|\mathcal{D}|\Gamma} \bar{\mathbf{z}}_n^* + \frac{\gamma_k}{\Gamma} \hat{\mathbf{z}}_{k,j}, \tag{7.26}$$

where $\hat{\mathbf{z}}_{k,j}$ is the j^{th} update from the k^{th} client (line 10 of Alg. 2). After the $(n+1)^{st}$ 'full' observation of the global data block is completed, ξ_i resets to the latest value of the statistic, $T(\{\hat{\mathbf{z}}^*\}_{n+1})$ (line 15 of Alg. 2).

After the estimate ξ_i and intermediate statistic value $\mathbf{t}_i$ are calculated, the server combines these with the prior knowledge of $\mathbf{z}^*$ using a Bayesian Inference step (line 7.4). In Algorithm 2, the notation is simplified by only including the posterior q. We then use the expected value of the posterior to calculate the server's i^{th} estimate of the model's parameter vector $\mathbf{w}_i^*$.

Recall that when the clients are synchronized, each iteration of client updates is returned simultaneously. In the synchronous case, each $\gamma_k = 1 \; \forall k \in \{1, ..., K\}$ and $\Gamma = 1$.

Algorithm 2 Federated Inference: Part B - Server

function FEDINFER-SERVER($T', q, \{|\mathcal{D}_k|\}_K, \{\gamma_k\}_K$)
  $n \leftarrow 0$
  $i \leftarrow 1$
  $\mathbf{t}_0 \leftarrow \mathbf{0}$
  Allow each client to return once:
    $\bar{\mathbf{z}}_0^* \leftarrow \frac{1}{K} \sum_{k=1}^K \hat{\mathbf{z}}_{k,1}$
  $\xi_0 \leftarrow \bar{\mathbf{z}}_0^*$
  loop While training
    if Client k returns with $\hat{\mathbf{z}}_{k,j}$ **then**
      $\xi_i \leftarrow \xi_{i-1} - \frac{|\mathcal{D}_k|\gamma_k}{|\mathcal{D}|\Gamma} \bar{\mathbf{z}}_n^* + \frac{\gamma_k}{\Gamma} \hat{\mathbf{z}}_{k,j}$
      $\mathbf{t}_i \leftarrow T'(\bar{\mathbf{z}}_n^*, \xi_i)$
      $\hat{\mathbf{w}}_i^* \leftarrow \arg\min_{\mathbf{w}} J(\mathbf{w}, \mathbb{E}[q(\mathbf{z}|\mathbf{t}_i)])$
      if $i \bmod \sum_{k=1}^K \frac{\Gamma}{\gamma_k}$ **is** 0 **then**
        $\bar{\mathbf{z}}_{n+1}^* \leftarrow \mathbf{t}_i$
        $\xi_i \leftarrow \bar{\mathbf{z}}_{n+1}^*$
        $n \leftarrow n + 1$
      end if
      yield $\mathbf{w}_i^*$
      $i \leftarrow i + 1$
    end if
  end loop
end function

7.3.6 Consistency of FedInfer

Next, we present a theorem on the consistency of FedInfer in the case of synchronous client updates; this algorithm consists of the combination of Algorithm 1 and Algorithm 2 in the synchronous updates case.

Remark 5. *In the presented analysis, we assume that the server is able to accommodate updates from the entire client set. While this is generally untrue, we note that the FedInfer algorithm is asynchronous and therefore can handle client updates as they arrive, rather than waiting to process the entire batch at once. This feature is not commonly seen in gradient based approaches which use batches to reduce the serves instantaneous demand. If the client base is too large for*

FedInfer to accommodate asynchronously, then randomized batching can be used. If the batches are chosen uniformly, the following analysis holds, but we leave the exact derivation for future work.

Theorem 7.3.1. *(Consistency of Federated Inference: Synchronous Updates) Let all clients return observations of their local data blocks synchronously, i.e.* $\gamma_k = 1$ *for all* $k \in \{1, ..., K\}$. *Take the distribution* R*, used by the clients to perturb their local data blocks* $\mathbf{z}_k$*, to be such that a convolution of* K *densities of similar shape, equation (7.16), satisfies the conditions of Theorem 7.2.3. Denote by* $\bar{\mathbf{z}}_n^*$ *a sufficient statistic for* $\mathbf{z}^*$ *constructed by applying* T *to the set of* n *observations* $\{\hat{\mathbf{z}}_1^*, ..., \hat{\mathbf{z}}_n^*\}$. *Let* ψ_n *be constructed as in Thm. 7.2.3, with* $\psi_n = \sqrt{n}(\mathbf{z} - \bar{\mathbf{z}}_n^*)$. *Let there exist a random variable* Y *with* $\mathbb{E}[Y] < \infty$ *such that* $\mathbb{E}[|q(\psi_n|\{\hat{\mathbf{z}}^*\}_n)|] < \mathbb{E}[Y]$ *for all* $n > 0$. *Then, the algorithm presented in this work, Federated Inference, when the client updates are synchronized, is strongly consistent.*

Proof. Consider the sequence of outputs generated by $\bar{\mathbf{z}}_n^*$ for $n > 1$. By construction $\bar{\mathbf{z}}_n^*$ is the output of the sufficient statistic T and so contains the same information about $\mathbf{z}^*$ as the entire sequence of observations $\{\hat{\mathbf{z}}^*\}_n$. Thus, the likelihood function given the set of observations $\{\hat{\mathbf{z}}^*\}_n$ and the likelihood function given the sufficient statistic $\bar{\mathbf{z}}_n^*$ are equivalent:

$$\ell(\mathbf{z}|\{\hat{\mathbf{z}}^*\}_n) = \ell(\mathbf{z}|\bar{\mathbf{z}}_n^*). \tag{7.27}$$

Thus, the posterior distribution can be equivalently written in terms of the sufficient statistic,

$$q(\mathbf{z}|\{\hat{\mathbf{z}}^*\}_n) = q(\mathbf{z}|\bar{\mathbf{z}}_n^*) \tag{7.28}$$

Next, we consider the right side of (7.28) in the limit as n tends to infinity. The additional assumption $\mathbb{E}[Y] < \infty$ bounding the expectation of the noise distribution allows the application Lebesgue's dominated convergence theorem (DCT) [194]. The DCT allows bringing the limit operation into the expectation:

$$\lim_{n\to\infty} \mathbb{E}[q(\psi|\bar{\mathbf{z}}_n^*)] = \mathbb{E}[\lim_{n\to\infty} q(\psi|\bar{\mathbf{z}}_n^*)] \tag{7.29}$$

By Theorem 7.2.3, the limiting density on the right side of (7.29) is a normal density with mean $\mathbf{z}^*$

$$\begin{aligned} \mathbb{E}[\lim_{n\to\infty} q(\psi|\bar{\mathbf{z}}_n^*)] &= \mathbb{E}][\sqrt{I(\mathbf{z}^*)}\phi(\psi \cdot \sqrt{I(\mathbf{z}^*)})] \\ &= \mathbf{z}^*. \end{aligned} \tag{7.30}$$

Thus, in the limit, the expected value of the posterior converges to the true value $\mathbf{z}^*$. □

Remark 6. *Questions arise as to the validity for real world data of the assumption that the noise satisfies the conditions of Theorem 7.2.3. However, with Federated Inference, the data is intentionally injected with a selected distribution for the sake of privacy. The distribution and its properties are thus tunable and can be chosen to satisfy the assumptions of Theorem 7.2.3.*

◀

Next, we extend the results to the case of asynchronous communication.

Theorem 7.3.2. *(Consistent Subsequence of Federated Inference: Asynchronous Updates) Let the clients return updates to the server asynchronously, i.e., there exists at least one $k \in \{1, ..., K\}$ such that $\gamma_k > 1$. Take the distribution R, used by the clients to perturb their local data blocks $\boldsymbol{z}_k$, to be such that convolution of weighted distributions of similar shape (Eq. (7.25)) satis-*

fies the conditions of Theorem 7.2.3. Denote by $\bar{z}_n^*$ a sufficient statistic for z^* constructed by applying T to the set of n observations $\{\hat{z}_1^*, ..., \hat{z}_n^*\}$. Let ψ_n be constructed as in Thm. 7.2.3, with $\psi_n = \sqrt{n}(z - \bar{z}_n^*)$. Let there exist a random variable Y with $\mathbb{E}[Y] < \infty$ such that $\mathbb{E}[|q(\psi_n|\{\hat{z}^*\}_n)|] < \mathbb{E}[Y]$ for all $n > 0$. Then, the algorithm presented in this work, Federated Inference, yields a strongly consistent subsequence.

Proof. Observe that the subsequence consisting of the $\bar{\mathbf{z}}_n^*$ for $n > 1$ is a sequence of i.i.d. samples of a distribution with mean $\mathbf{z}^*$ that satisfies Theorem 7.2.3. □

7.4 Rate of Convergence

Comparing the rate of convergence between Bayesian Inference and gradient techniques It can be shown [200, p. 343] that under suitable conditions Bayesian estimators are asymptotically efficient, achieving the lower bound dictated by the Information Inequality (Cramér-Rao bound) [204–206], i.e., that the variance of an unbiased estimator is lower bounded by the inverse of the Fisher information [207]. For most well behaved distributions, this is $O(n^{-1/2})$. This rate does highlight one of the limitation of this method, namely that modern gradient descent based approaches with constant step size can achieve linear convergence rates, $O(n^{-1})$, on similarly structure problems [208, 209]. However, as will be shown in the next chapter, linear convergence is often demonstrated in idealistic settings. When noise is injected into the update step per Differential Privacy, the learning rate must be reduced to stabilize the convergence. Additionally, the gradient diminishes as the model approaches the optimum. This can cause the injected noise to overwhelm the gradient step and prevent further convergence. This effect can be mitigated by progressively decreasing the step size at the cost of linear convergence.

7.5 Concluding Remarks

Here we discuss the key advantages offered by FedInfer at a conceptual level. In the next chapter, these advantages will be illustrated using case studies.

Federated Inference Learning represents a novel approach within the realm of federated learning, distinct from conventional gradient-based methods. Rather than relying on local gradient steps to update a global model, FedInfer employs Bayesian Inference update steps. This departure from the gradient descent paradigm introduces several notable advantages, rendering FedInfer a promising alternative for distributed machine learning tasks.

One key advantage of FedInfer lies in its inherent compatibility with differential privacy, a critical aspect of modern data protection. Through the initial step of noise injection, FedInfer is naturally differentially private. By implementing FedInfer, the sensitive information held by participating clients is obfuscated, and differential privacy methods can be used to obtain a measure of how much privacy is achieved. This integration of privacy-preserving techniques at the core of FedInfer underscores its suitability for applications demanding stringent data security measures.

Furthermore, FedInfer distinguishes itself by guaranteeing convergence to the true optimum. Unlike gradient-based approaches, which may encounter convergence challenges from system and statistical heterogeneity or stall due to noise interference, FedInfer assures convergence even in the presence of noise. This assurance stems from the method's independence from gradient information derived from local update steps. By leveraging Bayesian Inference, FedInfer takes advantage of the statistical proprieties of the noise-induced fluctuations to converge reliably towards the optimal solution. This reliability in convergence is instrumental in enhancing the efficiency and effectiveness of federated learning processes.

The divergence of FedInfer from gradient-based methodologies offers tangible benefits across various dimensions. By embracing Bayesian Inference and prioritizing differential privacy, FedInfer presents a holistic approach that addresses both performance and security concerns inherent in federated machine learning settings. Through its robust convergence guarantees and innate privacy safeguards, FedInfer is as a compelling framework for respectfully training machining leaning models on sensitive client data.

Chapter 8

Applications of Federated Inference

The assumptions that distinguish a federated environment from other Machine Learning environments introduce key challenges that hinder model training. Here we present simulations that demonstrate the core challenges to the training of a Machine Learning model utilizing Federated Learning algorithms, particular when compared with our suggested method Federated Inference.

8.1 Overview

Recall the three main assumptions of the federated environment: data privacy, statistical heterogeneity, and system heterogeneity.

Privacy is the first challenge of Federated Learning. The method aims to collaboratively train Machine Learning models while protecting the identity of the individuals that contribute data to the aggregated training set. As such, it is critical that the client data is not transparently transmitted between the client and the server. As analyzed in Chapter 6, simply choosing the gradient as the update step may not be sufficient in protecting client information from adversarial agents. To this end, we use Local Differential Privacy techniques to explore their effect on model convergence.

The second challenge, statistical heterogeneity, relates to the distribution of data across client devices. It is frequently assumed, especially when employing stochastic gradient decent, that the data is independent and identically distributed (i.i.d.). However, it is generally observed that because Federated Learning seeks to train models across large numbers of devices, each with a small local dataset, the assumption of i.i.d. data does not hold in practice. In the simulations to follow, we emulate statistical heterogeneity by adjusting the distribution of data held by each client.

The third challenge of a federated environment, system heterogeneity, is the lack of regularity between participating devices and the networks they utilize. Training Machine Learning models can be a computationally intensive task. Differences in the compute capabilities of client devices affect the length of time needed to calculate a model update. Moreover, variations in network reliability exacerbate the problem by slowing straggling devices and further delaying updates. Because the majority of federated algorithms are gradient based, updates to the global model are local to the model's current position in the cost space. This makes asynchronous methods particularly challenging to implement and result in a large field of synchronized algorithms, which each handle straggling devices through different update tricks.

To aid in the examination of these challenges, we employ case studies and perform associated simulations that demonstrate the impact that each challenge has on model training. In these simulations, we train a Machine Learning model in an increasingly difficult federated environment on three classification tasks. The first task is a simple synthetic dataset that is generated by sampling datapoints from two multidimensional Gaussians. The first set of results discuss the impact that Differential Privacy has on the model's accuracy throughout training. Next, we look at the Fisher Iris flower dataset which is composed of three classifications of iris flowers [210,211].

With this dataset, we allow each individual client to be an *experts* at a particular iris flower by holding the majority of datapoints for that classification. This skews the distributions maintained by each client and therefore induces different local objectives. In these simulations, we examine the impact that statistical heterogeneity has on this distributed training problem. Lastly, we investigate how modern methods mitigate the effect of straggling devices. On these final tests, we utilize a more realistic dataset comprised of breast cancer tissue samples [212]. This dataset is more representative of the types of problems that can benefit from Federated Learning algorithms.

On each of the following classification tasks, we compare the accuracy results of a selection of Federated Learning algorithms on a reserved set of test data. We opted to compare our method FedInfer against FedAvg [18], SCAFFOLD [112], FedLin [132], and FedInfer. FedAvg is from the seminal work of the field and is frequently used as a benchmark as it does not utilize measure to handle the challenges mentioned above. The SCAFFOLD algorithms introduces a control variate which is designed to address statistical heterogeneity by biasing local updates to more closely align with the global cost function. Then, we chose FedLin as it extended the work of SCAFFOLD with a more accurate update step and introduced local learning rates to address the final challenge of system heterogeneity. We next describe the specifics of the model we selected for these classification tasks.

Remark 7. *Each conclusion drawn from this section can be observed when training the model on any of the three datasets. The inclusion of each dataset serves to demonstrate that FedInfer is able to perform well even as the feature space increases in complexity.*

◀

8.2 Federated Support Vector Machines

To compare the training of a Machine Learning model by a Federated Learning algorithms, we choose to simulate the training of a Support Vector Machine (SVM) in three classification tasks. SVMs are powerful supervised learning algorithms used for classification and regression tasks. Introduced by Vladimir Vapnik and his colleagues in the 1990s, SVMs have since become widely popular due to their ability to handle high-dimensional data and effectively generalize to unseen examples. In this work, we follow after [213], which used a federated SVM to perform a binary classification of a medical condition.

At their core, SVMs work by finding the optimal hyperplane separating different classes of data points in a feature space. This hyperplane is optimized by maximizing the distance between the hyperplane and the nearest data points from each class. By maximizing the margin, SVMs are robustness to noise and uncertainty within the dataset.

We begin by assuming a linear model, which describes a linear relation between the variables, $\mathbf{x} \in \mathbb{R}^d$, and the parameters, $\mathbf{w} \in \mathbb{R}^p$:

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{x}^T \mathbf{w}_w + \mathbf{w}_b, \quad \mathbf{x} \in \mathcal{X}, \;\; \mathbf{w} \in \mathcal{W}. \tag{8.1}$$

Linear models are well understood, simple, and quick to implement. The parameters are split into a weight term, $\mathbf{w}_w$, the elements of which give the weights for the elements of the feature vector, and an additive bias term, $\mathbf{w}_b$. We follow a selection of other authors in the Federated Learning community and use the Mean Squared Error (MSE) as the local cost function for its

mathematical simplicity [115, 132]:

$$J_k(\mathbf{w}; \mathcal{D}_k) = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}_k} \frac{1}{2}||f(\mathbf{x}; \mathbf{w}) - \mathbf{y}||^2. \tag{8.2}$$

One trick we implement is to observe that the expression (8.2) is convex and its derivative with respect to $\mathbf{w}$ is affine:

$$\frac{\partial J_k}{\partial \mathbf{w}} = \frac{n_k}{n}(\mathbf{x}_k^T\mathbf{x}_k\mathbf{w} - \mathbf{x}_k^T\mathbf{y}_k). \tag{8.3}$$

The MSE satisfies the separability condition of Assumption 3:

$$\begin{aligned} \frac{\partial J_k}{\partial \mathbf{w}} &= \mathbf{z}_k \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix} \\ \text{with } \mathbf{z}_k &= \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \begin{bmatrix} \mathbf{x}_k^T\mathbf{x}_k & \mathbf{x}_k^T\mathbf{y}_k \end{bmatrix}. \end{aligned} \tag{8.4}$$

The derivative of the global objective is rewritten in the separated form:

$$\begin{aligned} \sum_{k=1}^{K} \frac{\partial J_k}{\partial \mathbf{w}} &= \sum_{k=1}^{K} \mathbf{z}_k \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix} \\ &= \left(\sum_{k=1}^{K} \mathbf{z}_k \right) \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix} \\ &= \mathbf{z}^* \begin{bmatrix} \mathbf{w} \\ 1 \end{bmatrix} \end{aligned} \tag{8.5}$$

Recognizing that the derivative is affine, the optimization is solved by identifying the zero cross-

ing,

$$\mathbf{z}^* \begin{bmatrix} \mathbf{w} \\ \\ 1 \end{bmatrix} = 0. \tag{8.6}$$

Thus we see that minimizing $J(\mathbf{w}; \mathcal{D})$ is tantamount to finding the null space of $\mathbf{z}^*$.

The choice of model in Equation 8.1 is quite restrictive. In cases where the data is not linearly separable, SVMs can still be effective by using the kernel trick. This technique allows SVMs to implicitly map the input data into a higher-dimensional space where it may become linearly separable. By finding the optimal separating hyperplane in this transformed space, SVMs can effectively classify non-linearly separable data. Some popular kernels include, the linear kernel,

$$K_{lin}(\mathbf{x}, \mathbf{y}; \sigma) = \sigma \mathbf{x}^T \mathbf{y}, \tag{8.7}$$

the polynomial kernel,

$$K_{poly}(\mathbf{x}, \mathbf{y}; c, d) = (\mathbf{x}^T \mathbf{y} + c)^d, \tag{8.8}$$

and the Radial Basis Function (RBF) kernel,

$$K_{RBF}(\mathbf{x}, \mathbf{y}; \gamma) = \exp(-\gamma ||\mathbf{x} - \mathbf{y}||^2). \tag{8.9}$$

In this work, we found that the addition of a scaled linear kernel with a constant kernel was sufficient for all datasets.

Remark 8. *Traditionally, Support Vector Machines use the Gram matrix to implement the kernel trick, which is derived by calculating the kernel from the dot product between all datapoints in the dataset [164]. In a federated environment, each client does not have access to the entire*

dataset. Therefore, every client would calculate a unique Gram matrices, each incompatible with the others. To circumvent this problem, we allow the server to select a set of basis-like vectors within the feature space that serve as a basis for the kernel calculation. For small feature spaces, selecting a grid of points is sufficient. However, as the dimension of the feature space increases, this approach quickly becomes infeasible. For the work presented here, it is sufficient to select vectors uniformly randomly within the feature space. However, an interesting avenue of future research may be partitioning the domain with a space filling curve to determine vectors that are uniformly distant from their nearest neighbor, without resorting to a full spacial grid.

◀

8.3 Client Privacy with a Synthetic Classification Dataset

Our analysis begins with a synthetic dataset constructed by sampling 1000 datapoints evenly from two Gaussian distributions. The SVM is trained to classify which distribution a given data point is from. Each datapoint consists of two feature values, $\mathbf{x} \in \mathbb{R}^2$ with the value of each point being from the set of classifiers $\{1, 2\}$. A depiction of this synthetic dataset is provided in Figure 8.1.

This classification problem mirrors the identification of medical ailments from a vector of health data. On this first dataset, we examine the first challenge: client privacy. For simplicity, we take two clients, with each client holding 50% of the data. Initially, we give the zero-privacy results in Figure 8.2. These results are obtained without Differential Privacy protecting the clients. Each of the gradient based methods reaches a near maximum after approximately 20 communication steps. By construction of the optimization problem, FedInfer solves for the optima of the

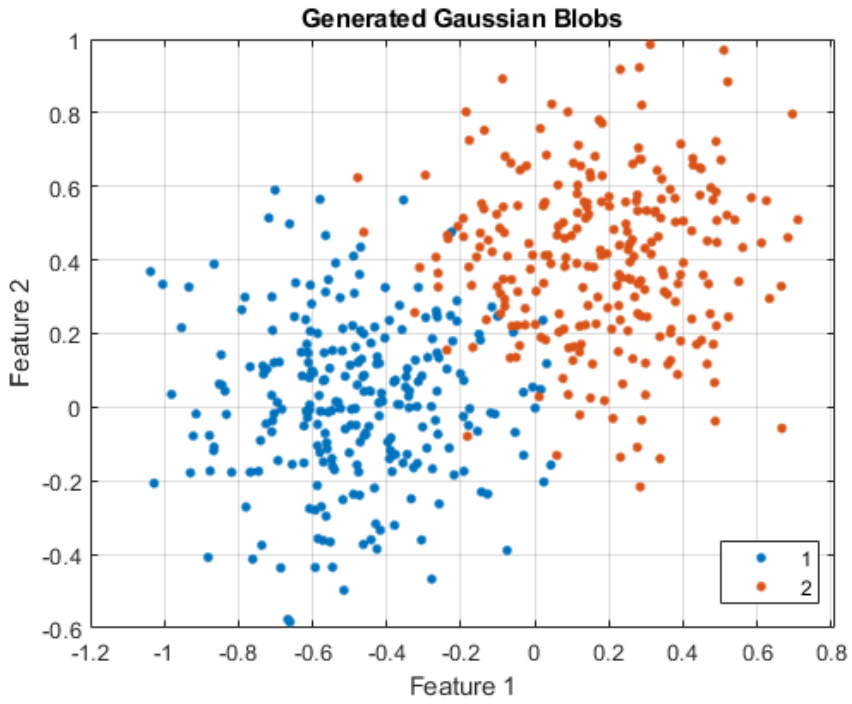

Figure 8.1: The synthetic classification dataset is generated by sampling from two two-dimensional Gaussian distributions. The distribution from which a point is drawn determines its classification. The objective of a Machine Learning model is to determine which distribution the datapoint was drawn from.

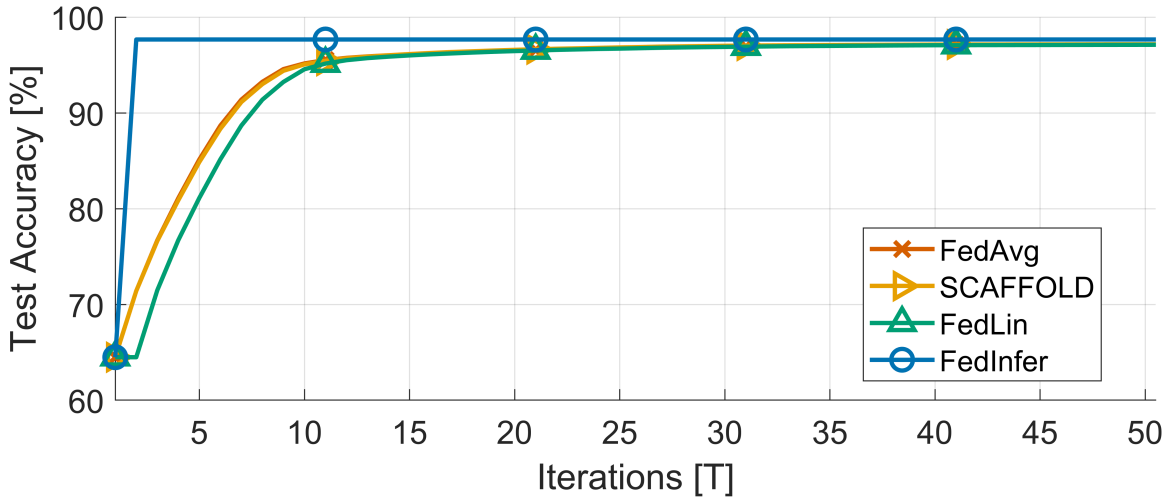

Figure 8.2: Without the presence of noise, statistical heterogeneity, or system heterogeneity, the synthetic dataset is quick to solve for all federated algorithms. Federated Averaging, shown with **red** $\times$ and SCAFFOLD, shown with **yellow** $\triangleright$, are aligned. FedLin, with **green** $\triangle$, is one step behind due to its initialization process. Without the presents of noise, Federated Inference, shown with **blue** $\circ$, is able to converge to the optimal solution in one step.

global cost function in a single step.

To implement Local Differential Privacy, the client updates are bound to within a finite range and noise is injected prior to transmission to the server. This is done by calculating the

ℓ_2-sensitivity of the matrix $\mathbf{z}^*$ and clipping the gradients. Gaussian noise is injected into the update step to protect the client's personal data. The objective of these tests is to observe how each algorithm behaves under federated conditions, we matched the privacy loss of each update but did not go so far as to calculate the total privacy loss after all iterations. Figure 8.3 depicts the averaged results of 500 trials in the high privacy regime, where an individual update has Differential Privacy parameters $(\varepsilon, \delta) = (0.16, 1e^{-4})$. From these results we observe that the

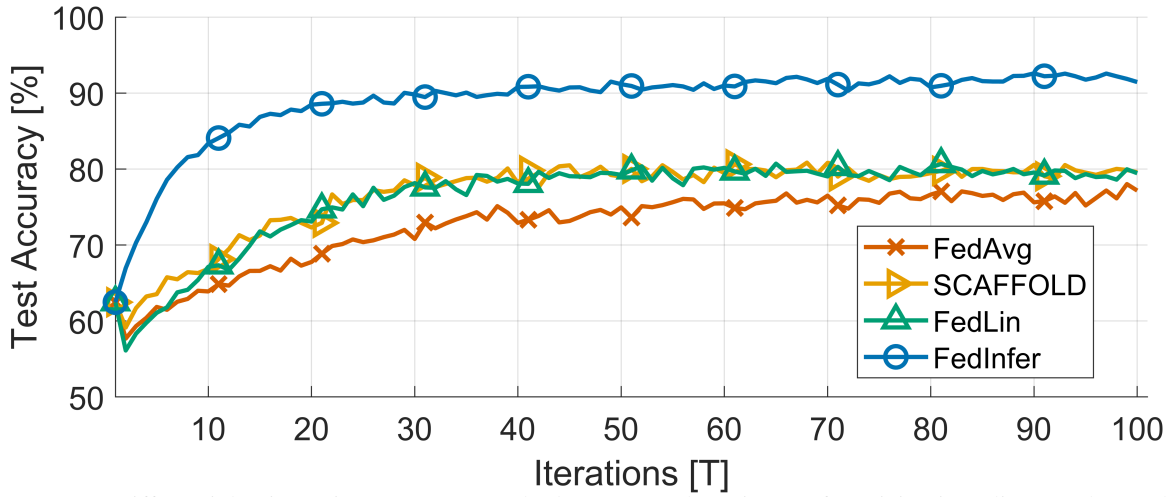

Figure 8.3: Differential Privacy is a common method to ensure the privacy of participating clients. The method involves injecting noise into the update step, in this case, we set the noise by fixing the privacy budget per step to $\varepsilon = 0.16$. This noise limits the peak accuracy of SCAFFOLD (**yellow** ▷) and FedLin (△) to 80%. Through its use of Bayesian Inference, FedInfer (**blue** ○) is able to converge to the true optima regardless of the level of injected noise.

injected noise directly limits the ability of gradient based optimizations from converging to the optimal solution. Because FedInfer naturally takes into account the injected noise, the algorithm is able to converge to the optimal global model via the Law of Large Numbers (LLN), Theorem 7.3.2.

Remark 9. *It should be noted that the results for the gradient based approaches are obtained with a fixed learning rate and that a higher accuracy could be obtained by progressively decreasing the learning rate over the course of learning. However, this would increase the number of communication steps these algorithms require to converge.*

8.4 Statistical Heterogeneity on the Fisher Iris Dataset

The next dataset we investigate is the Fisher Iris dataset [210, 211]. Each of the 150 datapoints is a four dimensional real-valued vector of unique measurements of a particular Iris flower petal. A pair plot showing the relationship between each feature vector is provided in Figure 8.4.

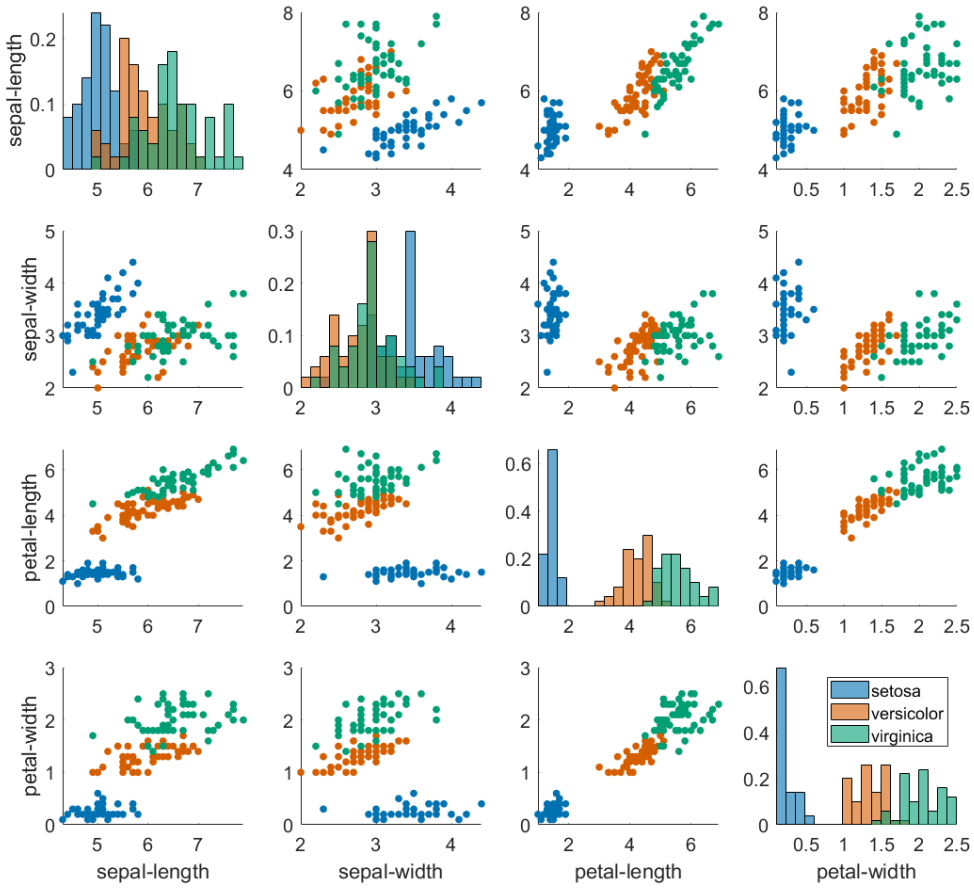

Figure 8.4: The Fisher Iris dataset is a collection of measurements of iris flower petals, which is frequently used as a classification benchmark dataset. Each scatter plot in the figure is a projection of the dataset onto two of the four features. The diagonal bar charts give the marginal distributions of each of the four features. From these graphs we can see that the Setosa iris is easy to classify, and that the main difficulty of this particular dataset is in distinguishing between Versicolors and Virginicas.

With this dataset, we additionally introduce the next challenge of a federated environment:

statistical heterogeneity. For the sake of comparison, we first give the results when the dataset is uniformly randomly split between three clients. In Figure 8.5, we once again see the depressed

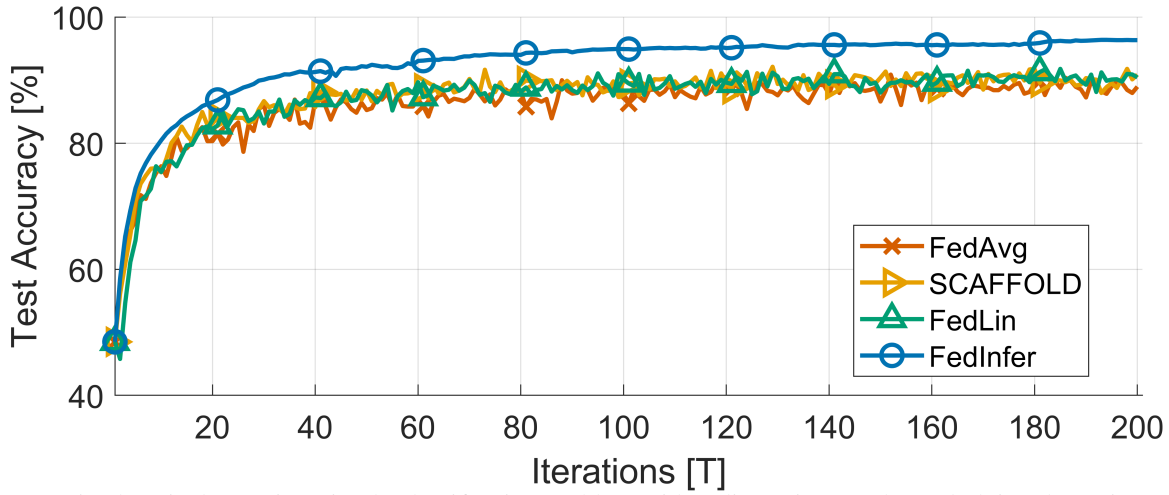

Figure 8.5: The Iris dataset is a simple classification problem with 4 dimensions to the underlying datapoints. We implement Local Differential Privacy and set each update to have a privacy budget of $\varepsilon = 1.38$. Due to the injected noise, all three gradient methods, FedAvg (**orange** $\times$), SCAFFOLD (**yellow** $\triangleright$), and FedLin (**green** $\triangle$), converge to a sub-optimal solution. FedInfer (**blue** $\circ$) is able to converge to the globally optimal solution regardless of noise.

accuracy results of the gradient based methods due to the injected noise, $(\varepsilon, \delta) = (1.38, 0.0095)$. Next, to simulate non-identically distributed data, each of the three clients is considered to be an *expert* in one specific flower type. To achieve this, each client holds all but three of the observed data points for a specific iris classification. The average results of 100 trials training under these conditions are seen in Figure 8.6. The first obvious observation is that we have clear delineations between all four methods. First, we observe that FedAvg achieves the lowest accuracy of 67%. This is because the method lacks any measure to mitigate the deviation of local updates from the global model to local optima. SCAFFOLD introduces a control variate to bias the update steps toward that of the global model. The results is observed in the significantly higher accuracy of 85%. FedLin improves upon SCAFFOLD with an accurate calculation of the control variate and so is able to reach 89% testing accuracy. Even when faced with severe statistical heterogeneity, FedInfer converges to the true global optima.

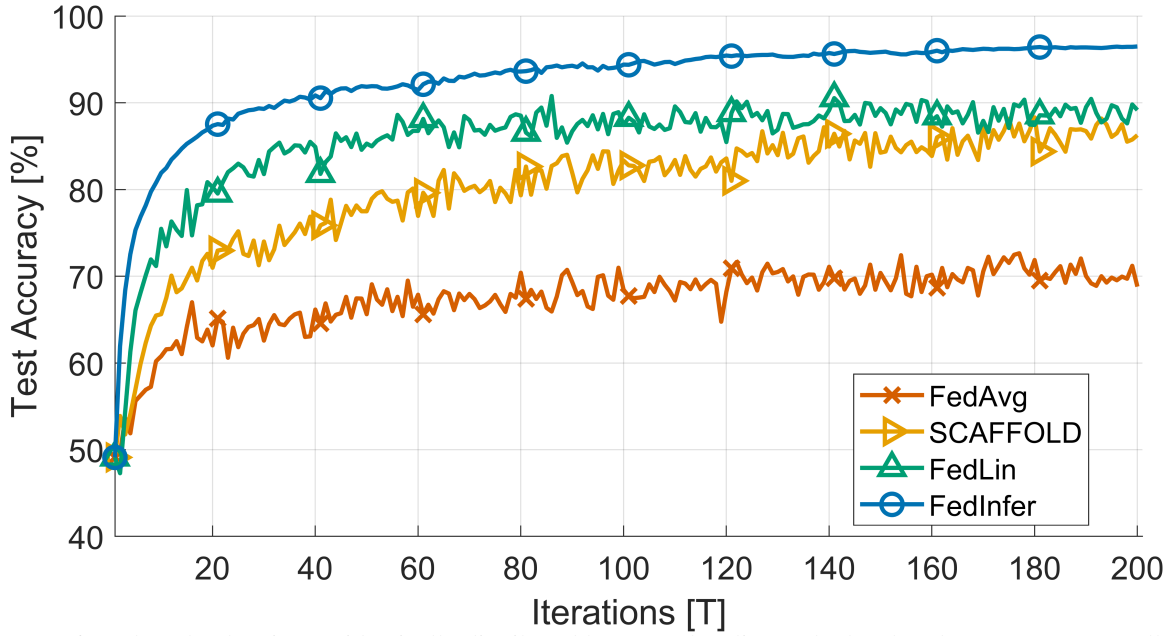

Figure 8.6: When the data is non-identically distributed between the clients, the local updates compete to pull the global model closer to their local optima. SCAFFOLD (**yellow** ▷) and FedLin (**green** △) use a control variant that help align the updates with the global cost function. FedInfer (**blue** ○) uses a globally consistent update steps to reach the same level of accuracy as centralized systems, even when the data is statistically heterogeneous.

8.5 System Heterogeneity with the Wisconsin Breast Cancer Dataset

The Wisconsin Breast Cancer (WBC) dataset is a widely used benchmark dataset in Machine Learning, primarily employed for classification tasks [212]. Originally collected by Dr. William H. Wolberg from the University of Wisconsin Hospitals, Madison, it comprises features computed from digitized images of fine needle aspirates (FNA) of breast masses. These features are then used to predict whether a detected mass is benign or malignant.

The dataset consists of 569 instances, with each instance representing a biopsy sample. There are 30 features computed from each biopsy, including attributes such as radius, texture, perimeter, area, smoothness, compactness, concavity, symmetry, fractal dimension, among others. These features are derived from images of the cell nuclei present in the FNA samples.

The target variable in this dataset indicates the diagnosis of the breast mass, with a value

of 0 representing a benign tumor and a value of 1 indicating a malignant tumor. As a result, the dataset poses a binary classification problem, where the goal is to develop models that accurately distinguish between benign and malignant breast masses based on the provided features.

Since privacy is often a crucial consideration with medical data, we again introduce Gaussian noise to ensure that update steps satisfy differential private guarantees. For the WBC dataset, each update is Locally Differentially Private with the privacy parameters $(\varepsilon, \delta) = (0.5, 0.0025)$. For this dataset, we selected four *clients* that could represent a group of collaborating hospitals. Without any heterogeneous assumptions, the test accuracy results from training the SVM are displayed in Figure 8.7. As with prior examples, the results are the average of 100 trials.

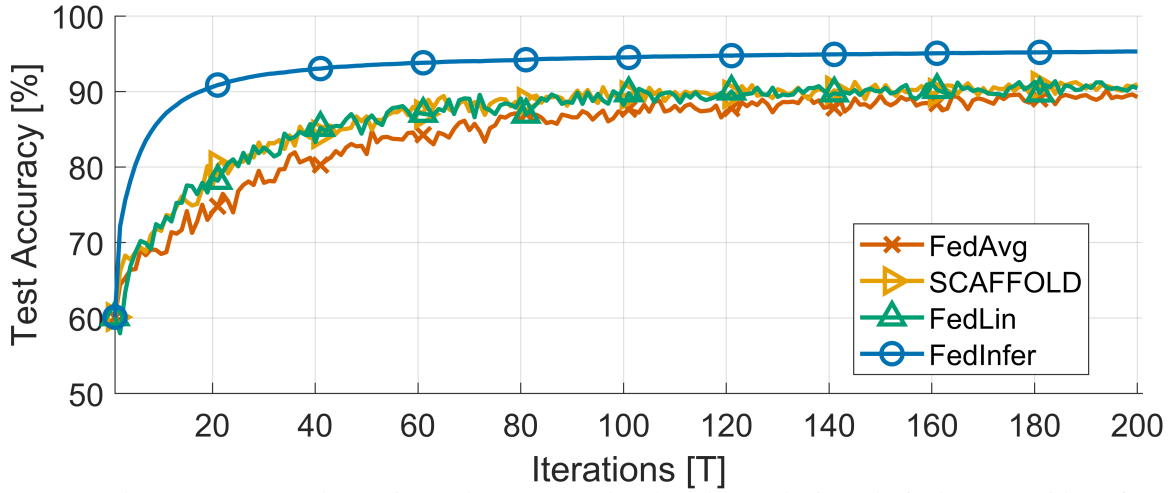

Figure 8.7: The Breast Cancer dataset is much more complex than the synthetic and Iris datasets, with 42 features per datapoint. The training presented here uses Locally Differentially Private updates with privacy budget $\varepsilon = 0.5$. Nevertheless, using kernel feature expansion, the SVM model trained using FedInfer (**blue** ○) is able to achieve 95% accuracy after 200 training iterations, above all three gradient methods, which all converge to 91% when not in the presence either system or statistical heterogeneity.

In the last set of trials, we introduce the final major challenge of the federated environment: system heterogeneity. To accomplish this, we assign each client a stagger rate that defines how long the server must wait, in terms of iterations, before receiving that client's update. The results displayed in Figure 8.8 have the following vector of stagger values $\{\gamma_k\} = [1\ 2\ 5\ 10]$ with value

γ_k indicating that the server receives that client's data every γ_k step, i.e., a value of 5 means the server receives data from that client every 5 steps. From these trajectories, we see that FedLin

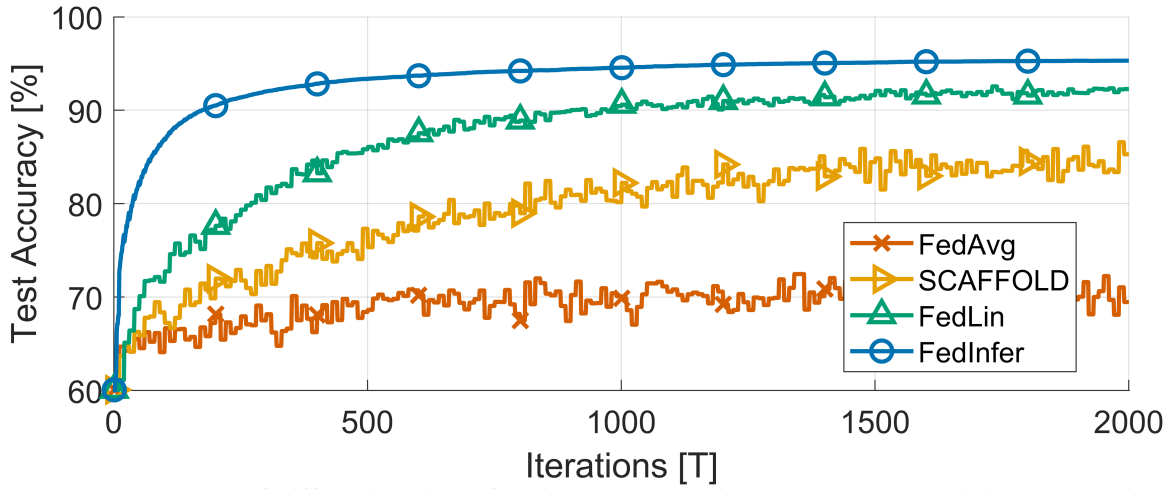

Figure 8.8: The presence of Differential Privacy for client level protection and system and statistic heterogeneity create a challenging environment to train Machine Learning models. In this simulation, we see the reduced convergence rates of both FedAvg(**orange** $\times$) and SCAFFOLD (**yellow** $\triangleright$). FedLin (**green** $\triangle$) is able to better accommodate the challenges of the environment with its control variate and local learning rates, but nevertheless struggles to reach 90% accuracy on this dataset. FedInfer (**blue** $\circ$), due to its global update step and asynchronous update scheme, quickly reaches $> 90\%$ accuracy.

has achieved significant advancements over the initial FedAvg algorithms that began the field of Federated Learning. However, when faced with statistical and system heterogeneity, FedLin does not reach 90% accuracy until 1000 iterations. Because FedInfer is an asynchronous algorithm, it is able to update the global model at every step. Additionally, by the Law of Large Numbers, the long term convergence of the model is unaffected by the non-identical distributions of data.

8.6 Concluding Remarks

It is important to acknowledge that some information loss by the clients is inevitable for the sake of achieving a global model. However, it is crucial for Federated Learning methods to recognize that gradient information may contain partial or even complete observations of the client's data. As such, it is necessary for federated environments to require privacy techniques

such as Local Differential Privacy. By injecting noise into the update step, we obfuscate the true direction of the gradient, negatively impacting the resulting model's accuracy on test data. FedInfer solves this problem by using Bayesian Inference to solve for the global optima, which does not rely on the gradient of local cost functions. Together with the Law of Large Numbers, this ensures that even in the presence of Differential Privacy techniques, the global model generated by FedInfer is guaranteed to converge to the optimal global solutions.

Moreover, we have shown how system and statistical heterogeneity, can bias the client gradient steps toward their local objectives, which may not fully align with the global model. While current Federated Learning techniques introduce different measures to mitigate the effect of these challenges, FedInfer is an asynchronous method that naturally accommodates both types of heterogeneity. In all the simulations above, FedInfer is able to converge to higher test accuracy using fewer communication steps.

These benefits are not without their limitations. FedInfer makes strong assumptions about the structure of the model being trained. For instance, it would likely struggle to converge if applied to a deep neural network. One area of research which may extend the applicability of FedInfer is Echo State Networks (introduced in Chapter 6). These types of networks utilize a mostly fixed network to extract features from data, and only train the final output layer which can be done by SVMs. In conclusion, Federated Inference is a novel algorithm in the Federated Learning field which uniquely approaches the core challenges of the federated environment.

Part III

System Identification

Chapter 9

Delayed System Identification

Many applications now require users to share private data with third-parties for processing and decision-making. Some example include smart grids, health monitoring, recommendation systems, traffic monitoring, fuel optimization, and industrial control systems in cloud computing [22–24]. In a smart grid, customers might share power consumption data for better rates, inadvertently revealing appliance usage and daily activities [26, 27]. It is for these reasons that robust privacy-preserving mechanisms are vital to address user concerns and increase participation, enhancing system efficiency. Federated Learning methods, like Federated Inference, can be used in system identification takes to ensure Differential Privacy for individual clients while constructing models that better predict general client behavior. In this chapter, instead of focusing on achieving privacy as in the prior chapters, we instead shift our focus to the types of problems that can be solved by system identification techniques.

9.1 Motivation

Privacy in control systems and system identification is paramount due to the sensitive nature of the data involved. Control systems often gather vast amounts of information about users'

behaviors, preferences, and activities, which can pose significant privacy risks if mishandled or compromised. System identification techniques, used to model and analyze these systems, inherently rely on data collection, potentially including personal or sensitive information. Therefore, ensuring robust privacy protection mechanisms, such as anonymization, encryption, and access control, is crucial to safeguarding individuals' privacy rights. By implementing these measures, control systems and system identification methodologies can operate effectively while mitigating the risks associated with unauthorized access or misuse of personal data.

Plant dynamics are rarely available *a priori*. System identification techniques have been developed in order to estimate the dynamics of systems that are unknown or have unknown parameters. For linear systems, least squares methods have had much success and are noted for their computational efficiency (see, e.g., [214], [215], [216]). A common technique for nonlinear system identification is to project the problem onto a linear space and then employ some of the aforementioned results [217]. Many of the methods have also been extended to stochastic approximations, assuming probability densities over the coefficients [218]. In recent years, deep learning techniques have emerged in which deep neural networks are used to approximate plant dynamics [219].

Dynamic mode decomposition (DMD) has shown much promise as an extension to least-squares methods [28]. The aim of dynamic mode decomposition is to identify the higher energy spatio-temporal frequencies from data. The method can be used to obtain a reduced-order model that characterizes the dominant elements of a system's dynamics. Dynamic mode decomposition has been shown to produce a finite approximation of the Koopman operator [220], an infinite dimensional linear operator that can fully characterize a nonlinear system. As a result, DMD is promising for modeling complex nonlinear dynamics given the right set of basis functions.

Extensions allow DMD to handle control system identification as well as online streaming data and time-varying systems [28], [221], [222].

DMD with control (DMDc) solves for a linear relationship between state observations and control inputs [221]. Delay between a control signal and its actuator is known to decrease the region of stability for delay-naive controllers (see, e.g., [223]. Delay (or dead-time) compensation has been studied for decades. A classical delay compensator is the Smith predictor [224]. In work by Laughlin et al. (1987) [225], controllers based on the Smith predictor showed robustness in the face of parameter uncertainty. In 2019, the predictor was used for cooperative adaptive cruise control systems in self-driving cars [226].

In this work, we extend DMDc to develop a method of dynamic mode decomposition with input-delayed control DMDidc [227]. This method automatically determines system dynamics as well as the input delay from data when a sufficiently rich input signal is applied during the data collection phase. Note that despite the fact DMD employs a discretized model, our method yields the system's continuous input delay.

9.2 Dynamic Mode Decomposition

In this section, we provide the context and necessary background for studying dynamic mode decomposition with input-delayed control. This paper focuses on approximating a continuous linear time-invariant (LTI) system with a delayed input. We will show that dynamic mode decomposition can be naturally extended to identify input delays that are a fractional multiple of the discrete-time sampling period.

Dynamic mode decomposition with control (DMDc) has been proposed as a natural exten-

sion of DMD [221]. DMDc simultaneously identifies the plant dynamics matrix $\mathbf{A}$ and the input matrix $\mathbf{B}$. Consider a continuous, LTI plant with a control vector $\mathbf{u}$:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_c\mathbf{x}(t) + \mathbf{B}_c\mathbf{u}(t). \tag{9.1}$$

Here $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector, $\dot{\mathbf{x}}(t) \in \mathbb{R}^n$ its time derivative, $\mathbf{A}_c \in \mathbb{R}^{n\times n}$ the (unknown) state dynamics matrix, $\mathbf{u}(t) \in \mathbb{R}^m$ in the control vector, and $\mathbf{B}_c \in \mathbb{R}^{m\times n}$ is the (unknown) control matrix. The control is taken to be piece-wise constant between sampling intervals:

$$\mathbf{u}(t) = \mathbf{u}_k, \quad t \in \Big[kT, (k+1)T\Big). \tag{9.2}$$

Dynamic mode decomposition approximates the plant in discrete-time using periodically observed data. Let the uniform sampling interval be T and write the discretized (sampled) state as:

$$\mathbf{x}_k := \mathbf{x}(kT), \quad k \in \mathbb{N}_0. \tag{9.3}$$

Given a sampling period T, DMDc solves for a discretization of the plant

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_k, \tag{9.4}$$

with discrete state dynamics and control matrices given by [228]

$$\mathbf{A}_d = e^{\mathbf{A}_c T}, \tag{9.5}$$

and

$$\mathbf{B}_d = \int_0^T e^{\mathbf{A}_c \tau} \mathrm{d}\tau \mathbf{B}_c. \tag{9.6}$$

In [221], the control signal is incorporated into the analysis by augmenting the dataset with the sequence of input values:

$$\mathcal{D} = \{(\mathbf{x}_0, \mathbf{u}_0), (\mathbf{x}_1, \mathbf{u}_1), (\mathbf{x}_2, \mathbf{u}_2), ..., (\mathbf{x}_K, \mathbf{u}_K)\}, \tag{9.7}$$

for some $K \in \mathbb{N}$. An augmented state matrix, $\mathbf{X}_{aug}$ is given by

$$\mathbf{X}_{aug} := \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{K-1} \\ \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_{K-1} \end{bmatrix}. \tag{9.8}$$

The one-step-ahead data matrix, $\mathbf{Y}$ is given by

$$\mathbf{Y} := \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_K \end{bmatrix}. \tag{9.9}$$

DMD estimates $(\mathbf{A}_d, \mathbf{B}_d)$ using least-squares approximation. In an over-determined system, the least-squares approximation is commonly taken as:

$$\begin{bmatrix} \hat{\mathbf{A}}_d & \hat{\mathbf{B}}_d \end{bmatrix} = \mathbf{Y}\mathbf{X}_{aug}^{\dagger} \tag{9.10}$$

where $\mathbf{X}^{\dagger}$ denotes the Moore-Penrose pseudoinverse of $\mathbf{X}$. The identified model is then given by

$$\mathbf{x}_{k+1} \approx \hat{\mathbf{A}}_d \mathbf{x}_k + \hat{\mathbf{B}}_d \mathbf{u}_k. \tag{9.11}$$

Remark 10. *One could consider using other pseudoinverses such as the unit-independent pseudoinverse proposed in [229].* ◀

9.3 Predictive Feedback Control

Consider a discrete, LTI plant with uniform input delay $w \in \mathbb{N}$ across all input channels:

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d \mathbf{u}_{k-w}. \tag{9.12}$$

When w is known, the controller can implement predictive feedback control, a form of model predictive control (MPC) [230]. Predictive feedback control works even if each input channel has a different delay, but for simplicity here we focus the case of single delay.

For a feedback gain matrix $\mathbf{K}$, future states can be extrapolated using the plant model:

$$\mathbf{u}_k = -\mathbf{K}\Big(\mathbf{A}_d^w \mathbf{x}_k + \sum_{i=k-w}^{k-1} \mathbf{A}_d^{k-1-i} \mathbf{B}_d \mathbf{u}_i\Big), \quad k \geq w. \tag{9.13}$$

Plugging $\mathbf{u}_{k-w}$ into Equation (9.12) results in

$$\mathbf{x}_{k+1} = (\mathbf{A}_d - \mathbf{B}_d \mathbf{K})\mathbf{x}_k, \quad k \geq r. \tag{9.14}$$

Thus, we can design $\mathbf{K}$ for an input-delayed system using standard methods such as pole placement or LQR.

9.4 Dynamic Mode Decomposition with Input Delayed Control

A main challenge in designing controllers is dealing with input delay. Large delays can render a naive controller unstable [223]. To address this, we propose dynamic mode decomposition with input-delayed control (DMDidc). Our method simultaneously solves for the dynamics of the state, for the controller, and for the input delay.

DMDidc is an offline method consisting of two stages. In the first stage, the plant is driven by an input signal that sufficiently excite the plant. Corresponding state-input pairs are recorded to a dataset. In the second stage, the dynamics of the plant are identified from the obtained dataset. We provide sufficient conditions on the input to ensure that the plant and delay can be identified given measurements over a sufficiently long interval. Additionally, we note a theorem of W.J. Culver (1966) [231] giving a sufficient condition for the existence of a unique real solution. We first provide a random input signal that is sufficiently rich and proceed to show that the system can be identified.

Like DMDc, our method identifies a discretized version of a continuous plant, extending the prior work to continuous-time plants with a constant input delay:

$$\dot{\mathbf{x}} = \mathbf{A}_c\mathbf{x}(t) + \mathbf{B}_c\mathbf{u}(t-r), \quad r \in [0, R). \tag{9.15}$$

We assume the delay r is positive and bounded by a known quantity R. In this paper, for simplicity of exposition we take r to be constant across all input channels. Although tedious, the method can be extended to handle distinct delays for the various input channels. Our approach can be implemented for any admissible control signals and does not involve the brute force approach of

multiple runs with step inputs to observe when the step appears in the state.

Consider a zero-order hold controller with fixed sampling period T. The input delay, r, can be decomposed into a whole, w, and a fractional, p, multiples of the sampling period:

$$r = wT + pT, \quad w \in \mathbb{N}_0,\ p \in [0, 1). \tag{9.16}$$

Let $\mathbf{F}_c$ be the concatenation of the state dynamics and input coefficient matrices:

$$\mathbf{F}_c := \begin{bmatrix} \mathbf{A}_c & \mathbf{B}_c \end{bmatrix}. \tag{9.17}$$

The objective of dynamic mode decomposition, in this context, is to solve for the delay, r, and the discretization $\mathbf{F}_d$ of the dynamics as summarized by $\mathbf{F}_c$:

$$\mathbf{Y} = \mathbf{F}_d\mathbf{X} = \begin{bmatrix} \mathbf{A}_d & \mathbf{B}_d \end{bmatrix} \mathbf{X}, \tag{9.18}$$

where $\mathbf{A}_d$ and $\mathbf{B}_d$ are given by Equations (9.4) and (9.5), respectively. As in DMDc, we augment the data matrix, $\mathbf{X}$, with a range of control values. Because we assumed the delay is bounded by a known value R, we can expand the matrix over these historical values to ensure the delay is

captured:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_R & \mathbf{x}_{R+1} & \cdots & \mathbf{x}_{R+K} \\ \mathbf{u}_R & \mathbf{u}_{R+1} & \cdots & \mathbf{u}_{R+K} \\ \mathbf{u}_{R-1} & \mathbf{u}_R & \cdots & \mathbf{u}_{R+K-1} \\ \vdots & \vdots & \cdots & \vdots \\ \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_K \end{bmatrix}. \tag{9.19}$$

The next-step matrix, $\mathbf{Y}$, remains the same and is given as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{x}_{R+1} & \mathbf{x}_{R+2} & \cdots & \mathbf{x}_{R+K+1} \end{bmatrix}. \tag{9.20}$$

When the control is sufficiently excited and the data collection is for an adequately long duration, the data matrix is invertible, allowing us to write

$$\begin{aligned} \hat{\mathbf{F}}_d &:= \mathbf{Y}\mathbf{X}^{-1}. \\ &= \begin{bmatrix} \hat{\mathbf{A}}_d & \hat{\mathbf{B}}_d^{(0)} & \hat{\mathbf{B}}_d^{(1)} & \cdots & \hat{\mathbf{B}}_d^{(R)} \end{bmatrix}. \end{aligned} \tag{9.21}$$

The block matrix, $\hat{\mathbf{B}}^{(k)}$, represents the control coefficients for an input delay of $r = k$ sampling periods. In this work, we leave the modeling of noise and perturbations for future research efforts.

9.4.1 Sufficient Input Excitation

Identifying control systems from data is a challenging task as the observations must contain enough information on all the dynamic modes [232]. When $\mathbf{B} \neq \mathbf{0}$, the problem includes design-

ing an input signal that can excite all modes of the plant. To this end, we employ discrete-time noise to persistently excite the plant dynamics [233]. This allows us to prove the richness of the input.

We take Θ to be a continuous random variable with zero mean, $\mathbb{E}[\Theta] = 0$, and finite variance, $\text{Var}[\Theta] < \infty$. During the data collection phase, the input is a discrete-time random process, $\mathbf{u}_k$, independently drawn from distribution Θ at each time-step k.

The goal of this section is to prove that the augmented data matrix (9.19) is invertible after sufficient enough number of observations. We begin with a lemma that ensures each block matrix in (9.52) necessarily increases the rank of the whole matrix. This lets us prove a basis for the matrix can be evenly distributed across the blocks.

Lemma 9.4.1. *Let $(\boldsymbol{A}, \boldsymbol{B})$ be a controllable discrete-time LTI plant. Let k be the smallest positive integer for which*

$$\textit{rank}(\begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} & \dots & \boldsymbol{A}^k\boldsymbol{B} \end{bmatrix}) = n. \tag{9.22}$$

Then:

$$\begin{aligned} 0 < \textit{rank}(\begin{bmatrix} \boldsymbol{B} \end{bmatrix}) < \textit{rank}(\begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} \end{bmatrix}) < \cdots \\ \cdots < \textit{rank}(\begin{bmatrix} \boldsymbol{B} & \dots & \boldsymbol{A}^k\boldsymbol{B} \end{bmatrix}) = n. \end{aligned} \tag{9.23}$$

Proof. Assume Equation (9.23) does not hold. That is, assume there exists a value $\ell < k$ such that

$$\begin{aligned} \text{rank}(\begin{bmatrix} \mathbf{B} & \dots & \mathbf{A}^\ell\mathbf{B} \end{bmatrix}) = \text{rank}(\begin{bmatrix} \mathbf{B} & \dots & \mathbf{A}^{\ell+1}\mathbf{B} \end{bmatrix}) \\ =: q < n. \end{aligned} \tag{9.24}$$

By adding columns to a matrix, the rank cannot decrease, so we are left with the case where the rank remains constant between steps. This implies that the matrix $\mathbf{A}^{\ell+1}\mathbf{B}$ can be written as a

linear combination of prior columns. Let $\{\mathbf{v}_1, \cdots, \mathbf{v}_q\}$ be a set of linearly independent vectors contained by the left side of Eq. (9.24). We can rewrite the i^{th} column of $\mathbf{A}^{\ell+1}\mathbf{B}$ as

$$\left[\mathbf{A}^{\ell+1}\mathbf{B}\right]_i = \sum_{j=1}^{q} \alpha_j^{(i)} \mathbf{v}_j. \tag{9.25}$$

This implies

$$\mathbf{A}\left[\mathbf{A}^{\ell+1}\mathbf{B}\right]_i = \sum_{j=1}^{q} \alpha_j^{(i)} \mathbf{A}\mathbf{v}_j. \tag{9.26}$$

By the above, $\mathbf{A}\mathbf{v}_j$ is also already in the span of the columns of the matrix on the left side of (9.24). As a result,

$$\text{rank}(\begin{bmatrix}\mathbf{B} & \cdots & \mathbf{A}^k\mathbf{B}\end{bmatrix}) = q < n, \tag{9.27}$$

a contradiction. □

The next auxiliary Lemma allows us to group unused terms into a single vector. It states that adding a set of arbitrary vectors $\{\mathbf{d}_i\}$ to a given set of linearly independent vectors $\{\mathbf{v}_1, ..., \mathbf{v}_n\}$, with multiplication by random scalars, does not affect the full-rank property of the vectors $\{\mathbf{v}_i\}$.

Lemma 9.4.2. *Let $\{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_n\}$ be a set of n linearly independent vectors and $\{\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_n\}$ be a set of arbitrary vectors. Take $\{\theta_1, ..., \theta_n\}$ to be a sequence of n i.i.d. samples of the random variable Θ, defined above. Then, the matrix*

$$\boldsymbol{H} := \begin{bmatrix}\boldsymbol{v}_1\theta_1 + \boldsymbol{d}_1 & \boldsymbol{v}_2\theta_2 + \boldsymbol{d}_2 & \cdots & \boldsymbol{v}_n\theta_n + \boldsymbol{d}_n\end{bmatrix} \tag{9.28}$$

has rank n with probability 1.

Proof. Consider the vector $\mathbf{d}_i$ in terms of $\{\mathbf{v}_1, ..., \mathbf{v}_n\}$:

$$\mathbf{d}_i = \alpha_{i,1}\mathbf{v}_1 + \alpha_{i,2}\mathbf{v}_2 + ... + \alpha_{i,n}\mathbf{v}_n. \tag{9.29}$$

Substituting (9.29) into matrix (9.28) gives

$$\mathbf{h}_i = \alpha_{i,1}\mathbf{v}_1 + ... + (\alpha_{i,i} + \theta_i)\mathbf{v}_i + ... + \alpha_{i,n}\mathbf{v}_n. \tag{9.30}$$

Because θ_i is drawn from a continuous distribution, the probability that it exactly cancels with $\alpha_{i,i}$ is 0.

We then proceed with a proof by contradiction. Without loss of generality we assume that $\mathbf{h}_n \in \mathcal{R}([\mathbf{h}_1, ..., \mathbf{h}_{n-1}])$. Then, there exists a set of constants $\{\beta_1, ..., \beta_{n-1}\}$ such that

$$\mathbf{h}_n = \sum_{i=1}^{n-1} \beta_i \mathbf{h}_i. \tag{9.31}$$

Because the representation of a vector in terms of a given basis is unique, the coefficients of $\{\mathbf{v}_i\}$ in the expansions of the two sides of Equation (9.31) must agree:

$$\alpha_{n,j}\mathbf{v}_j = \sum_{i=1}^{n-1} \beta_i \hat{\alpha}_{i,j}\mathbf{v}_j, \quad \hat{\alpha}_{i,j} = \begin{cases} \alpha_{i,j} & i \neq j \\ (\alpha_{i,i} + \theta_i) & i = j. \end{cases} \tag{9.32}$$

where $j \in \{1, 2, \ldots, n-1\}$.

Next, consider the coefficient of $\mathbf{v}_n$ in the basis expansion (9.30) of $\mathbf{h}_n$. For (9.31) hold, the

coefficients of $\mathbf{v}_n$ must match on both sides of the equation:

$$(\alpha_{n,n} + \theta_n)\mathbf{v}_n = \sum_{i=1}^{n-1} \beta_i \alpha_{i,n} \mathbf{v}_n. \tag{9.33}$$

However, because the β's have already been solved for, the equation is only true for a single value of θ_n. This leads to a contradiction as the probability θ_n takes a value in a set of Lebesgue measure zero is 0.

Thus, $\mathbf{h}_n \notin \mathcal{R}([\{\mathbf{h}_1, ..., \mathbf{h}_{n-1}])$ with probability 1. Since we focused arbitrarily on the n^{th} vector $\mathbf{h}_n$, this proof holds for each $\mathbf{h}_i$. Hence, the columns of $\mathbf{H}$ are linearly independent with probability 1, i.e., the rank of $\mathbf{H}$ is n with probability 1. □

The final lemma asserts that a sequence of n observed state vectors, under sufficient conditions, form a matrix of rank n.

Lemma 9.4.3. *Let $(\boldsymbol{A}, \boldsymbol{B})$ be a controllable discrete LTI system with state $\boldsymbol{x} \in \mathbb{R}^n$. Consider a sequence of n observations of the state vector under a randomized input sequence. Let each input be obtained by independent sampling from the probability distribution Θ. Then, the matrix*

$$\begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_n \end{bmatrix}, \tag{9.34}$$

has rank n with probability 1.

Proof. The matrix of state vectors (9.34) can be expanded using the variation of constants formula for the solution to (9.3):

$$\begin{bmatrix} \mathbf{A}\mathbf{x}_0 + \mathbf{B}\mathbf{u}_0, & \cdots, & \mathbf{A}^{n-1}\mathbf{x}_0 + \sum_{i=0}^{n-1} \mathbf{A}^{n-1-i}\mathbf{B}\mathbf{u}_i \end{bmatrix}. \tag{9.35}$$

Denote the controllability matrix by

$$\mathcal{C} := \begin{bmatrix} \mathbf{B} & \mathbf{AB} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix}. \tag{9.36}$$

Let $\{\mathbf{v}_1, ..., \mathbf{v}_n\}$ be a basis for the columns of $\mathcal{C}$ satisfying (possible by Lemma 9.4.1)

$$\begin{aligned} \mathbf{v}_1 &\in \mathcal{R}\{[\mathbf{B}]\} \\ \mathbf{v}_2 &\in \mathcal{R}\{[\mathbf{B}\ \mathbf{AB}]\} \\ &\vdots \\ \mathbf{v}_n &\in \mathcal{R}\{[\mathbf{B}\ \cdots\ \mathbf{A}^{n-1}\mathbf{B}]\}. \end{aligned} \tag{9.37}$$

Consider the first block matrix (leftmost element) in matrix (9.35). The control term $\mathbf{Bu}_0$ can be broken down into a linear combination of the columns of $\mathbf{B}$ scaled by the random input:

$$\mathbf{Bu}_0 = \mathbf{b}_1 u_0^{(1)} + \mathbf{b}_2 u_0^{(2)} + ... + \mathbf{b}_m u_0^{(m)}. \tag{9.38}$$

From (9.37), $\mathbf{v}_1 \in \mathcal{R}\{[\mathbf{B}]\}$. This lets us write $\mathbf{v}_1$ as a linear combination of the columns of $\mathbf{B}$:

$$\mathbf{v}_1 = \gamma_1 \mathbf{b}_1 + \gamma_2 \mathbf{b}_2 + ... + \gamma_m \mathbf{b}_m. \tag{9.39}$$

Define the random variable $\xi_{i,j}$ as follows:

$$\xi_{i,j} := u_{j-1}^{(i)} - \gamma_i \theta_j, \quad u_{j-1}^{(i)}, \theta_j \sim \Theta. \tag{9.40}$$

Eqs. (9.40), (9.39), and (9.38) together imply:

$$\begin{aligned}\mathbf{B}\mathbf{u}_0 &= \mathbf{b}_1(\gamma_1\theta_1 + \xi_{1,1}) + ... + \mathbf{b}_m(\gamma_m\theta_1 + \xi_{m,1}) \\ &= \mathbf{v}_1\theta_1 + \mathbf{b}_1\xi_{1,1} + ... + \mathbf{b}_m\xi_{m,1}.\end{aligned} \tag{9.41}$$

Next we aim to rewrite the first block of (9.35) in the form

$$\mathbf{A}\mathbf{x}_0 + \mathbf{B}\mathbf{u}_0 = \mathbf{v}_1\theta_1 + \mathbf{d}_1. \tag{9.42}$$

Note that Lemma 9.4.2 holds for an arbitrary vector $\mathbf{d}$, and so we can group the unused terms into a vector as follows:

$$\mathbf{d}_1 := \mathbf{A}\mathbf{x}_0 + \mathbf{b}_1\xi_{1,1} + ... + \mathbf{b}_m\xi_{m,1}. \tag{9.43}$$

The next block in the matrix (9.35) can be similarly written (again using Lemma 9.4.1) in terms of the second basis vector:

$$\mathbf{v}_2\theta_2 + \mathbf{d}_2. \tag{9.44}$$

This process can be continued for all n block matrices in (9.35). In this way, the matrix (9.35) can be rewritten in the form (9.28) which by Lemma 9.4.2 has rank n with probability 1.

□

Theorem 9.4.4. *(Augmented Matrix Invertibility: No delay) Consider a discrete-time, LTI plant:*

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \tag{9.45}$$

with $\boldsymbol{x}_k \in \mathbb{R}^n$ and $\boldsymbol{u}_k \in \mathbb{R}^m$. Assume the full state is observable and $(\boldsymbol{A}, \boldsymbol{B})$ is controllable. Let the

components of the input signals $\{\boldsymbol{u}_0, ..., \boldsymbol{u}_{n+m}\}$ *be drawn from the continuous random variable* Θ. *Then, the square matrix,*

$$\begin{bmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_{n+m} \\ \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_{n+m} \end{bmatrix}, \tag{9.46}$$

is invertible with probability 1.

Proof. We prove the theorem by induction on the dimension m of the input vector. Start with the case $m = 0$ (no input signals). The resulting matrix is invertible by Lemma 9.4.3. Next, we assume the matrix in (9.46) has full rank for an input dimension of m. Consider the induction step with $m + 1$:

$$\mathbf{G} := \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_{n+m+1} \\ \tilde{\mathbf{u}}_1 & \cdots & \tilde{\mathbf{u}}_{n+m+1} \end{bmatrix}, \tag{9.47}$$

where $\tilde{\mathbf{u}}_j \in \mathbb{R}^{m+1}$, the first m elements of which match those of $\mathbf{u}_j$. Denote the i^{th} column of (9.47) by $\mathbf{g}_i$. Because (9.46) has full rank, there exists a sequence of scalars, $\{\beta_1, \ldots, \beta_{n+m}\}$ such that the first $n + m$ elements of $\mathbf{g}_{n+m+1}$ can be written as:

$$g^{(j)}_{n+m+1} = \sum_{i=1}^{n+m} \beta_i g^{(j)}_i, \quad j \in [1, n + m]. \tag{9.48}$$

Now suppose that $\mathbf{g}_{n+m+1} \in \mathcal{R}([\mathbf{g}_1, ..., \mathbf{g}_{n+m}])$. Then:

$$g^{(n+m+1)}_{n+m+1} = \sum_{i=1}^{n+m} \beta_i g^{(n+m+1)}_i. \tag{9.49}$$

However, the probability the $(n + m + 1)^{st}$ element of $\mathbf{g}_{n+m+1}$ takes the value in (9.49) is 0. Therefore, the vector $\mathbf{g}_{n+m+1}$ is linearly independent from $\mathcal{R}([\mathbf{g}_1, ..., \mathbf{g}_{n+m}])$ and $\mathbf{G}$ is invertible

with probability 1. □

Next, we turn our attention to the augmented matrix (9.19) for the input-delayed problem. Recall that the delay is assumed to be bounded by R. To ensure that the delay is captured in the data, we excite the plant with a random input for at least R sample times. We aim to prove that the augmented data matrix $\mathbf{X}$ is invertible, implying that the coefficients of the plant can be uniquely determined.

Theorem 9.4.5. *(Augmented Matrix Invertibility: Delay) Let the plant coefficients and input distribution be as described in Theorem 9.4.4. Let the plant input be delayed by a constant scalar r bounded by R. Consider $n + mR$ observations of the state and input signal. Then, the augmented state matrix constructed by concatenating the last $R + 1$ input vectors to each state,*

$$\hat{\boldsymbol{G}} := \begin{bmatrix} \boldsymbol{x}_R & \boldsymbol{x}_{R+1} & \cdots & \boldsymbol{x}_{R+(m(R+1)+n)} \\ \boldsymbol{u}_R & \boldsymbol{u}_{R+1} & \cdots & \boldsymbol{u}_{R+(m(R+1)+n)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{u}_0 & \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_{m(R+1)+n} \end{bmatrix}, \tag{9.50}$$

is invertible with probability 1.

Proof. The proof follows by induction on the delay bound R. For the base step with $R = 0$ (no delay) follows directly from Theorem 9.4.4. Consider the inductive step; assume the matrix for

the bound $R-1$ is invertible with probability 1:

$$\begin{bmatrix} \mathbf{x}_{R-1} & \mathbf{x}_R & \cdots & \mathbf{x}_{R-1+(mR+n)} \\ \mathbf{u}_{R-1} & \mathbf{u}_R & \cdots & \mathbf{u}_{R-1+(mR+n)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{u}_0 & \mathbf{u}_1 & \cdots & \mathbf{u}_{(mR+n)} \end{bmatrix}, \tag{9.51}$$

We aim to show that the matrix $\hat{\mathbf{G}}$ is also invertible w.p.1.

Consider the sub-matrix in the upper right block of $\hat{\mathbf{G}}$:

$$\left[\begin{array}{ccc:ccc} \cdots & \mathbf{x}_{R+m-1} & \mathbf{x}_{R+m} & \cdots & \mathbf{x}_{R+(m(R+1)+n)} \\ \cdots & \mathbf{u}_{R+m-1} & \mathbf{u}_{R+m} & \cdots & \mathbf{u}_{R+(m(R+1)+n)} \\ & & \vdots & \ddots & \vdots \\ \cdots & \mathbf{u}_m & \mathbf{u}_{m+1} & \cdots & \mathbf{u}_{m(R+1)+n+1} \\ \hdashline \cdots & \mathbf{u}_{m-1} & \mathbf{u}_m & \cdots & \mathbf{u}_{m(R+1)+n} \end{array}\right] \tag{9.52}$$

Define $\tilde{\mathbf{G}}$ to be the upper right block indicated in (9.52). Note that $\tilde{\mathbf{G}}$ is (9.51) shifted in time by $m+1$ steps. Because the plant is time invariant, $\tilde{\mathbf{G}}$ is invertible w.p.1. by the induction step hypothesis.

Next, because the $\tilde{\mathbf{G}}$ has full rank, we can write the first $mR+n$ elements of the column vector $\hat{\mathbf{g}}_{R+m-1}$ as

$$\hat{g}^{(j)}_{R+m-1} = \sum_{i=R+m}^{R+(m(R+1)+n)} \beta_i \hat{g}^{(j)}_i, \quad j \in \{1, ..., Rm+n\}. \tag{9.53}$$

Now assume that $\hat{\mathbf{g}}_{R+m-1} \in \mathcal{R}([\hat{\mathbf{g}}_{R+m}, ..., \hat{\mathbf{g}}_{R+m(R+1)+n}])$. This implies:

$$\hat{\mathbf{g}}_{R+m-1} = \sum_{i=R+m}^{R+(m(R+1)+n)} \beta_i \hat{\mathbf{g}}_i \tag{9.54}$$

However, the first element of the $(m-1)^{st}$ control signal has components drawn independently from the distribution Θ:

$$\Pr\left(u_{m-1}^{(1)} = \sum_{i=m}^{m(R+1)+n} \beta_{R+i} u_i^{(1)}\right) = 0, \tag{9.55}$$

a contradiction. Therefore, the vector $\hat{\mathbf{g}}_{R+m-1} \notin \mathcal{R}([\hat{\mathbf{g}}_{R+m}, ..., \hat{\mathbf{g}}_{R+m(R+1)+n}])$. The same argument holds for the remaining $m-1$ columns. As a result, the matrix $\hat{\mathbf{G}}$ is invertible with probability 1. □

Remark 11. *The derivation above works in the noise free, linear case. For less ideal models, one can arrange convergence nearer to the true plant values by lengthening the data collection phase [234–236].* ◀

9.4.2 Delay Identification

Consider a continuous, LTI system with input delay r:

$$\dot{\mathbf{x}} = \mathbf{A}_c \mathbf{x}(t) + \mathbf{B}_c \mathbf{u}(t-r), \quad r \in [0, R). \tag{9.56}$$

The time domain solution to (9.56) is derived by [228]:

$$\mathbf{x}(t) = e^{\mathbf{A}_c(t-t_0)}\mathbf{x}(t_0) + \int_{t_0}^{t} e^{\mathbf{A}_c(t-\tau)}\mathbf{B}_c \mathbf{u}(\tau - r)\mathrm{d}\tau. \tag{9.57}$$

Recall that we can write the discretized state as:

$$\mathbf{x}_k = \mathbf{x}(kT). \tag{9.58}$$

Solving for $\mathbf{x}_{(k+1)}$ in terms of $\mathbf{x}_k$ with the change of variable, $\sigma := \tau - kT$, gives:

$$\mathbf{x}_{k+1} = e^{\mathbf{A}_c T}\mathbf{x}_k + \int_0^T e^{\mathbf{A}_c(T-\sigma)}\mathbf{B}_c\mathbf{u}\big(kT + \sigma - r\big)\mathrm{d}\sigma. \tag{9.59}$$

The presence of the continuous-time delay r can offset input value discontinuities from the sampling instants. Observe that the input $\mathbf{u}(t)$ can only take one or two values within a sampling interval. Recall that we split r into two components, a whole, w, and a partial, p, multiple of the sampling interval:

$$r = wT + pT, \quad w \in \mathbb{N}_0,\ p \in [0, 1). \tag{9.60}$$

When $p = 0$, the control is constant during the interval and can be pulled out of the integral. Then from Eqs. (9.4) and (9.3),

$$\mathbf{x}_{k+1} = \mathbf{A}_d\mathbf{x}_k + \mathbf{B}_d\mathbf{u}_{k-w}. \tag{9.61}$$

When $p \neq 0$, the control changes partway through the integral:

$$\mathbf{B}_d^{(w)}\mathbf{u}_{(k-w)} = \Big(\int_{pT}^{T} e^{\mathbf{A}_c(T-\sigma)}\mathrm{d}\sigma\mathbf{B}_c\Big)\mathbf{u}_{(k-w)}, \tag{9.62}$$

$$\mathbf{B}_d^{(w+1)}\mathbf{u}_{(k-w-1)} = \Big(\int_{0}^{pT} e^{\mathbf{A}_c(T-\sigma)}\mathrm{d}\sigma\mathbf{B}_c\Big)\mathbf{u}_{(k-w-1)}. \tag{9.63}$$

Expanding the plant dynamics in Equation (9.56) to include all historical input values, the control's influence on the dynamics can be seen to have the following structure:

$$\begin{bmatrix} & | & | & | & | & \\ \cdots & \mathbf{0} & \mathbf{B}_d^{(w)} & \mathbf{B}_d^{(w+1)} & \mathbf{0} & \cdots \\ & | & | & | & | & \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \\ \mathbf{u}_0 \end{bmatrix}. \tag{9.64}$$

Note that all but two of the vectors, $\mathbf{B}_d^{(w)}$ and $\mathbf{B}_d^{(w+1)}$, are nonzero. These terms complete the one-step discretization of the plant:

$$\mathbf{x}_{k+1} = \mathbf{A}_d \mathbf{x}_k + \mathbf{B}_d^{(w)} \mathbf{u}_{k-w} + \mathbf{B}_d^{(w+1)} \mathbf{u}_{k-w-1}. \tag{9.65}$$

The whole-valued delay component, w, can be found by extracting the first nonzero vector from the left matrix in (9.64):

$$\begin{aligned} w = &\min_j \ j \\ &\text{s.t. } \mathbf{B}_d^{(j)} \neq \mathbf{0}, \quad \mathbf{0} \in \mathbb{R}^m \end{aligned} \tag{9.66}$$

The partial component of the delay can be derived by solving the system of equations, (9.4), (9.62), and (9.63):

$$\mathbf{A}_d = e^{\mathbf{A}_c T} \tag{9.67}$$

$$\mathbf{A}_c \mathbf{B}_d^{(w)} = (e^{\mathbf{A}_c (1-p) T} - \mathbf{I}) \mathbf{B}_c \tag{9.68}$$

$$\mathbf{A}_c \mathbf{B}_d^{(w+1)} = (e^{\mathbf{A}_c T} - e^{\mathbf{A}_c (1-p) T}) \mathbf{B}_c. \tag{9.69}$$

The matrix logarithm is required to find $\mathbf{A}_c$ in (9.67). This has a unique real solution iff no Jordan block of $\mathbf{A}_d$ appears more than once [231]. If the state vector measurements are exact, and the discretization comes from a continuous-time LTI model, there exists *at least one continuous-time model and time delay*. It remains an open question whether (9.68) and (9.69) impose sufficient constraints for the uniqueness of $\mathbf{A}_c$. Empirical tests indicate converges to the correct values.

9.4.3 Feedback Stabilization with Fractional Delay

The stabilizing predictive feedback control described in 9.3 only handles a whole valued delay. Here, we present an alternative method to accommodate delays that do not coincide with the input discontinuities.

Consider the k^{th} observation $\mathbf{x}_{kT}$. To calculate a control value that induces standard closed-loop behavior, we must determine the state when the input will be applied:

$$\mathbf{x}(kT+r) = e^{\mathbf{A}_c r}\mathbf{x}_k + \int_{kT}^{kT+r} e^{\mathbf{A}_c(kT+r-\tau)}\mathbf{B}_c\mathbf{u}(\tau - r)\mathrm{d}\tau. \tag{9.70}$$

First, we apply the one-step update (9.65) w times to calculate $\mathbf{x}_{k+w}$. Then, to determine the contribution of the fractional component, we require a fractional state and control matrix:

$$\mathbf{A}_d^{(p)} := \mathbf{A}_d^p, \tag{9.71}$$

and

$$\mathbf{B}_d^{(p)} := \int_0^{pT} e^{A_c(pT-\tau)}d\tau B_c. \tag{9.72}$$

We can calculate $\mathbf{x}(kT + r)$ as follows:

$$\mathbf{x}(kT + r) = \mathbf{A}_d^{(p)}\mathbf{x}_{k+w} + \mathbf{B}_d^{(p)}\mathbf{u}_{k-1}. \tag{9.73}$$

Finally, given a feedback gain matrix $\mathbf{K}$, using the control

$$\mathbf{u}_k = -\mathbf{K}\mathbf{x}(kT + r), \tag{9.74}$$

at the k^{th} time step gives the following closed-form dynamics:

$$\mathbf{x}_{k+1} = (\mathbf{A}_d - \mathbf{B}_d\mathbf{K})\mathbf{x}_k, \quad k \geq r. \tag{9.75}$$

9.5 Simulation Results

In this section, we provide simulated results that demonstrates the necessity of fractional delay control. We compare the convergence of two controllers, a predictive feedback controller for the whole component of the delay w, and the fractional delay feedback controller described in 9.4.3.

We consider the following continuous matrices:

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1.1 \\ -1.1 & 0 \end{bmatrix} \text{ and } \mathbf{B}_c = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}. \tag{9.76}$$

To better illustrate the affect of an unaccounted fractional input delay, we select a sampling

interval of $T = 2$. The discretized plant for this sample interval is

$$\mathbf{A}_d = \begin{bmatrix} -0.589 & 0.809 \\ -0.809 & -0.589 \end{bmatrix}, \ \mathbf{B}_d = \begin{bmatrix} -0.709 & 2.153 \\ -2.179 & 2.914 \end{bmatrix}. \tag{9.77}$$

The larger interval allows the plant to evolve more between control updates. In this test, we compare the trajectories of three input delays, $(1T,\ 1.25T,\ 1.5T)$. For each of these delays, DMDidc returns a matrix of the following form:

$$\mathbf{Y}\mathbf{X}^{\dagger} = \begin{bmatrix} \mathbf{A}_d & \mathbf{0} & \mathbf{0} & \mathbf{B}_d^{(w)} & \mathbf{B}_d^{(w+1)} \end{bmatrix}. \tag{9.78}$$

The control matrices for the fractional delay terms are

$$\begin{bmatrix} \mathbf{B}_d^{(w)} & \mathbf{B}_d^{(w+1)} \end{bmatrix} \Big|_{r=1.00T} \approx \begin{bmatrix} -0.71 & 2.15 & 0.00 & 0.00 \\ -2.18 & 2.91 & 0.00 & 0.00 \end{bmatrix} \tag{9.79}$$

$$\begin{bmatrix} \mathbf{B}_d^{(w)} & \mathbf{B}_d^{(w+1)} \end{bmatrix} \Big|_{r=1.25T} \approx \begin{bmatrix} -0.08 & 1.06 & -0.63 & 1.10 \\ -1.89 & 2.79 & -0.29 & 0.12 \end{bmatrix} \tag{9.80}$$

$$\begin{bmatrix} \mathbf{B}_d^{(w)} & \mathbf{B}_d^{(w+1)} \end{bmatrix} \Big|_{r=1.50T} \approx \begin{bmatrix} 0.31 & 0.18 & -1.02 & 1.70 \\ -1.31 & 2.12 & -0.87 & 0.80 \end{bmatrix}. \tag{9.81}$$

Figure 9.1 illustrates the convergence results of predictive feedback control and fractional feedback control under the three delays. When the delay aligns with the sampling interval (blue), the controllers are equivalent. When the delay is increased by 25% (orange), we immediately see that the standard controller converges slower. Lastly, if the true delay is increased by 50% (red),

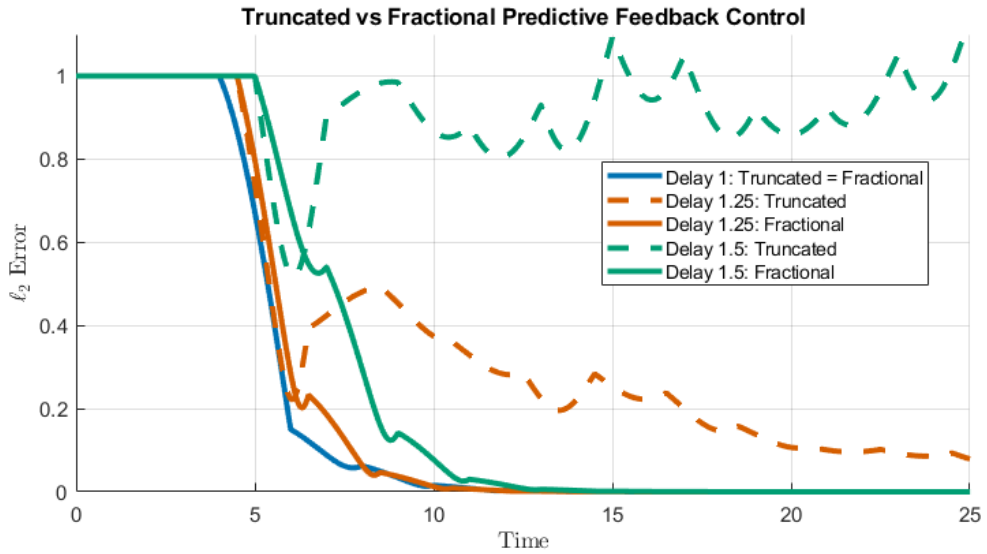

Figure 9.1: The truncated controller (dashed) is unable to stabilize the plant as the delay increases (**blue**, **orange**, to **green**). The fractions delay controller (solid) is able to stabilize the plant under all conditions.

the controller is unable to stabilize the plant.

9.6 Concluding Remarks

From the results of this work, a greater range of plants can be stabilized by accounting for input delay. Good approximations of the plant and delay allow predictive control systems to achieve performance similar to that for systems with zero delay. In this work, we showed that a fractional delay can be extracted from the discretized model provided by DMDidc. The method discussed here can naturally be combined with Differential Privacy and FedInfer to identify the dynamics of an input delayed system that is common to a set of distributed clients without compromising the participants privacy. Looking forward, the method could be adapted to learn delay bounds in order to limit the calculations needed for live updates. Furthermore, this method can potentially be used to account for packet drops by detecting, from data, how frequently information appears to be lost by the plant.

Chapter 10

Conclusions and Suggestions for Future Research

Privacy methods often sacrifice accuracy to increase the security of client data, limiting their effectiveness. In this dissertation, we presented contributions to the field of data privacy that narrow the gap in performance between private and non-private methods. In this chapter, we summarize our contributions and suggest directions for future research.

10.1 Differential Privacy

In Part I of this thesis, we presented the first in-depth study of the Symmetric alpha-Stable mechanism for data privacy. We contributed to the field in two notable ways. First, we proved that the SaS mechanism achieves pure-Differential Privacy. This is significant as the SaS density is closed under convolution, which allows the mechanism to be applied locally by clients prior to transmission to the server. In comparison, the only other mechanism with this property is the Gaussian mechanism, which only satisfies approximate-Differential Privacy. This result makes the SaS mechanism superior for applications involving Local Differential Privacy. Second, because the approximate form of Differential Privacy has superior composition results, we analyzed the privacy performance of the SaS mechanism in the context of (ε, δ)-DP. We showed through

numerical evaluations and simulation results that the SaS mechanism improves the privacy afforded to clients when compared with the Gaussian mechanism. In empirical evaluations on the MNIST, CIFAR-10, and Pediatric Pneumonia datasets, the SaS mechanism was shown to outperform the Gaussian mechanism by achieving higher testing accuracy for a given privacy budget. Notably, the SaS mechanism achieved just under a 20% accuracy improvement on CIFAR-10 with a privacy budget of $\varepsilon = 1$. In addition to these notable results, the neural network models employed were small when compared to the models used in many state-of-the-art results. We were constrained by the limited hardware used for training. Despite this, we were able to achieve results competitive with those obtained by larger models. For instance, the results on the CIFAR-10 dataset with privacy budget $\varepsilon = 1$ are state-of-the-art to the best of our knowledge. To better understand the magnitude of the advantage offered by the SaS mechanism, future work can employ this mechanism with larger Machine Learning models.

In this work, we arbitrarily choose $\alpha = 1.9999$ for the approximate-DP analysis in order to quickly and efficiently compare accuracy results with the Gaussian mechanism ($\alpha = 2$). While we showed that reducing α further increased the expected distortion, it should be noted that the change would additionally increase the level of privacy. As a result, for a fixed privacy budget ε, one could optimize the stability parameter α by minimizing the expected distortion. We note this as an avenue for future research.

Another possible research direction is in the development of systems that handle rare events. An emerging branch of robust control uses the mathematics of Differential Privacy to design controllers with bounded sensitivity to error in the state dynamics [237]. Our analysis of the heavy tailed Symmetric alpha-Stable mechanism could increase the robustness of these controllers by more accurately weighting the probability of rare events. Differential Privacy tech-

niques have been used to design game-theoretic solutions that avert strategic manipulation [238]. Thus, the analysis of rare events that follow a heavy tailed distribution may provide interesting results in the field of Mechanism Design [57].

An avenue of active research in the field of data privacy is Adversarial Classification, the online identification of adversaries. While Differential Privacy protects client data, the injected noise aids adversaries by misleading the classifier. Recent work has proposed countermeasures to mitigate this effect [239], but the analysis does not yet include the SaS mechanism.

One of the main drawbacks to Differential Privacy is that the injected noise necessarily distorts query responses. Current research efforts propose modifications to the definition of privacy to minimize this distortion. One notable modification is known as Individual Differential Privacy [240]. The definition in [240] offers the same level of privacy to individuals as the standard definition of DP, while reducing the overall level of distortion. An interesting direction for future work would be to examine the benefits of the SaS mechanism under this notion of privacy.

10.2 Federated Learning

While Federated Learning offers distinct advantages when it comes to client privacy, its adoption is hindered by the decreased model accuracy resulting from the constraints of a federated environment. To increase learning in this challenging environment, Federated Inference departs from conventional gradient-based methods by employing Bayesian Inference update steps. Our method offers several advantages over the traditional approach. First, it is inherently compatible with Differential Privacy, which is crucial for modern data protection. Second, we proved that FedInfer converges to the global optimum in the presence of system and statistical heterogeneity.

Its reliance on Bayesian Inference rather than gradient information from local updates makes FedInfer robust and reliable, enhancing the efficiency and effectiveness of Federated Learning.

Despite its strengths, FedInfer has limitations, particularly its assumptions about the structure of the Machine Learning model. The method is largely incompatible with complex models like Deep Neural Networks. Therefore, an avenue for further research could be employing FedInfer with Echo State Networks, a simpler type of Neural Network that relegates the learning process to a single linear layer. Another direction for future research would be the relaxation of simplifying assumption for the straggling clients. The current FedInfer analysis assumes that the rate at which clients return updates to the server is fixed and finite. An desirable update to our analysis could be to examine how the system handles infinite delay times (drop outs) as well as clients with variable return rates. One possible approach could be to sort clients into buckets based on update frequency and/or consistency.

We note a few several other interesting avenues for future research that directly extend the work on FedInfer. For example, one could investigating incorporating a measure of trust into the inference step of FedInfer. Developing a measure of trust for each participating client allows for the rejection of adversaries that attempt to poison the model during the learning process [51,241]. It is possible that FedInfer naturally extends to trust networks as the server could adjust the variance estimate of each client's update based on that client's trust measure. An additional direction is Incentive Design for Federated Learning systems. It is possible that rational clients may not have the same interests as the global network model [242, 243]. In these situations, the design of personalized, local models may be necessary to increase user participation. Because FedInfer models uncertainty, the method could be enhanced to provide incentives to participants that contribute less noisy results.

10.3 System Identification

Dynamic Mode Decomposition has proven itself to be a powerful tool for system identification. However, the discovered plant dynamics may be subject to controller input delay. Our method, DMDidc, solves this problem by simultaneously solving for the plant dynamics, control dynamics, and input delay. While these results improve the applicability of DMD, our analysis does not yet consider how DMD can incorporate privacy protecting methods such as Differential Privacy. Looking ahead, future research directions could analyze DMDidc through the lens of privacy by combining it with Differentially Private noise or by identifying system dynamics through a distributed dataset with Federated Learning.

10.4 Concluding Remarks

Data privacy has become an increasing concern in today's world. How we choose to handle and store our personal data profoundly affects our lives in many ways. While predictive models have the potential to enhance our quality of life, they demand ever-growing amounts of data for training. The research presented in this dissertation represents a step forward in the realm of data privacy. We hope that the methods and results of the dissertation will contribute to the development of essential predictive models, while protecting data privacy.

Bibliography

[1] DOMO. Data never sleeps 11.0, 2023. Available at: https://www.domo.com/solution/data-never-sleeps-11.

[2] *2021 Annual Data Breach Report*. Id Theft Center, Apr 2022.

[3] A. Holmes. 533 million facebook users' phone numbers and personal data have been leaked online, Apr 2021.

[4] D. Ramsbrock, R. Berthier, and M. Cukier. Profiling attacker behavior following ssh compromises. In *37th Annual IEEE/IFIP international conference on dependable systems and networks (DSN'07)*, pages 119–124. IEEE, 2007.

[5] US Federal Regulations. Internet Freedom 2011. *47 CFR § 8.1(a) 2011*.

[6] The European Union. Regulation (eu) 2016/679 (general data protection regulation), 2016.

[7] J. Koetsier. Privacy checkup: Limit ad tracking up 216% on ios, but down 85% on android, Jul 2021. https://www.singular.net/blog/limit-ad-tracking-privacy-checkup-in-2020/.

[8] L. Minto and M. Haller. Using federated learning to improve brave's on-device recommendations while protecting your privacy, Jun 2021.

[9] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems*, 10(05):557–570, 2002.

[10] Health insurance portability and accountability act of 1996. Public Law 104-191, 1996. Available at: https://www.govinfo.gov/app/details/PLAW-104publ191/PLAW-104publ191.

[11] California consumer privacy act. Assembly Bill No. 375, 2018. Available at: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375.

[12] California privacy rights act. Proposition 24, 2020. Available at: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=202320240AB2877&search_keywords=California+Privacy+Rights+Act.

[13] Brazilian general data protection law. Lei Geral de Proteção de Dados (LGPD), 2018. Available at: http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm.

[14] Florian Hartmann. Federated learning for firefox, Aug 2018.

[15] Differential privacy. Apple, 2017. Available at: https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf.

[16] K. O'Flaherty. Apple's privacy features will cost facebook $12 billion, Nov 2022. Available at: https://www.forbes.com/sites/kateoflahertyuk/2022/04/23/apple-just-issued-stunning-12-billion-blow-to-facebook/?sh=22e729761907.

[17] C. Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[18] H. B. McMahan, E. Moore, D. Ramage, and B. Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[19] Mastodon. Mastodon, 2016.

[20] A. Tsesis. The right to erasure: Privacy, data brokers, and the indefinite retention of data. *Wake Forest L. Rev.*, 49:433, 2014.

[21] G. Anthes. Data brokers are watching you, 2014.

[22] R. H. Weber. Internet of things – new security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30, 2010.

[23] P. Jain, M. Gyanchandani, and N. Khare. Big data privacy: a technological perspective and review. *Journal of Big Data*, 3:1–25, 2016.

[24] Electronic privacy information center (epic). epic. Available online: https://epic.org/.

[25] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

[26] George William Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.

[27] Michael Zeifman and Kurt Roth. Nonintrusive appliance load monitoring: Review and outlook. *IEEE transactions on Consumer Electronics*, 57(1):76–84, 2011.

[28] P.J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[29] C. Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

[30] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*, pages 486–503. Springer, 2006.

[31] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[32] Y. Lindell and E. Omri. A practical application of differential privacy to personalized online advertising. *Cryptology ePrint Archive*, 2011.

[33] H. Jiang, J. Pei, D. Yu, J. Yu, B. Gong, and X. Cheng. Applications of differential privacy in social network analysis: A survey. *IEEE transactions on knowledge and data engineering*, 35(1):108–127, 2021.

[34] E. Yilmaz, T. Ji, E. Ayday, and P. Li. Genomic data sharing under dependent local differential privacy. In *Proceedings of the twelfth ACM conference on data and application security and privacy*, pages 77–88, 2022.

[35] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng. Privacy-preserving federated brain tumour segmentation, 2019.

[36] T-H H. Chan, E. Shi, and D. Song. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):1–24, 2011.

[37] Jerome Le Ny and George J Pappas. Differentially private filtering. *IEEE Transactions on Automatic Control*, 59(2):341–354, 2013.

[38] S. L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[39] J. M. Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2867–2867, 2018.

[40] A. Cohen, M. Duchin, J.N. Matthews, and B. Suwal. Private numbers in public policy: Census, differential privacy, and redistricting. *Harvard Data Science Review*, (Special Issue 2), 2022.

[41] F. Fioretto, P. Van Hentenryck, and K. Zhu. Differential privacy of hierarchical census data: An optimization approach. *Artificial Intelligence*, 296:103475, 2021.

[42] A. Friedman and A. Schuster. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–502, 2010.

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

[44] Yi Liu, James J. Q. Yu, Jiawen Kang, Dusit Niyato, and Shuyu Zhang. Privacy-preserving traffic flow prediction: A federated learning approach. *arXiv*, abs/2003.08725, 2020.

[45] C. T. Kenny, S. Kuriwaki, C. McCartan, E. T.R. Rosenman, T. Simko, and K. Imai. The use of differential privacy for census data and its impact on redistricting: The case of the 2020 us census. *Science advances*, 7(41):eabk3283, 2021.

[46] J. Ficek, W. Wang, H. Chen, G. Dagne, and E. Daley. Differential privacy in health research: A scoping review. *Journal of the American Medical Informatics Association*, 28(10):2269–2276, 2021.

[47] Z. Lv and F. Piccialli. The security of medical data on internet based on differential privacy technology. *ACM Transactions on Internet Technology*, 21(3):1–18, 2021.

[48] A. Ziller, D. Usynin, R. Braren, M. Makowski, D. Rueckert, and G. Kaissis. Medical imaging deep learning with differential privacy. *Scientific Reports*, 11(1):13524, 2021.

[49] M. Adnan, S. Kalra, J. C. Cresswell, G. W. Taylor, and H. R. Tizhoosh. Federated learning and differential privacy for medical image analysis. *Scientific reports*, 12(1):1953, 2022.

[50] M. U. Hassan, M. H. Rehmani, and J. Chen. Differential privacy techniques for cyber physical systems: a survey. *IEEE Communications Surveys & Tutorials*, 22(1):746–789, 2019.

[51] A. Gholami, N. Torkzaban, and J. S. Baras. On the importance of trust in next-generation networked cps systems: An ai perspective. *arXiv preprint arXiv:2104.07853*, 2021.

[52] D. Durfee. Unbounded differentially private quantile and maximum estimation. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[54] L. Wasserman and S. Zhou. A statistical framework for differential privacy, 2009.

[55] Q. Geng and P. Viswanath. Optimal noise adding mechanisms for approximate differential privacy. *IEEE Transactions on Information Theory*, 62(2):952–969, 2015.

[56] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer, 2006.

[57] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 94–103. IEEE, 2007.

[58] C. Dwork and J. Lei. Differential privacy and robust statistics. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 371–380, 2009.

[59] C. Dwork, G. N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[60] C. Dwork and G. N. Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

[61] Alfréd Rényi. On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*, volume 4, pages 547–562. University of California Press, 1961.

[62] I. Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, August 2017.

[63] P. Kairouz, S. Oh, and P. Viswanath. The composition theorem for differential privacy. In *International conference on machine learning*, pages 1376–1385. PMLR, 2015.

[64] J. Murtagh and S. Vadhan. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*, pages 157–175. Springer, 2015.

[65] J. Dong, A. Roth, and W. J. Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

[66] S. Meiser and E. Mohammadi. Tight on budget? tight bounds for r-fold approximate differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 247–264, 2018.

[67] A. Koskela, J. Jälkö, and A. Honkela. Computing tight differential privacy guarantees using fft. In *International Conference on Artificial Intelligence and Statistics*, pages 2560–2569. PMLR, 2020.

[68] A. Koskela, J. Jälkö, L. Prediger, and A. Honkela. Tight differential privacy for discrete-valued mechanisms and for the subsampled gaussian mechanism using fft. In *International Conference on Artificial Intelligence and Statistics*, pages 3358–3366. PMLR, 2021.

[69] A. Koskela and A. Honkela. Computing differential privacy guarantees for heterogeneous compositions using fft. *arXiv preprint arXiv:2102.12412*, 2021.

[70] S. Gopi, Y. T. Lee, and L. Wutschitz. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34:11631–11642, 2021.

[71] S. Meiser. Approximate and probabilistic differential privacy definitions. *Cryptology ePrint Archive*, 2018.

[72] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 2512–2520. IEEE, 2019.

[73] B. Balle, G. Barthe, and M. Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences, 2018.

[74] M. Bun, C. Dwork, G. N. Rothblum, and T. Steinke. Composable and versatile privacy via truncated cdp. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 74–86, 2018.

[75] M. Bun and T. Steinke. Concentrated differential privacy: Simplifications. *Extensions, and Lower Bounds*, 2016.

[76] J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia. The limits of differential privacy (and its misuse in data release and machine learning). *Communications of the ACM*, 64(7):33–35, 2021.

[77] Y. Lindell. Secure multiparty computation (mpc). *Cryptology ePrint Archive*, 2020.

[78] R. Cramer, I. B. Damgård, and Nielesn J. B. *Secure multiparty computation*. Cambridge University Press, 2015.

[79] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78(110):1–108, 1998.

[80] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten. Crypten: Secure multi-party computation meets machine learning. *Advances in Neural Information Processing Systems*, 34:4961–4973, 2021.

[81] X. Li, R. Dowsley, and M. De Cock. Privacy-preserving feature selection with secure multiparty computation. In *International Conference on Machine Learning*, pages 6326–6336. PMLR, 2021.

[82] A. Sarmadi, H. Fu, P. Krishnamurthy, S. Garg, and F. Khorrami. Privacy-preserving collaborative learning through feature extraction, 2022.

[83] P. C. Weeraddana, G. Athanasiou, C. Fischione, and J. S. Baras. Per-se privacy preserving solution methods based on optimization. In *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC)*, pages 206–211, 2013.

[84] P. C. Weeraddana, G. Athanasiou, M. Jakobsson, C. Fischione, and J. S. Baras. On the privacy of optimization approaches. *arXiv preprint arXiv:1210.3283*, 2012.

[85] B. Alaya, L. Laouamer, and N. Msilini. Homomorphic encryption systems statement: Trends and challenges. *Computer Science Review*, 36:100235, 2020.

[86] Alexander Wood, Kayvan Najarian, and Delaram Kahrobaei. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Comput. Surv.*, 53(4), aug 2020.

[87] B. Li and D. Micciancio. On the security of homomorphic encryption on approximate numbers. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I 40*, pages 648–677. Springer, 2021.

[88] T. Ogilvie. Differential privacy for free? harnessing the noise in approximate homomorphic encryption. *Cryptology ePrint Archive*, 2023.

[89] R. J. Carroll and P. Hall. Optimal rates of convergence for deconvolving a density. *Journal of the American Statistical Association*, 83(404):1184–1186, 1988.

[90] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy, data processing inequalities, and statistical minimax rates, 2014.

[91] P. Billingsley. *Probability and measure*. A Wiley-Interscience publication. Wiley, 3. ed edition, 1995.

[92] P. Lévy. *Calcul des probabilités*. Gauthier-Villars, 1925.

[93] J. P. Nolan. *Univariate stable distributions*. Springer, 2020.

[94] C. Zawacki and E. Abed. The symmetric alpha-stable privacy mechanism. *arXiv preprint arXiv:2311.17789*, 2023.

[95] H. Bergström. On some expansions of stable distribution functions. *Arkiv för Matematik*, 2(4):375–378, 1952.

[96] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2023. Published electronically at `http://oeis.org`.

[97] M. Kwaśnicki. A new class of bell-shaped functions. *Transactions of the American Mathematical Society*, 373(4):2255–2280, 2020.

[98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[99] A. Krizhevsky. Learning multiple layers of features from tiny images. *Ph.D. dissertation*, 2009.

[100] D. S. Kermany, M. Goldbaum, W. Cai, C. C.S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, P. Jacqueline, M. Y. L. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zhang, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. N. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, and K. Zhang. Identifying medical diagnoses and treatable diseases by image-based deep learning. *cell*, 172(5):1122–1131, 2018.

[101] Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

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

[103] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[104] S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv preprint arXiv:2204.13650*, 2022.

[105] D. Kermany, K. Zhang, and M. Goldbaum. Large dataset of labeled optical coherence tomography (oct) and chest x-ray images, 2018.

[106] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[107] D. C Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 23(3):1622–1658, 2021.

[108] J. Wen, Z. Zhang, Y. Lan, Z. Cui, J. Cai, and W. Zhang. A survey on federated learning: challenges and applications. *International Journal of Machine Learning and Cybernetics*, 14(2):513–535, 2023.

[109] J. Shi, W. Wan, S. Hu, J. Lu, and L. Y. Zhang. Challenges and approaches for mitigating byzantine attacks in federated learning. In *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 139–146. IEEE, 2022.

[110] Simon Lacoste-Julien, Mark Schmidt, and Francis R. Bach. A simpler approach to obtaining an o(1/t) convergence rate for the projected stochastic subgradient method. *arXiv*, abs/1212.2002, 2012.

[111] R. Pathak and M. J. Wainwright. Fedsplit: An algorithmic framework for fast federated optimization. *CoRR*, abs/2005.05238, 2020.

[112] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. SCAFFOLD: stochastic controlled averaging for on-device federated learning. *CoRR*, abs/1910.06378, 2019.

[113] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data, 2019.

[114] Grigory Malinovsky, Dmitry Kovalev, Elnur Gasanov, Laurent Condat, and Peter Richtárik. From local SGD to local fixed point methods for federated learning. *arXiv*, abs/2004.01442, 2020.

[115] Zachary Charles and Jakub Konečný. On the outsized importance of learning rates in local update methods. *arXiv*, abs/2007.00878, 2020.

[116] Z. Charles and J. Konečný. Convergence and accuracy trade-offs in federated learning and meta-learning. *arXiv*, abs/2103.05032, 2021.

[117] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):5693–5700, Jul. 2019.

[118] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. When edge meets learning: Adaptive control for resource-constrained distributed machine learning. *CoRR*, abs/1804.05271, 2018.

[119] Hao Yu, Rong Jin, and Sen Yang. On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization, 2019.

[120] Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[121] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv*, abs/1812.06127, 2018.

[122] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *CoRR*, abs/2007.07481, 2020.

[123] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient, 2013.

[124] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[125] Aaron Defazio, Francis R. Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. *CoRR*, abs/1407.0202, 2014.

[126] Aaron Defazio and Leon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[127] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011.

[128] Heinz H. Bauschke and Patrick L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2019.

[129] Patrick L. Combettes. Monotone operator theory in convex optimization, 2018.

[130] Ernest K. Ryu and Stephen P. Boyd. A primer on monotone operator methods, 2015.

[131] D. W. Peaceman and H. H. Rachford, Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41, 1955.

[132] A. Mitra, R. H. Jaafar, G. J. Pappas, and H. Hassani. Achieving linear convergence in federated learning under objective and systems heterogeneity. *CoRR*, abs/2102.07053, 2021.

[133] Q. Li, B. He, and D. Song. Model-contrastive federated learning, 2021.

[134] X. Mu, Y. Shen, K. Cheng, X. Geng, J. Fu, T. Zhang, and Z. Zhang. Fedproc: Prototypical contrastive federated learning on non-iid data, 2021.

[135] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[136] Z. Dai, B. K. H. Low, and P. Jaillet. Federated bayesian optimization via thompson sampling. In *Advances in Neural Information Processing Systems*, volume 33, pages 9687–9699. Curran Associates, Inc., 2020.

[137] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[138] WILLIAM R THOMPSON. ON THE LIKELIHOOD THAT ONE UNKNOWN PROBABILITY EXCEEDS ANOTHER IN VIEW OF THE EVIDENCE OF TWO SAMPLES. *Biometrika*, 25(3-4):285–294, 12 1933.

[139] Kallista A. Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019.

[140] Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized federated learning with gaussian processes. In *Advances in Neural Information Processing Systems*, volume 34, pages 8392–8406. Curran Associates, Inc., 2021.

[141] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *arXiv*, abs/2006.07242, 2020.

[142] Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy, 2020.

[143] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning, 2020.

[144] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning, 2020.

[145] C. He, M. Annavaram, and S. Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. *Advances in Neural Information Processing Systems*, 33:14068–14080, 2020.

[146] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Róbert Ormándi, George E. Dahl, and Geoffrey E. Hinton. Large scale distributed neural network training through online distillation. *arXiv*, abs/1804.03235, 2018.

[147] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. *arXiv*, abs/1903.11341, 2019.

[148] Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks, 2018.

[149] Iou-Jen Liu, Jian Peng, and Alexander G. Schwing. Knowledge flow: Improve upon your teachers, 2019.

[150] SeongUk Park and Nojun Kwak. Feed: Feature-level ensemble for knowledge distillation, 2019.

[151] Ancong Wu, Wei-Shi Zheng, Xiaowei Guo, and Jian-Huang Lai. Distilled person reidentification: Towards a more scalable system. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1187–1196, 2019.

[152] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1285–1294, New York, NY, USA, 2017. Association for Computing Machinery.

[153] J. Zhang, S. Guo, X. Ma, H. Wang, W. Xu, and F. Wu. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems*, 34:10092–10104, 2021.

[154] Chengxi Li, Gang Li, and Pramod K Varshney. Decentralized federated learning via mutual knowledge transfer. *IEEE Internet of Things Journal*, 9(2):1136–1147, 2021.

[155] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32, 2019.

[156] B. Zhao, K. R. Mopuri, and H. Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[157] A. Wainakh, F. Ventola, T. Müßig, J. Keim, C. G. Cordero, E. Zimmer, T. Grube, K. Kersting, and M. Mühlhäuser. User label leakage from gradients in federated learning. *arXiv preprint arXiv:2105.09369*, 2021.

[158] W. Wei, L. Liu, M. Loper, K. Chow, M. E. Gursoy, S. Truex, and Y. Wu. A framework for evaluating gradient leakage attacks in federated learning. *arXiv preprint arXiv:2004.10397*, 2020.

[159] D. Scheliga, P. Mäder, and M. Seeland. Dropout is not all you need to prevent gradient leakage. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9733–9741, 2023.

[160] X. Zhou and X. Wang. A memory-efficient federated kernel support vector machine for edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[161] Divya G. Nair, C.V. Aswartha Narayana, K. Jaideep Reddy, and Jyothisha J. Nair. Exploring svm for federated machine learning applications. In *Advances in Distributed Computing and Machine Learning: Proceedings of ICADCML 2022*, pages 295–305. Springer, 2022.

[162] S. Bebortta, S. S. Tripathy, S. Basheer, and C. L. Chowdhary. Fedehr: a federated learning approach towards the prediction of heart diseases in iot-based electronic health records. *Diagnostics*, 13(20):3166, 2023.

[163] C. Koetsier, J. Fiosina, J. N. Gremmel, J. P. Müller, D. M. Woisetschläger, and M. Sester. Detection of anomalous vehicle trajectories using federated learning. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 4:100013, 2022.

[164] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

[165] C. Gentile and M. K. K. Warmuth. Linear hinge loss and average margin. *Advances in Neural Information Processing Systems*, 11, 1998.

[166] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[167] K. Nakajima and I. Fischer. *Reservoir computing*. Springer, 2021.

[168] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[169] W. Wang, Y. Tang, J. Xiong, and Y. Zhang. Stock market index prediction based on reservoir computing models. *Expert Systems with Applications*, 178:115022, 2021.

[170] K. Stanek. *Reservoir computing in financial forecasting with committee methods*. PhD thesis, M. Sc. Thesis in Engineering, Technical University of Denmark, Kongens . . . , 2011.

[171] S. Wakabayashi, T. Arie, S. Akita, K. Nakajima, and K. Takei. A multitasking flexible sensor via reservoir computing. *Advanced Materials*, 34(26):2201663, 2022.

[172] J. Platt, H. Abarbanel, S. Penny, A. Wong, and R. Clark. Robust forecasting through generalized synchronization in reservoir computing. In *AGU Fall Meeting Abstracts*, volume 2021, pages NG25B–0522, 2021.

[173] Rui Hu, Yuanxiong Guo, Hongning Li, Qingqi Pei, and Yanmin Gong. Personalized federated learning with differential privacy. *IEEE Internet of Things Journal*, 7(10):9530–9539, 2020.

[174] C. Gallicchio, A. Micheli, P. Barsocchi, and S. Chessa. User movements forecasting by reservoir computing using signal streams produced by mote-class sensors. In *Mobile Lightweight Wireless Systems: Third International ICST Conference, MOBILIGHT 2011, Bilbao, Spain, May 9-10, 2011, Revised Selected Papers 3*, pages 151–168. Springer, 2012.

[175] J. Moon, W. Ma, J. H. Shin, F. Cai, C. Du, S. H. Lee, and W. D. Lu. Temporal data classification and forecasting using a memristor-based reservoir computing system. *Nature Electronics*, 2(10):480–487, 2019.

[176] A. Jalalvand, B. Vandersmissen, W. De Neve, and E. Mannens. Radar signal processing for human identification by means of reservoir computing networks. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–6. IEEE, 2019.

[177] F. C. Bauer, D. R. Muir, and G. Indiveri. Real-time ultra-low power ecg anomaly detection using an event-driven neuromorphic processor. *IEEE transactions on biomedical circuits and systems*, 13(6):1575–1582, 2019.

[178] X. He, T. Liu, F. Hadaeghi, and H. Jaeger. Reservoir transfer on analog neuromorphic hardware. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 1234–1238. IEEE, 2019.

[179] F. Hadaeghi. Reservoir computing models for patient-adaptable ecg monitoring in wearable devices. *arXiv preprint arXiv:1907.09504*, 2019.

[180] Z. Ansari, F. Pourhoseini, and F. Hadaeghi. Heterogeneous reservoir computing models for persian speech recognition. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2022.

[181] H. Hauser. Physical reservoir computing in robotics. *Reservoir Computing: Theory, Physical Implementations, and Applications*, pages 169–190, 2021.

[182] P. Bhovad and S. Li. Physical reservoir computing with origami and its application to robotic crawling. *Scientific Reports*, 11(1):13002, 2021.

[183] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[184] M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer science review*, 3(3):127–149, 2009.

[185] D. Li, M. Han, and J. Wang. Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems*, 23(5):787–799, 2012.

[186] W. Ullah, T. Hussain, and S. W. Baik. Vision transformer attention with multi-reservoir echo state network for anomaly recognition. *Information Processing & Management*, 60(3):103289, 2023.

[187] S. Reza, M. C. Ferreira, J. JM Machado, and J. M. RS Tavares. A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks. *Expert Systems with Applications*, 202:117275, 2022.

[188] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[189] D. Bacciu, D. Di Sarli, P. Faraji, C. Gallicchio, and A. Micheli. Federated reservoir computing neural networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2021.

[190] C. Gallicchio. Sparsity in reservoir computing neural networks. In *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, pages 1–7. IEEE, 2020.

[191] B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007*, pages 471–482, 2007.

[192] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[193] S. J Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[194] K. B. Athreya and S. N. Lahiri. *Measure Theory and Probability Theory*, volume 19. Springer, 2006.

[195] I. A. Ibragimov and R. Z. Has'minskii. *Statistical Estimation: Asymptotic Theory*, volume 16. Springer, 2013.

[196] A. W. Drake. *Fundamentals of applied probability theory*. Mcgraw-Hill College, 1967.

[197] J. L Doob. Application of the theory of martingales. *Le calcul des probabilites et ses applications*, pages 23–27, 1949.

[198] L. Le Cam. The central limit theorem around 1935. *Statistical Science*, 1(1):78–91, 1986.

[199] A. DasGupta. *Asymptotic theory of statistics and probability*, volume 180. Springer, 2008.

[200] P. J. Bickel and K. A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics, vols. I-II package*. CRC Press, 2015.

[201] A. W. van der Vaart. *Bayes Procedures*, page 138–152. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998.

[202] M. J. Schervish. *Theory of statistics*. Springer Science & Business Media, 2012.

[203] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR, 2019.

[204] C. R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. In *Breakthroughs in Statistics: Foundations and basic theory*, pages 235–247. Springer, 1992.

[205] H. Cramér. A contribution to the theory of statistical estimation. *Scandinavian Actuarial Journal*, 1946(1):85–94, 1946.

[206] M. Fréchet. Sur l'extension de certaines évaluations statistiques au cas de petits échantillons. *Revue de l'Institut International de Statistique*, pages 182–205, 1943.

[207] R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222(594-604):309–368, 1922.

[208] R. Battiti. First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 4(2):141–166, 1992.

[209] Y. Cherruault. Convergence of adomian's method. *Kybernetes*, 18(2):31–38, 1989.

[210] E. Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 22(3):281–283, 1935.

[211] R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[212] W. N. Street, W. H. Wolberg, and O. L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. UCI Machine Learning Repository, 1993. Dataset available at: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

[213] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. Ch. Paschalidis, and W. Shi. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, 112:59–67, 2018.

[214] R. González-García, R. Rico-Martínez, and I.G. Kevrekidis. Identification of distributed parameter systems: A neural net based approach. *Computers & Chemical Engineering*, 22:S965–S968, 1998. European Symposium on Computer Aided Process Engineering-8.

[215] C. Paleologu, J. Benesty, and S. Ciochina. A robust variable forgetting factor recursive least-squares algorithm for system identification. *IEEE Signal Processing Letters*, 15:597–600, 2008.

[216] K.J. Åström and P. Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971.

[217] S. Chen, S.A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, 50(5):1873–1896, 1989.

[218] A. Tsiamis and G.J. Pappas. Finite sample analysis of stochastic system identification. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 3648–3654, 2019.

[219] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön. Deep learning and system identification. *IFAC-PapersOnLine*, 53(2):1175–1181, 2020. 21th IFAC World Congress.

[220] M.O. Williams, I.G. Kevrekidis, and C.W. Rowley. A Data-Driven Approximation of the Koopman Operator: Extending Dynamic Mode Decomposition. *Journal of NonLinear Science*, 25(6):1307–1346, December 2015.

[221] J.L. Proctor, S.L. Brunton, and J.N. Kutz. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161, 2016.

[222] H. Zhang, C.W. Rowley, E.A. Deem, and L.N. Cattafesta. Online dynamic mode decomposition for time-varying systems. *SIAM Journal on Applied Dynamical Systems*, 18(3):1586–1609, 2019.

[223] A.F. Snyder, D. Ivanescu, N. Hadjsaid, D. Georges, and T. Margotin. Stabilizing controller input delay effect on wide-area stability control. In *PowerTech Budapest 99. Abstract Records. (Cat. No.99EX376)*, pages 106–, 1999.

[224] O.J.M. Smith. A controller to overcome dead time. *ISA J*, 6(2):28–33, Feb 1959.

[225] D.L. Laughlin, D.E. Rivera, and M. Morari. Smith predictor design for robust performance. *International Journal of Control*, 46(2):477–504, 1987.

[226] H. Xing, J. Ploeg, and H. Nijmeijer. Smith predictor compensating for vehicle actuator delays in cooperative acc systems. *IEEE Transactions on Vehicular Technology*, 68(2):1106–1115, 2019.

[227] C. C. Zawacki and E. H. Abed. Dynamic mode decomposition for control systems with input delays. *IFAC-PapersOnLine*, 56(3):97–102, 2023.

[228] W.J. Rugh. *Linear system theory*. Prentice-Hall, Inc., 1996.

[229] J. Uhlmann. Unit consistency, generalized inverses, and effective system design methods. *arXiv preprint arXiv:1604.08476*, 2016.

[230] E.F. Camacho and C.B. Alba. *Model predictive control*. Springer science & business media, 2013.

[231] W.J. Culver. On the existence and uniqueness of the real logarithm of a matrix. *Proceedings of the American Mathematical Society*, 17(5):1146–1151, 1966.

[232] K.S. Narendra and A.M. Annaswamy. Persistent excitation in adaptive systems. *International Journal of Control*, 45(1):127–160, 1987.

[233] M. Verhaegen. Identification of the deterministic part of mimo state space models given in innovations form from input-output data. *Automatica*, 30(1):61–74, 1994.

[234] A. Marcet and T.J Sargent. Convergence of least squares learning mechanisms in self-referential linear stochastic models. *Journal of Economic theory*, 48(2):337–368, 1989.

[235] Y. Pan, T. Sun, and H. Yu. On parameter convergence in least squares identification and adaptive control. *International Journal of Robust and Nonlinear Control*, 29(10):2898–2911, 2019.

[236] K. Nassiri-Toussi and W. Ren. On the convergence of least squares estimates in white noise. *IEEE Transactions on Automatic Control*, 39(2):364–368, 1994.

[237] V. Pacelli and A. Majumdar. Robust control under uncertainty via bounded rationality and differential privacy. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3467–3474. IEEE, 2022.

[238] L. Zhang, T. Zhu, P. Xiong, W. Zhou, and P. S. Yu. More than privacy: Adopting differential privacy in game-theoretic mechanism design. *ACM Computing Surveys (CSUR)*, 54(7):1–37, 2021.

[239] J. Giraldo, A. Cardenas, M. Kantarcioglu, and J. Katz. Adversarial classification under differential privacy. In *Network and Distributed Systems Security (NDSS) Symposium 2020*, 2020.

[240] J. Soria-Comas, J. Domingo-Ferrer, D. Sánchez, and D. Megías. Individual differential privacy: A utility-preserving formulation of differential privacy guarantees. *IEEE Transactions on Information Forensics and Security*, 12(6):1418–1429, 2017.

[241] S. Agreste, P. De Meo, E. Ferrara, S. Piccolo, and A. Provetti. Trust networks: Topology, dynamics, and measurements. *IEEE Internet Computing*, 19(6):26–35, 2015.

[242] S. Kim. Incentive design and differential privacy based federated learning: A mechanism design perspective. *IEEE Access*, 8:187317–187325, 2020.

[243] B. Luo, Y. Feng, S. Wang, J. Huang, and L. Tassiulas. Incentive mechanism design for unbiased federated learning with randomized client participation, 2023.